FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

A Vision-Based System for Fingertip Detection on Tracked Interactive Surfaces

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Medieninformatik

eingereicht von

Markus Autengruber
Matrikelnummer 0255571

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung
Betreuer: Priv.-Doz. Mag. Dr. Hannes Kaufmann
Mitwirkung: Res.-Ass. Michael Mehling

Wien, 11.11.2010

(Unterschrift Verfasser) (Unterschrift Betreuer)

Erklärung

Markus Autengruber
Markgraf-Rüdiger-Straße 24/26
1150 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, am 11.11.2010

Vorwort & Danksagung

Die vorliegende Diplomarbeit enstand im Rahmen meines Masterstudiums der Medieninformatik an der Technischen Universität Wien – am Institut für Softwaretechnik und Interaktive Systeme (E188), innerhalb der Arbeitsgruppe Interactive Media Systems Group (E188/2). Ich möchte mich an dieser Stelle bei all jenen Menschen bedanken, die mir durch ihre kräftige Unterstützung dabei geholfen haben diese Arbeit zu erstellen.

Mein erster Dank gilt Herrn Prof. Hannes Kaufmann (Interactive Media Systems Group, Technische Universität Wien), der sich dazu bereit erklärt hat meine Diplomarbeit zu betreuen und mit dem gemeinsam das Thema dieser Arbeit herausgearbeitet wurde.

Mein ganz besonderer Dank richtet sich an Herrn Michael Mehling (Interactive Media Systems Group, Technische Universität Wien), der mich den gesamten Verlauf meiner Arbeit begleitet hat und mir stets mit Rat und Tat zur Seite stand. Nicht zuletzt durch seine freundliche, hilfsbereite und kollegiale Betreuung hat mir diese Arbeit sehr viel Freude bereitet.

Ich danke Herrn Julien Letessier (HiLabs, Grenoble) für den überaus freundlichen Kontakt und dem Bereitstellen von weiterführenden Informationen zu dem Paper das einen wesentlichen Grundstein dieser Arbeit bildet.

Weiters bedanke ich mich bei Fernando Vilariño (Computer Vision Center, Barcelona), der schon früher in meinem Studium – während eines Projektpraktikums – mein Interesse an Maschinellem Sehen geweckt hat und bei dem ich viel über wissenschaftliches Arbeiten lernen konnte.

Mein ganz besonderer Dank gilt abschließend meinen Eltern, die mir meine Ausbildung ermöglicht und in jeglicher Hinsicht die Grundsteine für diesen Weg gelegt haben.

Zusammenfassung

Ein wesentlicher Trend der Mensch-Computer-Interaktion der letzten Jahre beschäftigt sich mit der Entwicklung großer, berührungsempfindlicher Interaktionsflächen zur simultanen Benützung durch mehrere Anwender. Ein Forschungsschwerpunkt liegt hierbei auf dem Aufbau einer entsprechenden Schnittstelle, die den Benutzern die direkte Manipulation von digitalen Inhalten ermöglichen soll. Besonders relevant sind hierfür das Design der interaktiven Oberfläche sowie die automatische Erkennung von Fingerspitzen die sich auf und über dieser Fläche bewegen.

Allgemein verbreitete Technologien zur Erkennung von Fingerspitzen und Oberflächenkontakten beruhen entweder auf der optischen Ablenkung von Lichtstrahlen in einer kontrollierten Umgebung oder auf kapazitiver Oberflächentechnologie. Aufgrund des für solche Techniken speziell benötigten Hardware-Aufbaus, meist bestehend aus individuell angefertigten Komponenten, sind entsprechende Interaktionssyteme in der Regel teuer und zumeist auch eingeschränkt in ihrer Portabilität und Skalierbarkeit. Etliche Forscher im Bereich der „*Multi-Touch*“-Interaktion arbeiten daher an der Entwicklung alternativer Schnittstellen, welche einzig und allein auf herkömmlicher und kostengünstiger Video-Hardware beruhen. Zur Erkennung von Fingerspitzen auf meist natürlichen Oberflächen bedient man sich hier moderner Algorithmen des Maschinellen Sehens.

Die vorliegende Arbeit widmet sich zunächst der Beschreibung, Diskussion und Evaluierung gängiger berührungsempfindlicher Oberflächentechnologien sowie relevanten Ansätzen beruhend auf Maschinellem Sehen. Das Hauptaugenmerk liegt dann auf der Auswahl geeigneter Bildverarbeitungs- und Mustererkennungsalgorithmen und der Entwicklung einer einfachen Software-Applikation zum *Tracking* einer rechteckigen, bewegbaren Oberfläche sowie zur Erkennung von Fingerspitzen die sich über diese Fläche hinweg bewegen. Die Arbeit wird abgerundet durch die Präsentation relevanter empirischer Resultate bezüglich des Erkennens von rechteckigen Flächen sowie Fingerspitzen in einem natürlichen Innenraum anhand von visueller Bildinformation.

Abstract

Multi-touch sensing on interactive tabletops and other flat surfaces has become a major trend in the field of human-computer interaction over the past years. The main objective is to provide a touch interface for the direct manipulation of digital content by multiple users at the same time. Within these terms the appropriate design of the interactive surface as well as the automatic detection and tracking of fingertips are crucial.

Popular techniques for fingertip and touch detection use specific contact-sensitive computer hardware that is either relying on optical sensing in a controlled environment or capacitive surface technology. Since such hardware is usually custom-made, those interaction systems are mostly expensive, inconvenient to move, install and operate and not scalable. To overcome these drawbacks, a number of multi-touch researchers strive for alternative techniques to provide more adjustable interfaces. Here, everyday surfaces shall be augmented with the functionality of touch-based user interfaces, while using none but off-the-shelf and affordable vision hardware and relying on state-of-the-art computer vision methods and algorithms.

This work starts off with the description, discussion and evaluation of common surface hardware technologies as well as existing techniques based on simple video hardware. After that, a set of straightforward computer vision algorithms is selected in order to develop a stand-alone software application. The application is capable of continuously tracking a rectangular surface as well as detecting multiple fingertips that hover above its top. This work is concluded by providing relevant empirical results on computer vision-based rectangle and fingertip detection in natural indoor environments.

Contents

List of Figures

List of Tables

CHAPTER 1

Motivation & Problem Statement

Multi-touch interaction on tabletops and other flat surfaces has become a major trend in human-computer interaction research over the past years. The objective of this field is to provide an interface based on an interactive surface for the detection of manual user input. For this purpose, no further input device, such as a computer mouse, keyboard or stylus, is intended. Each manipulation is performed by the user directly by hand. Basically, the multi-touch interface needs to be capable of recognizing finger touches on physical surfaces. In comparison to standard input devices, multi-touch interfaces provide a basic advantage in usability and interactivity. Generally, the direct form of manipulation is considered to be more natural and intuitive to the user. This is promising for application domains, where pointing at and selecting and dragging of digital content by multiple users is requested. Corresponding fields of application are found in computer-supported collaborative work environments and large kiosk terminals.

Over the years, numerous multi-touch interfaces have been presented to the public and currently both commercial and non-commercial interaction systems are available. Those systems usually consist of special hardware setups that provide touch-sensitive interactive surfaces. The most popular technologies used here, amongst others, are optical-based solutions, such as Frustrated Total Internal Reflection, Diffused Illumination (e.g. Front Diffused Illumination and Rear Diffused Illumination), Light Plane illumination (e.g. Laser-Light Plane and LED-Light Plane illumination), Diffused Surface Illumination and capacitive surface technology. Consisting of non-standard computer hardware (e.g. infrared light sources and cameras, radio-frequency transmitters, etc.), those setups are usually expensive, complicated to build and in the majority of all cases rather bulky and heavy. Typically, once such an interaction system is built up, it is limited to a certain size. Hence, it cannot easily be rescaled to specific application needs. Portability of the final system is affected as well. Therefore, such systems are inconvenient to use, especially when trying to operate large interactive surfaces.

That is why alternative methods, which are totally based on simple vision hardware, have become more and more popular among multi-touch researchers. Here, all of the detection and tracking tasks are performed by software algorithms applied to images obtained by computer video cameras. Those approaches do usually not require bulky hardware setups or specially equipped surfaces. Generally, standard video cameras are sufficient. This leads to many advantages. For example, the interactive surface may be sized individually (e.g. different interactive surfaces may be used with the same application) and even relocated dynamically at runtime (e.g. interactive surfaces may be used as portable control panels for other applications). Furthermore, such systems can be set up in a much shorter amount of time and used almost everywhere, if lighting conditions do allow for it. Altogether, a major advantage in both scalability and portability of the final multi-touch interaction system is recorded. Basically, any arbitrarily shaped sheet, panel, table or even a regular computer display may be used as an interactive surface. Unfortunately, these approaches do have disadvantages as well. Due to the use of vision hardware and relying on digital images, illumination issues have to be regarded wisely. The main problem is, in contrast to the previously mentioned hardware-based approaches, that environmental conditions cannot be controlled as in, for example, a self-contained rear projection tabletop system. Different light sources at varying spatial positions, shadows and other visual artifacts need to be regarded. The segmentation of the scene is more complex than in hardware-based setups, where these conditions are generally consistent over time. Moreover, many computer vision algorithms are rather complex and computational performance is an issue as well. Robustness, autonomy and usability of the final interaction system are affected.

The vision-based[1] system presented in this work shall be capable of detecting and tracking an individually designed and dynamically moving, rectangular interactive surface as well as fingertips that hover above the surface's top. Any detection and tracking tasks should be performed under indoor lighting conditions and altogether the system should be a convenient alternative to heavy and bulky approaches, which use specific touch-sensitive hardware. Especially for large interactive surfaces and when portability, scalability and cost are important. The final multi-touch interaction system should perform autonomous to a large extent, while demanding only a few parameter changes at runtime.

[1]For the sake of simplicity in this work, the phrase vision-based refers to setups that are exclusively based on standard computer vision hardware. This does not include optical-based touch-sensitive hardware technologies, although those are relying on (more elaborated) vision hardware as well.

CHAPTER 2

Introduction

Ever since humans (i.e. users) utilize information processing machines (i.e. computers) in order to accomplish various tasks, interfaces need to be provided at which the interaction process between users and computers may occur. These interfaces are generally based on both software components (i.e. user interfaces) and peripheral computer hardware (i.e. input devices). Human-computer interaction (HCI) research strives to study this process and is often regarded as an intersection of computer science and behavioral science. HCI is sometimes also referred to as computer-human interaction (CHI) or man-machine interaction (MMI). Due to the fact that HCI interfaces are software- and hardware-based, generally both software and hardware design principles need to be considered in conjunction when thinking about HCI interfaces. On the human side of HCI, this includes social sciences, cognitive psychology, communication theory and linguistics. Whereby on the technology side, electronics, electro-technics, computer science, interaction design and industrial design are regarded. [Wik10c]

This work focuses on the examination and comparison of popular multi-touch technologies based on specific touch-sensitive hardware, the presentation of a vision-based alternative and its implementation by the use of modern computer vision algorithms. Concerns belonging to the adequate design of user interface software components are not part of this work.

In the following, the multi-touch interface for HCI is introduced in section 2.1 and the corresponding field of application is pointed out (section 2.2). After explaining necessary requirements, restrictions and constraints in section 2.3, the current state of the art of multi-touch interaction systems is presented in section 2.4. Chapter 3 is dedicated to the description of related work and theoretical foundations, where commonly used technologies for multi-touch interaction on tabletops and other large, flat surfaces are discussed in detail. After that, a multi-touch HCI interface based on standard vision hardware is presented in chapter 4 and implementation details, including the introduc-

tion of computer vision approaches for surface tracking and fingertip detection, are explained extensively. Moreover, experimental and practical results are discussed later on. This work is then completed by chapter 5 and providing final conclusions and an outlook on possible future work.

2.1 The Multi-Touch Interface

Generally, computer input devices enable users to provide data and control signals to computers for the purpose of entering various kinds of information and controlling graphical user interface (GUI) components. Input devices may be distinguished according to:

- the modality of the provided data (e.g. audio/video data, etc.),
- the continuity of the provided control signal (i.e. discrete or continuous signal) and
- the degree of freedom (i.e. the dimensionality of the provided data). [Wik10d]

For the understanding of touch-based systems (i.e. touch systems) within the context of computer input devices, the knowledge of computer keyboards, pointing devices (e.g. computer mice) and video input devices (e.g. webcams) is beneficial. Touch systems are commonly operated by using a stylus or one or more fingertips for providing manual user input on a so-called interactive surface.

Computer Keyboards

Computer keyboards (or simply keyboards) are basic hardware components, where mechanical keys are used for entering one-dimensional input data and providing discrete control signals to the computer. Moreover, physical response is provided to the user (e.g. if a key is pressed or released).

For the purpose of entering data on interactive surfaces, so-called virtual keys and projected keyboards have been presented during the past years. Virtual keys are GUI components (cp. figure 2.1), which simulate regular computer keyboards on-screen. By the use of a pointing device, virtual keys may be utilized just as their mechanical equivalents. Projected keyboards are visual artifacts (cp. figure 2.2), which are projected onto flat surfaces, such as tabletops, for example. Those usually take part in a vision-based system, which is watching for manual user input along the projected region. It can be easily understood that a major advantage of both virtual keys and projected keyboards is that they do not actually require any physical space and may be faded in and out, whenever they are needed or not. Due to their virtual nature, character layouts and

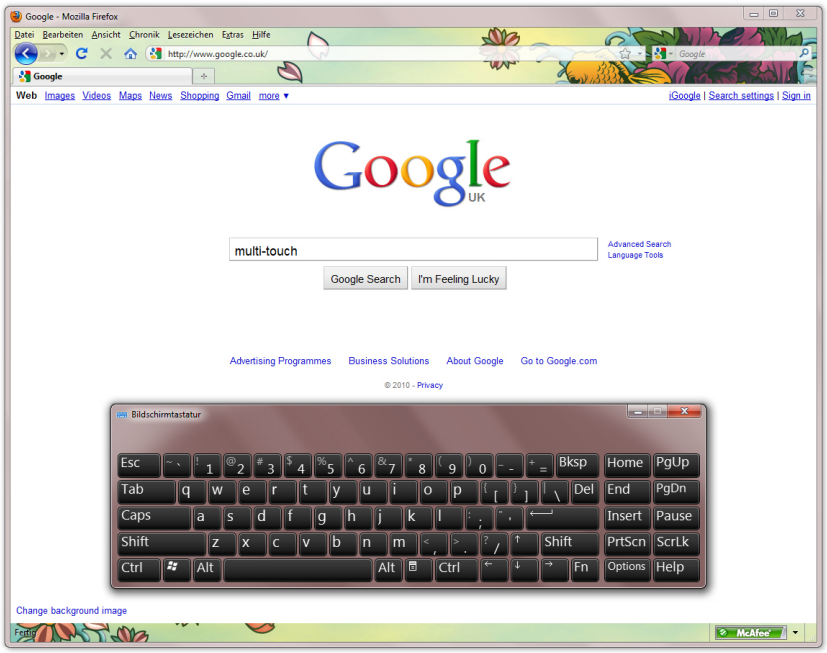


Figure 2.1: The term virtual keys refers to simple GUI components that simulate the mechanical keyboard on-screen.

graphical designs can be switched quickly, whereas regular computer keyboards are initially predefined and are not meant to be converted afterwards. A big disadvantage of virtual keys and projected keyboards is the lack of physical response, which may result in bad usability.

Pointing Devices

Pointing devices (e.g. computer mice, trackpads, trackballs, etc.) are hardware components, which provide multi-dimensional (i.e. spatial) input data and continuous control signals to the computer. These signals are widely used for controlling computer software via GUI components. Here, HCI is mostly performed by using simple manual gestures, such as pointing, clicking and dragging, which are directly delegated to the cursor, virtually representing the spatial position of the pointing device within the GUI. Pointing devices may be classified according to:

- the kind of input (i.e. direct or indirect input) and
- the type of the provided spatial data (i.e. absolute or relative spatial position). [Wik10d]

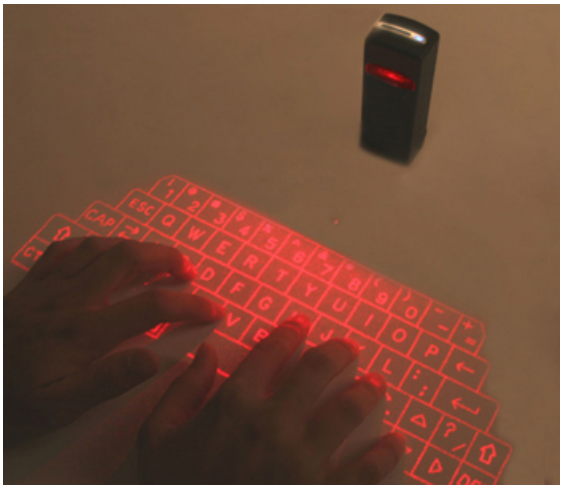


Figure 2.2: Projected keyboards are artificial visual artifacts that are typically front-projected onto tabletops or similar flat surfaces. Source: `http://www.netlingo.com/imagearchive/5650_virtualkeyboard_word_large.jpg`

Conventional pointing devices, such as computer mice (or simply mice), provide indirect input. Here, the physical device is not spatially collocated with the display device and may even be moved, while not receiving any input signals (e.g. when a mouse is lifted and moved to another position without sensing the ground). The actual position of the pointing device does not coincide with the position of the cursor and the corresponding spatial information is considered as relative. In comparison to that, manual interaction with touch systems provides direct user input. The established spatial position of the stylus or fingertip coincides with the actual position of the cursor within the GUI. Therefore, the corresponding spatial information is absolute.

Over the years, so-called multi-touch types have evolved from touch interfaces, which allow two or more simultaneous manual input. Those systems usually take fingertips rather than multiple styluses in order to offer more complex and intuitive manipulations. To this end, elaborated manual gestures (e.g. for scrolling, zooming in/out, etc.) have been developed. Figure 2.3 illustrates some examples from the multi-touch trackpad used in Apple's MacBook laptop computer. The movement of the fingertips is analyzed by gesture detection algorithms, which are situated at the software side of the HCI interface and hand over adequate signals to the GUI. Nowadays, so-called multi-touch gestures are pretty much common in modern HCI interfaces, such as trackpads and multi-touch displays or tabletop systems. In this work, only multi-touch systems based on fingertip input are regarded. It has to be mentioned that the term multi-touch may either refer to systems taking multi-touch gestures by one user or others allowing such gestures from multiple users (i.e. multi-user interaction).

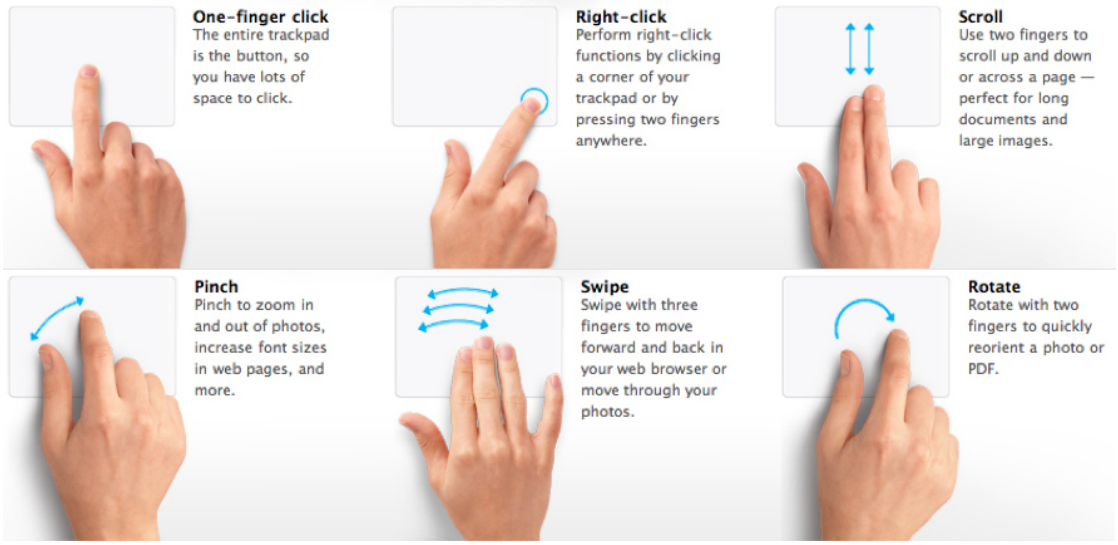


Figure 2.3: Multi-finger gestures supported by the Apple multi-touch trackpad. Source: `http://www.appleiguide.co.uk/iGuide/Tips,_Tricks_&_News/Entries/2010/5/3_Multi-Touch_your_MacBook_files/Gestures.png`

Video Input Devices

Video input devices, such as (infrared) video cameras, stereo vision pairs and since recently time-of-flight (TOF) cameras, provide continuous visual data. In the terms of touch-based interaction systems, video cameras may be used for processing images of the scene for the purpose of visually detecting and tracking various physical objects (e.g. fingertips, visual markers).

Many popular multi-touch setups that rely on optical sensing use special infrared cameras in conjunction with infrared light sources. Usually, the infrared spectrum of light is blocked to the lense of the camera, so that only visible light can pass. For the purpose of object tracking under infrared lighting conditions, exactly the opposite is required. Infrared-pass filters are used in front of the lense to block all but infrared light. This leads to further image segmentation and object detection opportunities. A number of relevant techniques based on infrared lighting are described in the following chapter 3 to common touch technologies.

Computer stereo vision, similar to human binocular vision, provides two different views of the scene at the same time. The two views are brought into correspondence by calculating a certain epipolar geometry that is used for obtaining depth information of the scene. A number of vision-based techniques use stereo matching for detecting surface touch and hover events. A selection is described in the following chapter.

TOF cameras are generally based on the measurement of time it takes for a certain object (e.g. light wave) to travel a distance through a medium (e.g. air). Here, the scene is illuminated by light (e.g. laser) impulses and the time delay that occurs until light rays are being reflected by nearby objects is measured. This delay is directly proportional to the actual object distance in space. TOF cameras generally provide low resolution (e.g. 320-by-240 pixels or lower) but rather high frame rates of up to 100 frames per second (abbr.: fps). [Wik10h] To this end, there is no popular vision-based multi-touch sensing approach using TOF cameras.

Multi-Touch Interface – Summary

Multi-touch interaction systems are hardware-based interfaces for HCI, where multi-touch gestures by a single user or simultaneous input by multiple users are allowed. The main objective of multi-touch interaction systems on tabletops or other flat surfaces is to provide space for collaborative work, together with allowing direct manual user input on large interactive surfaces. Artificial visual artifacts, such as virtual keys or projected keyboards, support this task.

2.2 Applications

Over the past years, multi-touch interaction systems have become more and more robust and accurate on the one hand and less error-prone on the other hand. Due to the support of multi-user interaction, such systems are nowadays widely used for collaborative work assignments in business and research situations. That is why many applications may be found in the domains of computer-supported collaborative work (CSCW). In the terms of CSCW, multi-touch tabletop systems are used for enhancing interaction processes between co-workers and improving communication. Generally, this means viewing and manipulating digital content on an interactive surface, where multiple users are allowed to perform various tasks simultaneously. For instance, this can be showing content (e.g. images, videos, etc.) to colleagues or working together on the same task (e.g. collaboratively sorting images according to a certain scheme). Another field of application for multi-touch interaction systems are kiosk and terminal applications and also multi-user computer games and tangible user interfaces.

Multi-touch interaction systems are often used when digital content is presented to a large amount of people and manipulated by one or more of them at the same time. Interactive surfaces are increasingly common for presentational purposes, including news, weather and sport broadcasts.

2.3 Requirements & Constraints

When designing multi-touch interaction systems consisting of relatively large interactive surfaces, such as tabletops or any other large, flat surfaces, there are certain requirements and restrictions to the hardware setup and used materials. In the following, basic preliminaries are examined by providing a brief outlook on touch-sensitive surface hardware. After that, relevant constraints are introduced.

2.3.1 Interactive Surface

The interactive surface of a touch-based interaction system represents the basic interface for detecting manual user input. In setups based on touch-sensitive hardware, the surface is typically made out of many layers of special materials, like acrylic and silicone, whereas each fulfills a specific purpose. In many cases, the materials are expensive and need to be specially prepared. Especially when building large interactive surfaces, such as tabletop systems or multi-touch walls, there are limitations to the configuration. For example, the acrylic material, which is typically used for the main panel of the interactive surface, needs to provide a certain thickness not to get bent when users are touching and pressing against it. Furthermore, the acrylic panel needs to be very clear not to inhibit the incidence of light beams, as it is crucial in optical-based hardware setups, for example. Commonly, once such a system is built-up, it is limited to a certain size and cannot be rescaled by any means. This generally affects the scalability of the whole system. Another important issue of hardware-based setups is their heaviness and bulkiness. Usually, a special frame is used to hold the surface structure and other hardware components. Portability of the interaction system is limited. Moreover, many optical-based systems use back projection for providing visual feedback to the user(s). Back projection either requires a certain depth behind (or below) the interactive surface or wide angle lenses for the projector and camera devices.

Vision-based techniques that rely on standard video cameras and computer vision algorithms instead of touch-sensitive hardware commonly allow more scalable and as well more portable systems. Basically, any flat surface or even a regular computer display may be used as an interactive surface and does not necessarily have to be specially prepared. For example, a regular panel, tabletop or wall may be used, as visual feedback is typically front-projected. Nevertheless, there are restrictions to the interactive surface here as well. Basically, the surface needs to be designed in a way that it can be easily segmented from the distracting background in the scene. Quite often, a rectangular, flat, white sheet or panel or the bright surface of a tabletop is used. The size of the interactive surface can be restricted as well, due to the limited viewing angle of the camera. Commonly, the interactive surface is fixated and may not be moved during runtime, because most systems are not able to track a moving surface.

2.3.2 Constraints

In order to describe, evaluate and compare different approaches, significant constraints have to be defined. Below, some relevant specifications for failure assessment and usability in multi-touch interaction systems are provided.

Accuracy

Accuracy generally describes a system's degree of closeness to the actual result, which is expected in a perfect system under perfect conditions [Wik10a]. As touch-based interfaces provide two-dimensional spatial positions of manual user manipulations, accuracy in this context may be regarded as the difference between the actual and the detected position of the object that is used to perform the input. Here, accuracy can be expressed in millimeters (abbr.: mm). Optimal values depend on the size of the interactive surface as well as the size of the single GUI elements. Typically, touch-based interfaces need to provide accuracy at level of a few millimeters, because otherwise the usability of the system may be affected.

Other reasonable measurements for accuracy, regarding correct and incorrect user input detection are described in the following:

- True positive (TP): A user input that is correctly detected as such.
- False positive (FP): A detection, while no user input actually occurs.

Robustness

In the terms of software and hardware design, robustness describes the ability of a system to withstand upcoming errors and discontinuities at runtime [Wik10g]. In multi-touch interaction, robustness usually describes the capability of the system to sustain a certain continuity against varying conditions. These may either refer to general conditions affecting usually all multi-touch systems (e.g. different fingertip or hand sizes, variable skin color, hand occlusions, etc.) or specific types, such as varying lighting conditions and shadows, that primarily affect optical- and vision-based approaches.

Latency

Typically, the term latency refers to a measurement of time delay perceived in a system and is measured in milliseconds (abbr.: ms) [Wik10e]. In the context of multi-touch interaction systems, latency describes the delay of time between the occurrence of a user manipulation at the hardware interface and the actual receiving of the corresponding control signal at the software side of the interface. Small values represent good, whereas

great values denote bad latency of the system. In multi-touch interfaces, latency should not be higher than 50 ms in order to provide good usability [WB94].

Usability

Whether the usability of an interface is considered to be good or bad is a matter of the system's capabilities and depends on both software and hardware components as well as the ability of the user herself [Wik10i]. The term usability basically describes how intuitive the interaction process of the HCI interface is. In multi-touch systems, interaction is commonly performed by using one hand but bi-manual interaction may be used as well. Due to the lack of other peripheral devices, multi-touch interaction on large interactive surfaces is considered to be natural and intuitive. Usability is generally influenced by the previously mentioned parameters (i.e. accuracy, robustness and latency).

Autonomy

Autonomy describes the ability of a system to perform without any further parameter adaptation at runtime required by the user. In multi-touch systems, this refers to automatic parameter estimation approaches. If a system performs well (i.e. with a certain amount of accuracy) under various conditions, it may be considered autonomous.

2.4 State of the Art

Over the past years, numerous different multi-touch interfaces and stand-alone interaction systems have been presented to the public and by now a number of both commercial and non-commercial systems are available. In the following, the current state of the art of proprietary multi-touch tracking devices, multi-user interaction systems and open technologies for multi-touch sensing in tabletop-like environments is presented. Furthermore, a collection of relevant scientific approaches is mentioned.

2.4.1 Proprietary Systems

When the Apple iPhone was initially launched in 2007 [Hon07], multi-touch interfaces gained popularity among the broad public and virtually immediately many other manufacturers of mobile phones and smartphones, such as Samsung and HTC, redesigned the idea and presented own multi-touch interfaces. Even though Apple often describes itself as being the inventor of multi-touch technology, the term multi-touch already occurred back in the early eighties, when relevant research initially began [Bux09].

The first touchpad for laptop computers allowing multi-touch gestures has been introduced with the launch of the Apple MacBook Air in 2008 [Blo08]. Today, many

Figure 2.4: Multiple views of the Apple iPad. Courtesy of Apple.com [App10a].

different manufacturers provide proprietary multi-touch interfaces in smartphones and portable computer systems like the Apple iPad (cp. figure 2.4) [App10a] and the Linux-based WeTab (previously known as WePad) by the German Manufacturer Neofonie (cp. figure 2.5) [Wet10]. In 2010, Apple introduced a peripheral input device called the Magic Trackpad (cp. figure 2.6) [App10b], which allows multi-touch gestures on Macintosh desktop computers.

Commercial systems for multi-user interaction on tabletops commonly consist of ready-made hardware setups, which can be purchased and used out-of-the-box. Here, the interactive surface, which is mostly based on a display or rear projection device, is mounted on a table or frame to enable multiple users to interact in a collaborative manner. Popular manufacturers of such systems are Microsoft (e.g. Microsoft Surface), GestureTek (e.g. GestureTek Illuminate Multi-Touch Table) and Circle Twelve Incorporated (e.g. DiamondTouch).

2.4.1.1 Microsoft Surface

Microsoft Surface (cp. figure 2.7) is a stand-alone, fully functional computer system for simultaneous multi-user interaction. The hardware interface is based on a 30 inch display device, which is horizontally mounted on an enclosed rack. Special Infrared cameras and image recognition in the infrared spectrum are used to recognize fingertips that are touching or moving above the display surface. The system allows visually tagged objects for software manipulation as well (using bit code patterns). Interaction results are rear-projected from the bottom of the rack, so that no occlusion of the projected image by the user(s) occurs. Microsoft Surface uses a special hardware technology called

Figure 2.5: Product image of the WeTab by the German manufacturer Neofonie (with optional stand and external hard drive). Courtesy of Wetab.mobi [Wet10].

Figure 2.6: Product image of the Apple Magic Trackpad for multi-touch tracking on Macintosh desktop computers. Courtesy of Apple.com [App10b].

Rear Diffused Illumination, which is explained in section 3.2 and furthermore in 3.2.2.

Microsoft distributes a collection of applications known as the Microsoft Touch Pack, which includes a map-exploration and a collage tool for manipulating digital maps and images, respectively. The system is said to provide a fundamental change in the way people interact with digital content as well as tremendous potential for improving communication in business situations [Mic10].

2.4.1.2 GestureTek Illuminate Multi-Touch Table

Basically, the GestureTek Illuminate Multi-Touch Table (cp. figure 2.8) offers similar interaction to Microsoft Surface, whereby it does not support other objects for manipulation than human hands. Again, the configuration is based on a display (30, 40 or 50 inch) and consists of a video camera for image recognition and a projector device for rear projection. GestureTek uses its patented hand tracking technology, where a thin curtain of infrared light is projected along the display surface. When a fingertip or hand

Figure 2.7: Product image of the Microsoft Surface rear projection multi-touch table. Courtesy of Microsoft.com [Mic10].

is illuminated by the infrared light source, the camera instantly determines its position on the surface. This technique is similar to the Light-Plane illumination technology, which is described in 3.3. According to GestureTek, the Illuminate Multi-Touch Table supports nearby pointing in front of the screen as well, where actually no physical touch is required (i.e. hovering of fingertips). [Ges10]

2.4.1.3 DiamondTouch

DiamondTouch is a multi-user touch technology, which has been initially presented by the Mitsubishi Electric Research Laboratories (MERL) and was later commercialized by Circle Twelve Inc. [Mer10, Cir10]. In comparison to Microsoft Surface and the Illuminate Multi-Touch Table by GestureTek, DiamondTouch does neither use video cameras nor rely on the sensing of light. The recognition of user input is obtained by capacitive surface technology instead, which is explained in section 3.5. Figure 2.9 highlights the basic hardware setup and working principle. As it can be understood from the figure,

Figure 2.8: Illustration of the GestureTek Illuminate Multi-Touch Table. Courtesy of Gesturetek.com [Ges10].

the DiamondTouch technology requires the user(s) to wear cables and receivers, which results in bad usability. Nevertheless, the system is able to distinguish between input from different users by retracing the signals back to their (unique) receivers. DiamondTouch uses overhead projection. Visual occlusions of the projected content by the users are unavoidable.

2.4.2 Open Systems & Scientific Approaches

Important work on multi-touch hardware technology is known by Jefferson Y. Han and the NUI Group. Furthermore, two relevant vision-based approaches for multi-touch sensing in tabletop environments are described. This includes the Microsoft PlayAnywhere system and another vision-based approach by Microsoft Research for high precision multi-touch sensing.

2.4.2.1 Multi-Touch Interaction Experiments by Jeff Han

Being consulting research scientist at the New York University Department of Computer Science [Han06], Jefferson Y. Han (shortly called Jeff Han) presented his approach for

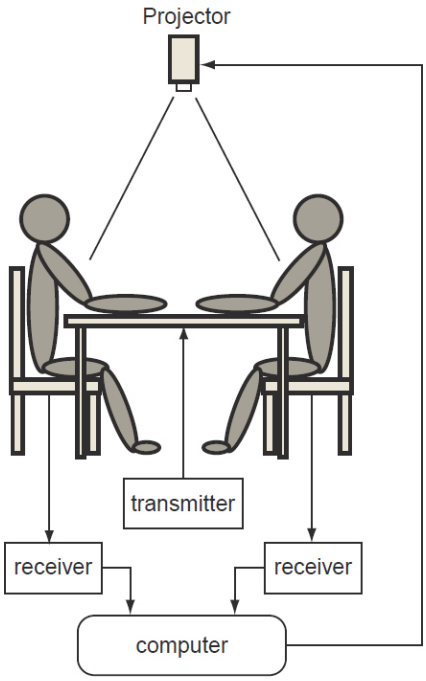


Figure 2.9: A schematic diagram of the MERL DiamondTouch hardware setup and basic working principle. The tabletop system uses capacitive surface technology based on electronics for touch detection. Overhead projection provides visual feedback to the user(s). Courtesy of P. Dietz and D. Leigh [DL01].

multi-touch sensing hardware at the 18th annual ACM Symposium on User Interface Software and Technology in 2005. According to Han, the Frustrated Total Internal Reflection-based technology should lead to scalable and, above all, affordable multi-touch interfaces [Han05]. By publishing videos of his work and presenting his multi-touch table at the TED conference in 2006 [Ted06], Jeff Han soon became a major expert for multi-touch research across all over the world. The Frustrated Total Internal Reflection technology for multi-touch detection may be found in a number of both proprietary and open systems and is explained in detail in section 3.1.

2.4.2.2 NUI Group

The NUI Group describes itself as a global open source research community focused on the development of Natural User Interfaces, hence the name, and has been established in 2006. Throughout its existence, the NUI Group made important contributions to the field of HCI by providing the NUI Group Community Wiki [Nui10c] and the first

open book for multi-touch technologies: NUI Group Community Book - Multi-Touch Technologies, which can be retrieved at [Nui10b] and is available for free download in the portable document format as well. The group currently consists of more than 10.000 members, which are widely spread across all over the world, and is open to new members for free registering [Nui10a].

Popular multi-touch hardware technologies, such as the Rear Diffused Illumination approach, which is used in Microsoft Surface, or Laser-Light illumination, which is used in the Microsoft LaserTouch prototype [Nui10a, Gre08], and LED-Light Plane illumination have been or are being developed and enhanced within the community. As the efforts of the NUI Group are of major importance for state-of-the-art multi-touch interaction systems, the Diffused Illumination technology with its different forms and the different Light Plane illumination techniques are explained in sections 3.2 and 3.3. Furthermore, another recent technology called Diffused Surface Illumination, which is developed within the community of the NUI Group as well, is explained in section 3.4.

2.4.2.3 Microsoft PlayAnywhere

PlayAnywhere is a front-projected interactive tabletop system, which has been presented by Andrew D. Wilson from Microsoft Research in Redmond in 2005. Generally, PlayAnywhere consists of a rather compact hardware setup, where an infrared light source and a camera are mounted on a portable projector device, as it is illustrated in figure 2.10. The unit is positioned on a table, whereby the interactive surface is projected on the tabletop in front of the projector to allow bimanual interaction to the user. The surface is illuminated by the infrared light source, which is mounted off-axis from the camera to generate shadows of incoming objects (e.g. hands). Touch detection is performed by analyzing these shadows. [Wil05] The working principle of the approach is explained in detail in 3.7.1).

2.4.2.4 Microsoft Vision-Based Approach

Another relevant multi-touch sensing approach has been presented by the Cambridge Microsoft Research Group in 2007. Ankur Agarwal et al. use a stereo camera pair, which is mounted overhead on a stand, viewing the interactive surface (i.e. a tablet display) from above (cp. figure 2.11). The system uses a novel approach, where machine learning strategies and a geometric finger model are combined in order to train the system to work under different physical conditions (e.g. variable lighting, various hand sizes, etc.). [AICB07] The function of the approach is described in 3.7.2).

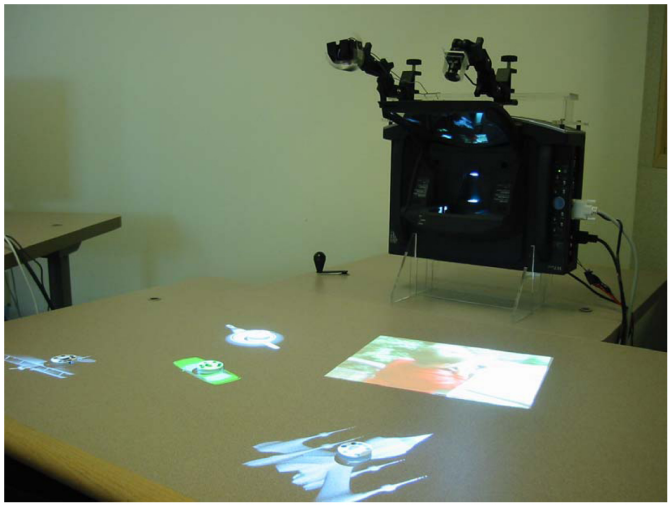


Figure 2.10: Microsoft PlayAnywhere refers to a compact hardware setup, consisting of a portable projector device with an infrared light source and camera attached to it. Visual feedback is projected in front of the system and touches on the interactive tabletop surface are detected by analyzing the shadows of incoming objects (e.g. hands). Courtesy of A. Wilson [Wil05].

Figure 2.11: Microsoft vision-based approach, which consists of a stereo camera pair mounted on a stand that views the scene. The interactive surface (i.e. a tablet display) is fixated at the bottom of the stand. Courtesy of A. Agarwal et al. [AICB07].

CHAPTER 3

Related Work & Theoretical Foundations

In this chapter, popular technologies for multi-touch sensing on tabletops and other large, flat surfaces are explained. This includes a number of optical-based (i.e. light sensing) solutions and a capacitive surface technology for providing touch-sensitive computer hardware. In addition, relevant computer vision and machine learning approaches for the detection of fingertips are described at the end of this chapter.

Optical-based solutions basically consist of a special surface structure that is representing the interactive surface, various light sources, optical sensors (e.g. video cameras) and either a projector or LCD display device for providing visual feedback. Typically, infrared (IR) light is used to illuminate the structure. When objects (e.g. fingertips) are touching or hovering above the interactive surface, light beams are deflected and scattered or diffused light is sensed by the video camera. Usually, the video stream is processed by image recognition software. Hovering and touching produces either bright or dark blobs in the video frames, depending on the specific technique. Blob detection algorithms are used to obtain the two-dimensional positions of the spots, which correspond to their absolute positions on the interactive surface.

Generally, the surface structure is mounted on a frame, usually having the proportions of a regular table. This allows multiple users to gather around the setup and interact with each other as well as with the system. Different setups are categorized according to the configuration of the surface structure that consists of various layers and different materials and furthermore according to the kind and position of the light sources. Three basic hardware setups are observed:

- Frustrated Total Internal Reflection (FTIR)
- Diffused Illumination (DI)

- Light Plane (LP) illumination

FTIR (section 3.1) is very popular and widely used for multi-touch sensing in tabletop systems. Here, the interactive surface consists of a main panel, a compliant layer and an optional diffuser. The main panel is illuminated internally by a frame of IR light-emitting diodes (LEDs). In DI (section 3.2), the main panel is equipped with a diffuser and is illuminated by multiple light sources from either above or below. Basically, two forms of DI are distinguished: Front and Rear DI (3.2.1 and 3.2.2), where Rear DI is the more common approach. LP illumination technology (section 3.3) again requires a diffuser layer and IR light sources are used to generate a so-called plane of light just above the surface structure. According to the light source, two different types are distinguished: Laser-Light Plane (LLP) and LED-Light Plane (LED-LP) illumination (3.3.1 and 3.3.2). [Nui10b]

Another technique, similar to both FTIR and DI, is called Diffused Surface Illumination (DSI), which is explained in section 3.4. Here, the configuration of the surface structure and the illumination technique are analog to FTIR but the basic working principle is similar to DI. [Nui10b]

Another popular approach for multi-touch sensing on flat surfaces is capacitive surface technology (section 3.5). The underlying principle of capacitive coupling has already been used for a number of years for touch detection in computer trackpads and mobile devices (e.g. smartphones, MP3 players, etc.). Here, a grid of capacitors is installed right underneath the touch surface and the coupling of electrical signals is used to obtain the surface touch positions.

Section 3.6 provides a summary of the common hardware technologies by pointing out important pros and cons. At the end of this chapter, alternative vision-based techniques for the detection of fingertips are presented in section 3.7. This includes state-of-the-art computer vision and machine learning algorithms to process digital images obtained by standard video cameras. This work is widely based on the paper by Letessier and Bérard (subsection 3.7.3), which provides a straightforward approach.

3.1 Frustrated Total Internal Reflection (FTIR)

A multi-touch setup based on Frustrated Total Internal Reflection has been introduced by Jeff Han in 2005 at the 18th annual ACM Symposium on User Interface Software and Technology (UIST '05). Since then, the term multi-touch is commonly used by the community of HCI researchers. [Han05]

The phrase actually refers to the underlying optical principle. In optics, a certain phenomenon occurs when light beams travel from one material into another, while the two materials do not share the same refractive index. In this case, light beams are deflected in a specific angle, which can be calculated mathematically and depends on

the materials' refractive indices. The refractive index refers to the ratio the speed of light travels in vacuum, relative to its velocity in a corresponding medium (i.e. material) [Wik10f]. When light travels from a material with a specific refractive index into another with a lower one, the light beams are totally reflected if the angle of deflection is greater than the previously calculated one [Nui10b, Han05]. The effect may be observed when sunlight enters water, for example. FTIR setups rely on this optical phenomenon.

Figure 3.1 shows a schematic diagram of a typical FTIR setup. Basically, a plexiglass (i.e. acrylic) panel is installed in front of a camera. Due to the use of IR light, the camera needs to be equipped with an IR-pass filter allowing only light in the infrared spectrum to the lense. IR LEDs are installed at the sides of the panel, so that light beams can enter the acrylic and illuminate it internally. As visualized in figure 3.1, the light beams are completely trapped inside the acrylic, due to the principle of total internal reflection. If the user touches the surface, light beams are no longer totally reflected at the points of contact and can pass through into the contact material (e.g. skin). The light beams are then said to be frustrated, hence the name [Nui10b]. Being reflected from the contact material, light scatters downwards towards the lense of the camera.

The acrylic panel needs to provide a certain thickness not to get bent, like it has been initially mentioned in 2.3.1. For large interactive surfaces a thickness of 10 mm is recommended, whereas the minimal thickness is 6 mm [Han05]. This constraint may be generalized to the other optical-based techniques described in this work as well. The sides of the acrylic need to be polished with very fine sandpaper or even wet sandpaper in order to be very clear, so that light beams can enter the material smoothly and without any disturbances. FTIR setups use a diffuser to remove visual noise (e.g. darker objects in the back of the scene). Only bright objects (i.e. touches) are allowed by the diffuser. Generally, touch detection in FTIR works better when the fingertips are wet or greasy, because in that case the contact to the surface is better and light beams get frustrated easier [Nui10b]. For that reason, a compliant layer (e.g. made of silicone) is installed on top of the acrylic. This makes the interactive surface physically sensitive to variable touch pressures as well. Compliant layers are only used in FTIR setups. Figure 3.2 shows some examples of Jeff Han's multi-touch experiments.

An advantage of FTIR is that the final interaction system does not necessarily have to be enclosed. No self-contained box is needed. Another advantage of FTIR is that the detected blobs provide strong contrast and even different blobs produced by variable touch pressures are recognized by the system. Furthermore, FTIR is able to recognize objects as small as styluses and pen tips. On the other hand, FTIR is not able to recognize visual markers (e.g. fiducials). The installation of the LED frame is difficult, because it requires complex soldering work. Moreover, FTIR calls for a compliant surface. Obviously, neither glass nor acrylic can be used here. Hovering above the surface is not detected. [Nui10b]

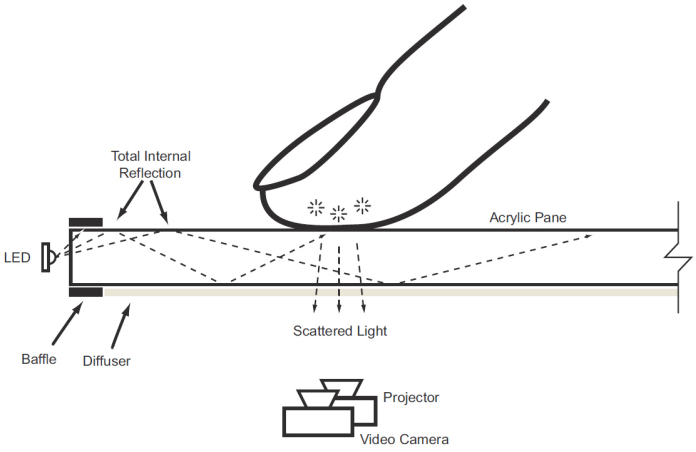


Figure 3.1: In FTIR setups, the acrylic panel is illuminated internally by multiple LEDs positioned at the sides. Light beams are trapped inside the material, due to the principle of total internal reflection, and scatter downwards towards the camera lense, if a contact material (e.g. skin) touches the top of the surface. Courtesy of J. Han [Han05].

3.2 Diffused Illumination (DI)

Diffused Illumination requires a hardware setup similar to FTIR but the basic working principle is different. The main panel of the interactive surface is usually made out of acrylic or even glass. Basically, any transparent material can be used. Unlike FTIR, DI does not require a compliant surface, which makes the surface physically insensitive to variable touch pressures. The scene is illuminated by one or more (mostly) IR light sources installed either above or below the surface. Basically, the contrast between a known image and the images produced when objects are touching or hovering above the surface is regarded for touch detection. [Nui10b]

In the following, two basic forms of DI are explained: Front and Rear Diffused Illumination. Both techniques are based on the same working principle but use slightly different hardware configurations.

3.2.1 Front Diffused Illumination (Front DI)

The hardware configuration for the Front Diffused Illumination technology is comparatively simple. In comparison to the other optical-based solutions explained in this work, Front DI does not exclusively rely on built-in IR light sources. The surface structure is illuminated by both ambient light from the surroundings and multiple IR light sources positioned above the interactive surface. An IR video camera is mounted below the

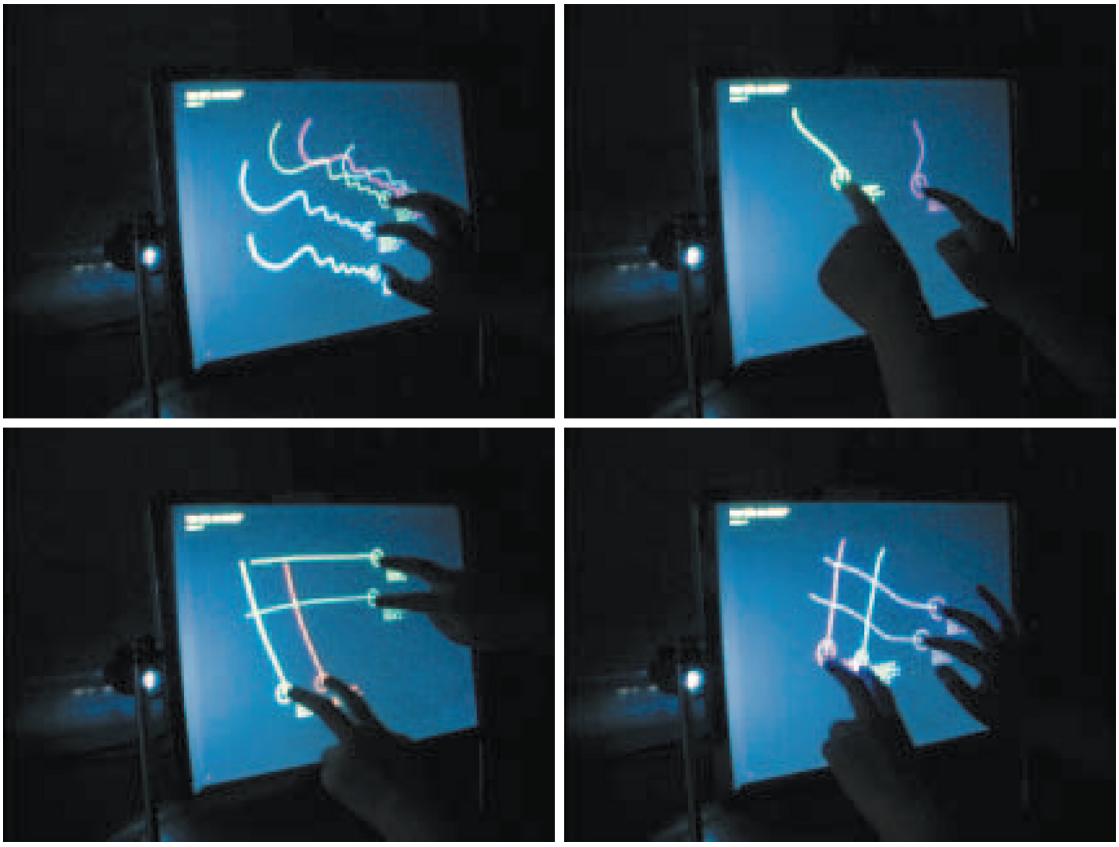


Figure 3.2: The FTIR multi-touch technology is able to recognize simultaneous touches and rear-projects visual feedback on the semi-transparent contact surface. Courtesy of J. Han [Han05].

surface. Either above or below the main panel a diffuser layer is installed to prevent background objects from disturbing the scene. When fingertips are moving near the interactive surface, shadows are sensed by the camera and their corresponding positions are obtained by blob detection algorithms. Figure 3.4(a) shows a typical detection image produced by the Front DI technology. Here, the image is bright, whereas touches are presented as dark blobs. Other optical-based techniques obtain inverted detection images.

A major advantage of the Front DI technology is its simple hardware setup. Similar to FTIR, no enclosed box is required to surround the bottom of the setup. Furthermore, the interactive surface does not need to be equipped with a special compliant layer and standard IR light sources and cameras are sufficient. Nevertheless, Front DI has a lot of drawbacks. Due to the use of ambient light, lighting conditions are hardly controllable, which affects the system's robustness. Small illumination changes may

have great impact on the detection results. Furthermore, constant illumination on the whole surface is difficult and touches may not be detected equally on the whole area of the interactive surface. Due to the lack of a compliant layer, Front DI is not pressure-sensitive. Like FTIR, Front DI is not able to detect visual markers. It mainly depends on the diffuser layer if hovering of fingertips is detected. [Nui10b]

3.2.2 Rear Diffused Illumination (Rear DI)

Figure 3.3 shows a schematic diagram of the basic Rear DI hardware setup. Unlike in Front DI, the interactive surface is lighted by multiple IR illuminants installed below the surface in an enclosed box that surrounds the bottom of the setup. Either on top of or underneath the transparent main panel a diffuser layer is installed. The choice of the diffuser material is critical. On the one hand, it needs to allow as much light, so that near fingertips are illuminated brightly, while the background stays dark. On the other hand, the diffuser layer needs to inhibit strong reflections from nearby objects in the back. Generally, strong contrast between nearby objects and the distracting environment is important. This makes later blob detection much more successful. In contrast to Front DI, Rear DI obtains dark detection images, whereas touches are presented as bright blobs (cp. figure 3.4(b)). A popular example for the Rear DI technology is the Microsoft Surface multi-touch table, which has been initially described in 2.4.1.1.

Rear DI is very popular for multi-touch sensing interfaces, because the hardware setup is relatively simple, in comparison to FTIR, for example. No complex IR LED frame is required and standard IR illuminants are sufficient. Depending on the diffuser hovering of fingertips is detected. Furthermore, Rear DI systems are able to detect visual markers such as fiducials and other visual bit-code patterns. Like in Front DI, constant illumination over the whole surface is difficult, because the IR light sources do not provide lighting with even intensity over the whole light cone. This may result in bad detection results, especially at the corners of the surface. Rear DI systems are not pressure-sensitive. [Nui10b]

3.3 Light Plane (LP) Illumination

Like DI, Light Plane setups use transparent acrylic or glass for the main panel of the interactive surface and the video camera is mounted below the surface structure. The LP setup does not require an enclosed box. Basically, multiple IR light sources are installed on top of the panel to generate a plane of light just above its surface. Whenever objects break through this plane, light beams are deflected and sensed by the camera as bright spots in the corresponding detection images. LP uses a diffuser layer to suppress visual noise in the back of the scene. According to the kind of the light source, two forms of LP are distinguished: Laser-Light Plane and LED-Light Plane illumination.

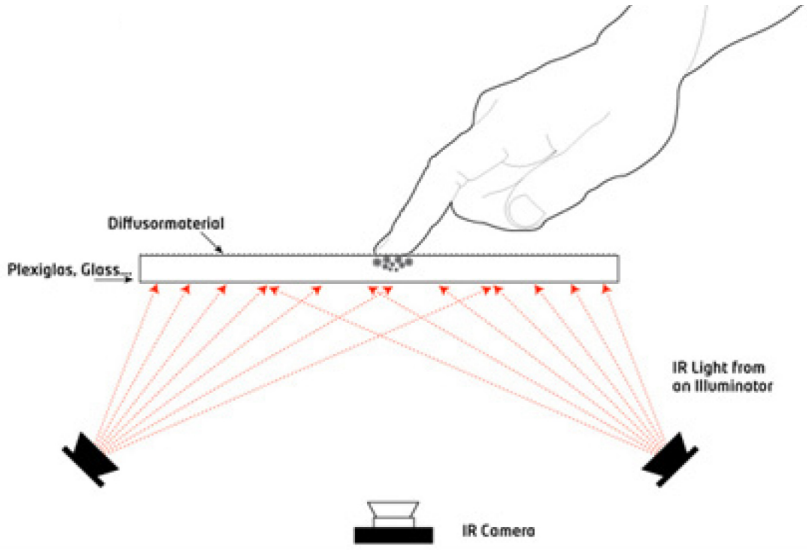
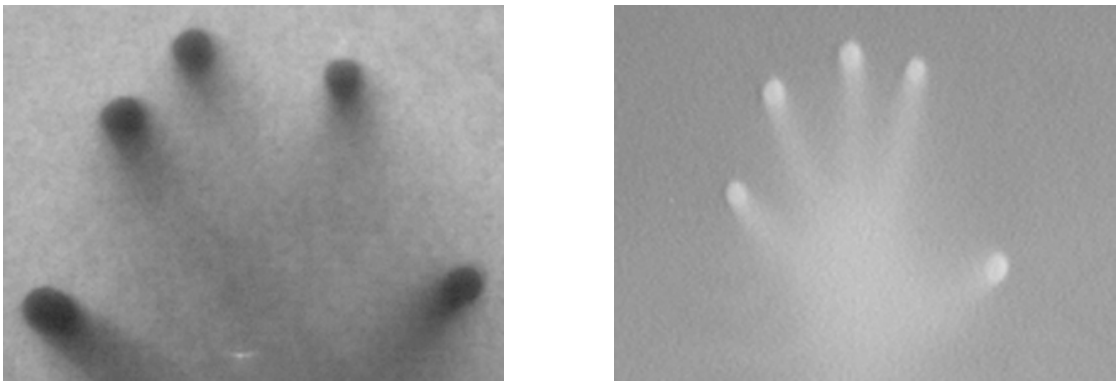


Figure 3.3: In Rear DI, the panel is illuminated from below, while light beams are deflected by fingers that are touching the top of the surface. Diffused light is sensed by the camera, which is mounted below the surface structure. Courtesy of the NUI Group [Nui10b].



(a) (b)

Figure 3.4: Example detection images for Front DI (a) and Rear DI (b). In Front DI, the background of the scene is bright, whereas touches are presented as dark blobs. Rear DI obtains inverted detection images, as it is shown in (a). Other optical-based technologies show similar results. Courtesy of the NUI Group [Nui10c].

Both techniques have been pioneered within the community of the NUI Group by Alex Popovich and Nima Motamedi, respectively. [Nui10c, Nui10b]

3.3.1 Laser-Light Plane (LLP) Illumination

In the case of Laser-Light Plane illumination, one or more laser devices are used for illumination. Those are usually installed at the corners of the interactive surface to spread an equal plane of light just above the surface structure. Specific line lenses are used to provide a certain angle of horizontal illumination. Both the kind of the line lenses and the position of the laser devices need to be chosen wisely, so that the interactive surface is illuminated equally and the system is affordable. Commonly, about 120° line lenses and 2–4 laser devices are used in LLP setups. The laser plane is typically about 1 mm thick. [Nui10b] On the lower side of the panel, a diffuser is installed to prevent from visual noise (e.g. other bright objects in the back). When fingertips are moving very close to the interactive surface, light beams are deflected and strive downwards to the lense of the IR camera. As in FTIR and Rear DI, the detection images are mainly dark, whereby scattered light is presented as bright blobs. Figure 3.5 shows the basic setup and working principle of the LLP technology.

Due to the use of lasers, safety is a great issue in LLP setups. IR lasers cannot be perceived by the human eye and serious damage can occur to the retina when laser light is used inappropriately. It is common to use laser devices of 5–25 milliwatt (abbr.: mW) of power [Nui10b].

A major advantage of LLP is its comparatively simple hardware setup. No enclosed box has to be prepared and IR laser devices and line lenses are widely available and can be used out-of-the-box. Furthermore, no complex IR LED frame is required. Like in DI setups, the main panel is made out of relatively cheap glass or acrylic and does not have to be specially equipped with a compliant layer. Only a diffuser layer is used. On the other hand, LLP is not able to detect visual markers and the interactive surface is not pressure-sensitive. Light intensity does not change with different touch pressures. Moreover, LLP comes with a specific drawback. If too few laser devices are used (e.g. just one or two lasers), occlusions can occur. For example, if laser beams are deflected by one object, another object in the back of the first object is not illuminated anymore. In that case, the second object would not be detected. [Nui10b]

3.3.2 LED-Light Plane (LED-LP) Illumination

Generally, the LED-Light Plane illumination technology relies on the same working principle as LLP illumination, namely the establishment of a light plane on top of the interactive surface. Nevertheless, the basic hardware setup is similar to FTIR. A frame of IR LEDs is used to illuminate the scene. In contrast to FTIR, the main panel is not illuminated internally. In the case of LED-LP, the LED frame is installed right on top of

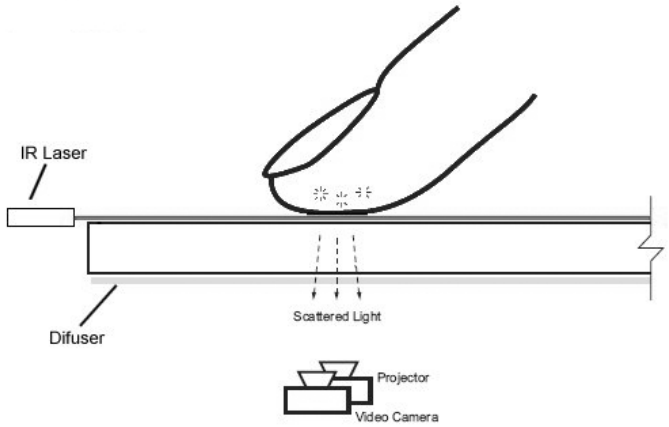


Figure 3.5: The LLP illumination technology uses infrared laser devices to generate a plane of light just above the top of the surface. When fingertips are touching the surface, light beams are deflected and scatter downwards towards the lense of the camera. Courtesy of the NUI Group [Nui10b].

the surface structure and the LEDs are pointed to the inside of the surface. Thus, a plane of IR light is generated over the surface, just like it is in the LLP approach. Since the light of LEDs is conical and not flat (like laser light), a bezel is used on top of the LED frame to narrow the spread of the light beams in the vertical direction. This effectively prevents further objects from being illuminated as well.

As LLP, LED-LP illumination does not require an enclosed setup and transparent acrylic or glass can be used for the main panel. Again, no compliant layer is required. On the other hand, LED-LP illumination requires a complex LED frame with a bezel on top. Furthermore, both visual markers and hovering of fingertips are not detected. LED-LP setups do not provide touch pressure sensitivity.

3.4 Diffused Surface Illumination (DSI)

Diffused Surface Illumination has been developed within the NUI Group community and is primarily inspired by the work of Tim Roth [Nui10b]. DSI is based on the FTIR setup, while the acrylic is replaced with another, slightly different type. The material used here is called EndLighten and refers to a special type of plexiglass. EndLighten consists of very small particles acting like thousands of tiny mirrors inside the structure. If IR light is sent into the material from the panel's sides, like it is in FTIR by using a frame of IR LEDs, the light beams are reflected by the particles and forced to diffuse out of the panel in every possible direction. This effectively produces constant illumination

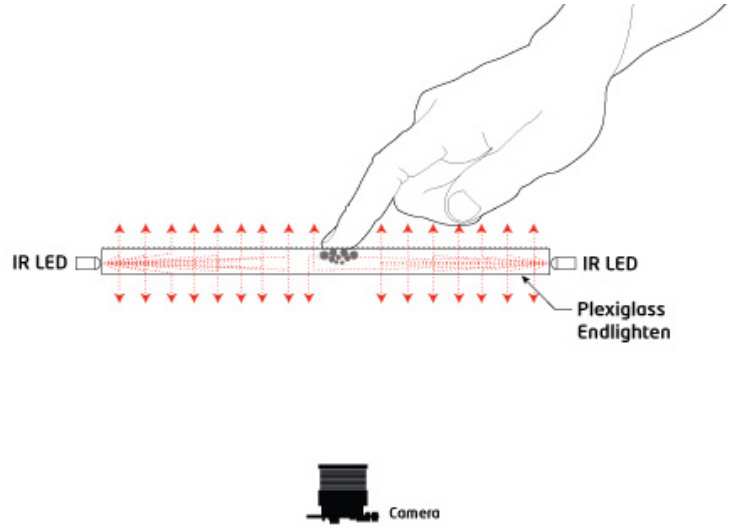


Figure 3.6: In the DSI hardware setup, the acrylic is illuminated internally, just like in FTIR, but small particles inside the material force light beams to diffuse out of the panel, which is similar to DI. Scattered light at the bottom of the surface is sensed by the camera below the surface structure. Courtesy of the NUI Group [Nui10b].

of the whole panel surface. Altogether, this effect is similar to DI. Figure 3.6 shows a schematic diagram of the basic DSI hardware setup.

DSI setups do not require a compliant surface but are pressure-sensitive. Because of the constant illumination, different touch pressures are recognized via varying intensity of the diffused light beams. By replacing the acrylic panel, the system can be switched easily from DSI to FTIR and back. Furthermore, DSI is able to detect visual markers and hovering of fingertips. Unfortunately, EndLighten is much more expensive than regular plexiglass and a complex LED frame is required as well. Despite constant illumination on the touch surface, DSI provides lower contrast blobs than, for example, FTIR and LLP. DSI setups require an enclosed box. [Nui10b]

3.5 Capacitive Surface Technology

Capacitive surface technology is based on the effect of capacitive coupling, which is commonly known in electronics [Wik10b]. In comparison to the previously described techniques, capacitive surface technology (or capacitive sensing) does not rely on the sensing of light beams in any kind of way. Instead, signals in the form of electrical impulses are regarded.

Basically, a grid of capacitors is installed right underneath the touch surface. If a conductive object (e.g. a fingertip) comes near the surface, an electrical impulse is transmitted from the object to the near capacitor(s). The system is then able to recognize the two-dimensional position of the impulse's origin on the surface. Commonly, two types of capacitive sensing are distinguished: mutual and self (or absolute) capacitance. In mutual capacitance, driving lines (containing current) and sensing lines (detecting current) are placed crosswise on two distinctive layers (cp. figure 3.7(a)). Self (or absolute) capacitance uses just one layer consisting of individual electrodes connected with capacitance-sensing circuitry (cp. figure 3.7(b)). [Wil10] Both methods are commonly used in small mobile devices like smartphones (e.g. Apple iPhone) and mobile phones.

A popular example for capacitive sensing in tabletop environments is the DiamondTouch table by MERL, which has initially been mentioned in 2.4.1.3. Generally, DiamondTouch (cp. figure 2.9) is a front projection system, whereby the projector device is installed above the interactive surface. The multi-touch functionality is provided by using a specific type of capacitive sensing. Here, an array of antennas is embedded right underneath the top layer of the surface that transmit very small radio-frequency signals. Each one of the users is connected to a specific receiver, which is typically mounted on the users' chairs. If a user touches the surface, a small amount of the signal from a number of antennas is coupled to the user's receiver. Since every participant is connected to a unique receiver, the system is able to assign multiple manipulations to the different users. In comparison to optical-based touch-sensitive hardware setups and other vision-based approaches, this is a major advantage, since the assignment of multiple input to various users is not trivial in that case. Figure 3.8 shows a schematic diagram of the surface structure.

3.6 Hardware-Based Techniques – Summary

As a matter of fact, neither an ideal touch-sensitive hardware setup nor the best multi-touch surface technology can be presented. Each of the previously described techniques has specific pros and cons, which are summarized in the tables 3.1 and 3.2. Searching for an optimal technique is not trivial. One needs to understand the field of application and the expected performance (i.e. autonomy, usability, robustness, etc.) of the system.

In tabletop-like environments, it is commonly desirable to have a simple and compact hardware setup, which can be set up and moved easily and performs equally well in different locations and under varying environmental conditions. Usability, scalability, portability and robustness are important specifications here. Moreover, cost and computational performance can be issues as well. In the following, relevant vision-based techniques for multi-touch sensing on large interactive surfaces, addressing these requirements, are described.

Frustrated Total Internal Reflection (FTIR)	
Pros	Cons
• Pressure-sensitive due to different blobs resulting from variable touch pressures • Detection images provide strong contrast blobs • Requires a specially prepared acrylic panel • No enclosed box is required	• A compliant layer (e.g. silicone) is required • A complex frame of LEDs is required • Visual markers (e.g. fiducials) are not detected • Hovering is not detected

(a)

Front Diffused Illumination (Front DI)	
Pros	Cons
• Simple hardware setup • No compliant layer is required • No complex frame of LEDs is required • No enclosed box is required • Hovering is detected	• Constant illumination is difficult • Robustness depends on ambient lighting conditions • Visual markers (e.g. fiducials) are not detected • Not pressure-sensitive

(b)

Table 3.1: Pros and cons of the FTIR (a), Front DI (b), Rear DI (c), LLP illumination (d), LED-LP illumination (e) and DSI (f) multi-touch technologies [Nui10b].

Rear Diffused Illumination (Rear DI)	
Pros	Cons
• Visual markers (e.g. fiducials) are detected • No compliant layer is required • No complex frame of LEDs is required	• An enclosed box is required • Constant illumination is difficult • Detection images provide low contrast blobs • Not pressure-sensitive

(c)

Laser-Light Plane (LLP) illumination)	
Pros	Cons
• Simple hardware setup • No compliant layer is required • No complex frame of LEDs is required • No enclosed box is required	• Visual markers (e.g. fiducials) are not detected • Hovering is not detected • Not pressure-sensitive • Occlusions might occur

(d)

Table 3.1: (continued)

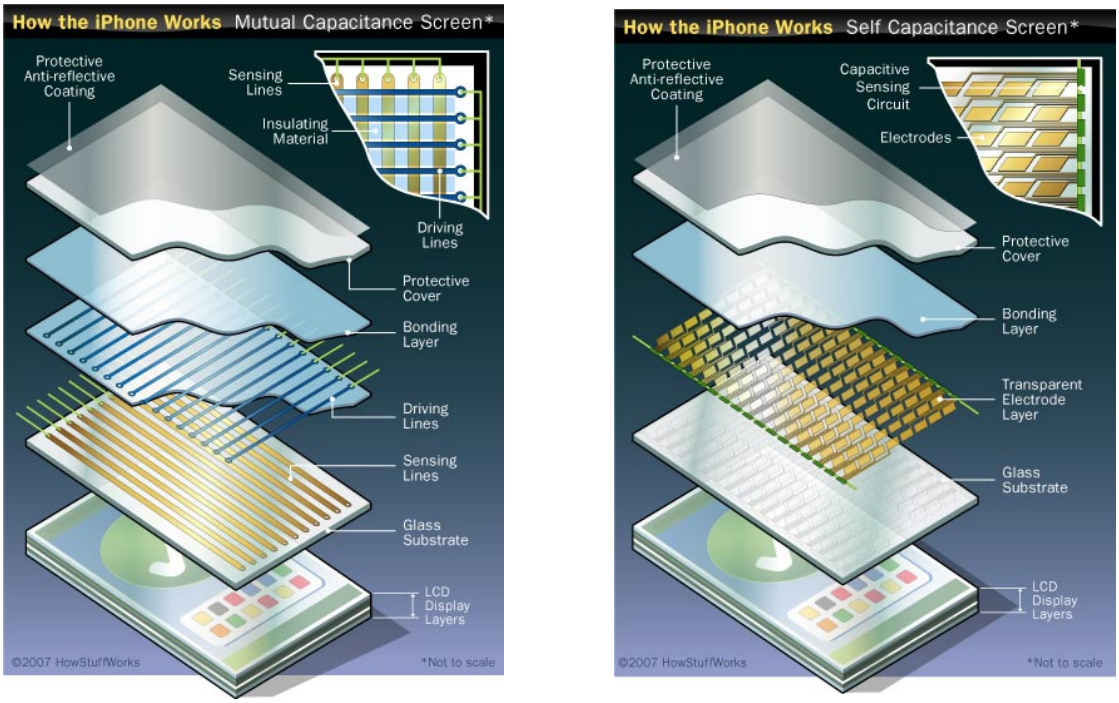
LED-Light Plane (LED-LP) illumination)	
Pros	Cons
• No compliant layer is required • No enclosed box is required	• A complex frame of LEDs is required • Visual markers (e.g. fiducials) are not detected • Hovering is not detected • Not pressure-sensitive

(e)

Diffused Surface Illumination (DSI))	
Pros	Cons
• Possibility to switch from DSI to FTIR • Pressure-sensitive • Visual markers (e.g. fiducials) are detected • Hovering is detected • Constant illumination • No compliant layer is required	• Expensive EndLighten plexiglass is required • Detection images provide low contrast blobs

(f)

Table 3.1: (continued)



(a) (b)

Figure 3.7: Mutual (a) and self (or absolute) capacitance (b) used in popular mobile devices such as the Apple iPhone. Courtesy of Howstuffworks.com [Wil10]

3.7 Vision-Based Techniques

Over the past years, systems that are primarily based on computer vision have become more and more popular among multi-touch researchers. The effort is to abandon complex touch-sensitive hardware panels that mostly provide expensive, non-scalable and non-portable interfaces and go for the use of everyday surfaces (e.g. sheets of paper or cardboard, tabletops, walls, etc.) instead.

Obviously, such natural artifacts cannot provide interactivity in the first place. A simple panel made out of cardboard or a tabletop is by no means able to detect finger touches and deliver the corresponding spatial position to a computer system. That is where computer vision approaches come in. Interactive functionality is simply added to the regular surface by the combination of computer vision hardware and software. The scene is watched by video cameras that obtain digital images of both the surface and incoming objects (e.g. fingertips, visual markers, etc.). Those images are then preprocessed by different low-level computer vision algorithms in order to detect features (e.g. different shapes, connected components, etc.) for further image recognition. In

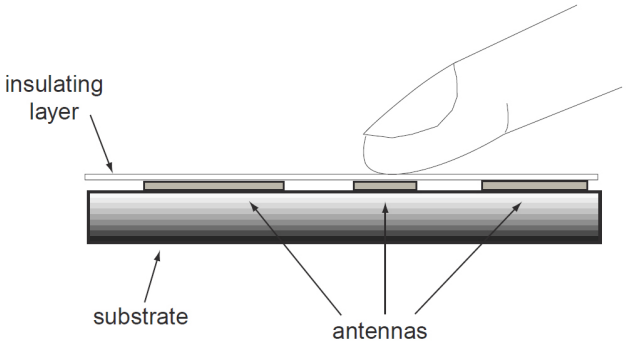


Figure 3.8: The surface structure of the DiamondTouch system consists of a number of antennas installed right underneath an insulating layer that transmit very small radio-frequency signals. If a conductive object (e.g. a fingertip) touches the surface, signals from near antennas are coupled between the surface and a receiver, which is situated at the object. Courtesy of P. Dietz and D. Leigh [DL01].

Capacitive Surface Technology	
Pros	Cons
• Fast detection • Not affected by lighting conditions • Pressure-sensitivity depends on software • Easy assignment of multiple input to various users (if unique receivers are used)	• Complex hardware setup • Non-conductive objects (e.g. styluses, pens, etc.) are not detected • Visual markers (e.g. fiducials) are not detected • Not convenient for large interactive surfaces

Table 3.2: The table summarizes pros and cons of the capacitive surface technology for multi-touch detection on large interactive surfaces.

some cases, machine learning strategies are used to train the system to work under variable conditions, such as varying lighting and finger widths, for example. As video cameras are commonly available and environmental lighting is used, the required hardware setup is inexpensive and comparatively simple, in contrast to custom-made touch-sensitive hardware. On the other hand, illumination is crucial for the performance of the system and needs to be considered predominantly.

A major task in vision-based approaches is the segmentation of the scene. Desired foreground (e.g. hands, markers, etc.) needs to be segmented from the distracting background (e.g. camera noise, other objects residing on the interactive surface, projected visual feedback, etc.). Image segmentation is rather difficult here, because illumination conditions change over time and can make parameterized approaches inefficient. Furthermore in natural settings, the image background typically contains a lot of visual clutter.

Due to the variety of computer vision and machine learning algorithms, a number of different approaches on vision-based multi-touch sensing can be found in the literature. In the following, some relevant and representative techniques are described. This includes the underlying working principle of the PlayAnywhere interaction system using infrared lighting [AICB07], a stereo-vision approach for multi-touch sensing [AICB07] and last but not least a straightforward computer vision approach for the detection of bare fingers [LB04].

3.7.1 Wilson / PlayAnywhere

As explained in 2.4.2.3, PlayAnywhere refers to the front-projected interactive tabletop system developed by Andrew D. Wilson from Microsoft Research (cp. figure 2.10). Wilson's prototype of a compact interaction system basically addresses installation, calibration and portability issues, which are all relevant in tabletop systems. Its underlying computer vision technique provides a number of contributions to the field of image recognition, including a shadow-based touch detection algorithm and the continuous tracking of sheets of paper [Wil05]. Relevant parts of the basic working principle (i.e. image rectification, fingertip detection, page tracking) are described in the following:

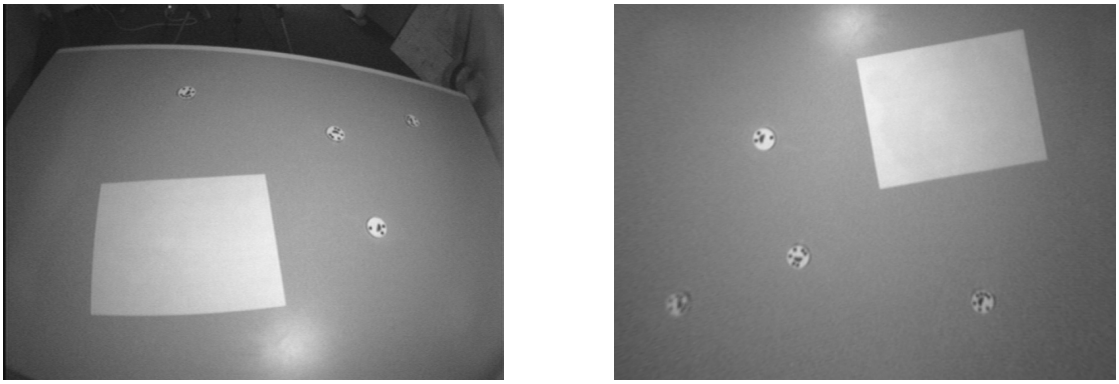
1. *Image rectification.* As an initial calibration step, barrel and projective distortions, imparting from the specific camera lense and slightly oblique position of the camera, respectively, are removed from the image by standard bilinear interpolation. Figure 3.9 shows the initial input and corrected (rectified) output images. This step does not have to be performed again if the projector unit is moved to a different location, since the position of the camera on the unit and the distance and angle between the lense and the surface are constant. Generally, image rectification is used to bring spatial two-dimensional coordinates into correspondence. A certain object on the table will have the same (scaled) dimensions in the rec-

tified image gathered by the camera. A minor drawback here is that, due to the oblique camera view, objects in the back will have lower resolution than objects in the front. Image rectification is used in this work as well and is furthermore explained in 4.3.2.

2. *Fingertip detection, touching and hovering.* With PlayAnywhere, Wilson proposes a modern computer vision approach based on the analysis of shadow shapes. Due to the hardware configuration, where an IR illuminant is mounted on the projector unit (off-axis from the camera), objects (e.g. hands) that are moving along the tabletop will produce shadows on its surface. Figure 3.10 shows how the shape of these shadows change when the fingertip actually touches the surface. In this case, the corresponding shape is completely covered by the fingertip. The distance between the actual fingertip and its shadow on the surface is used to determine the actual height of the fingertip above the tabletop. According to [Wil05], this height can be calculated exactly, due to the constant angle between the surface and the camera lense, and may be effectively used for hovering detection. The shadow images are produced by a simple thresholding operation performed on the rectified input image. The fingertips themselves are detected by finding the one point on each of the distinctive shadows that is closest to the projector unit, while assuming that hands enter the scene from the bottom of the projected image and move towards the unit. The positions of these points are obtained by the use of a connected-component analysis. It has to be said that only one point (i.e. one fingertip) per shadow is detected, which prevents the system from recognizing multi-finger gestures with one hand.

3. *Page tracking.* PlayAnywhere offers functionality for continuously tracking sheets of paper of known dimensions. This is obtained by applying an edge detection algorithm on the rectified image and using the Hough transform to build up a histogram over orientation and perpendicular distance to the origin of strong lines in the image, commonly known as the Hough space. The Hough space is searched for appropriate pairs or parallel lines with a certain distance and angle. Figure 3.11 shows a detection example of the page tracking approach. Both edge detection and the principle of the Hough transform are furthermore explained in 4.3.1 to the surface tracking approach that is provided by this work.

A major advantage of the system is its portability. Hence the name, PlayAnywhere is meant to be used in virtually any environment, if lighting conditions are appropriate. For good segmentation, the projected image is completely blocked to the camera lense, due to the use of an infrared-pass filter. On the other hand, tracking results depend on how infrared light beams are reflected by natural objects in the scene. Stray infrared light in the environment can be an issue. Since computer vision algorithms are comparably complex, computational cost is important. Moreover, the detection of just one fingertip



(a) (b)

Figure 3.9: Before (a) and after (b) image rectification via bilinear interpolation. Courtesy of A. Wilson [Wil05].

per hand is a major drawback and the robustness of the detected position cannot match touch-sensitive surface technology. False positive detections are frequent, due to the loose shape filtering approach (e.g. if not the index finger but another hand part is the topmost point). Moreover, PlayAnywhere uses front projection, so that occlusions of the projected image by the hands of the users are unavoidable and may disturb the user(s).

3.7.2 Agarwal et al.

Agarwal et al. choose a different hardware setup to Wilson's PlayAnywhere. Here, a standard tablet display is used instead of a projector system for providing visual feedback. This shall prevent from occlusions but provide enough space for bimanual interaction. In this approach, only light in the visible spectrum is regarded and neither IR illuminants nor infrared-pass filters are required. Instead, a stereo camera pair is mounted on a stand, viewing the scene from above (cp. figure 2.11) and providing two different views of the interactive surface. Agarwal et al. present a novel technique for multi-touch sensing based on machine learning and a geometric finger model. The system is said to provide high precision touch sensing that matches the accuracy of touch-sensitive hardware [AICB07]. In the following, the basic function of the technique is explained, including the prime tasks calibration, image segmentation, fingertip, touch and hover detection:

1. *Calibration.* At the beginning of the main algorithm, the camera views are brought into correspondence. For this purpose, straightforward edge detection is used to detect the corners of the tablet display in both of the stereo pair's images. Afterwards, both the homographic transform and depth plane equation (necessary

(a) (b)

Figure 3.10: Detection examples of the PlayAnywhere's shadow analysis approach. The rectified input image (a) illustrates the change in distance between the actual fingertip and the corresponding shadow on the surface when the finger touches the tabletop. After thresholding, shadows are processed by a connected-component analysis in order to determine the topmost point in the shapes (b). Courtesy of A. Wilson [Wil05].

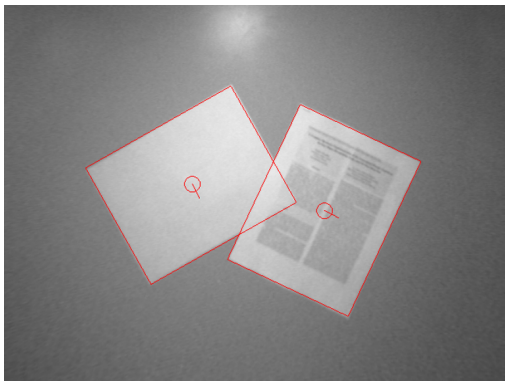


Figure 3.11: A page tracking example of the PlayAnywhere interaction system. White sheets of paper of known dimensions are recognized by the system, even if partly overlapping occurs. Courtesy of A. Wilson [Wil05].

for stereo matching) are calculated and stored for later usage during the further procedures. As the interactive surface is fixated at the bottom of the stand, this calibration step only needs to be performed if the whole system configuration is arranged newly. Image rectification (similar to PlayAnywhere) is not required, because of the straight angle of the cameras, which prevents from perspective distortion.

2. *Image segmentation.* Due to the use of an LCD display device, segmentation of the background is comparably simple. Basically, light emitted by LCD devices is polarized and appropriate polarizing filters for the camera allow all but polarized light. This effectively removes the LCD image from the scene, leaving just foreground objects (i.e. foreground pixels) behind. Furthermore, the view of both cameras is narrowed to the non-moving interactive tablet in order to remove visual clutter in the background of the scene.

3. *Fingertip detection.* In contrast to the shadow shape analysis approach by Wilson, which is based on a relatively simple computer vision heuristic, Agarwal et al. propose a complex machine learning strategy for the detection of fingertips. Here, machine learning shall provide a robust classifier for the identification of points (i.e. tip points) that have high probability of belonging to a fingertip. Probability is calculated by shape and appearance cues, which are used in combination to provide significant probability values. Generally, many input images are used as a training set for the classifier, where tip points and non-tip points have been predefined manually. These images are encoded using local image patches of 8-by-8 pixels, which are initially normalized with respect to rotation and intensity, and are then raster-scanned into 64-dimensional feature vectors (cp. figure 3.12). This effectively produces a specific signature vector for each of the patches, which are handed over to a linear decision rule that is in that case a Support Vector Machine. The machine learning approach is able to classify any new patch to match a tip or non-tip point according to the previously learned data. Multiple tip points form pixel clusters, which are further used for recognizing multiple touches on the surface.

4. *Touching and hovering.* A second Support Vector Machine is used to recognize touching and hovering of fingertips on and above the interactive surface. For this purpose, Agarwal et al. propose the calculation of a plane equation, whereby that plane slices the fingertip as it is illustrated in figure 3.13. At each point $X_i = (x_i, y_i)$ on the boundary of the tip point cluster from step 3, a disparity measure is expressed as $d_i = \alpha^T X_i + \beta$, where α and β are the parameters of the desired plane and $\alpha = [\alpha_1 \alpha_2]^T$. The points X_i are measured directly from the local coordinate system attached to the finger tip and d_i is obtained by the stereo matching technique. A discriminative classifier, providing a decision rule

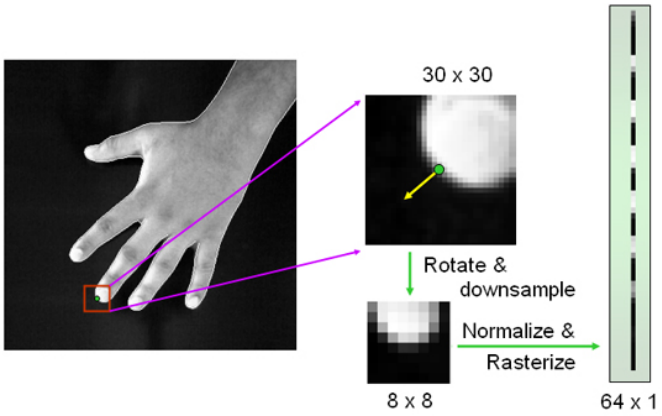


Figure 3.12: Scheme of the encoding process that converts 8-by-8 image patches into 64-dimensional signature vectors to be used with the Support Vector Machine. Courtesy of A. Agarwal et el. [AICB07].

on the parameters α and β of the equation, is learned by the second Support Vector Machine from manually labeled training data. The machine learning approach is said to provide robust touch and hovering detections by regarding the centroid of the calculated elliptical shape [AICB07].

According to [AICB07], the system provides touch sensing with a precision of 2–3 mm, which is similar to the accuracy of touch-sensitive surface hardware and a very good result for systems based on stereo vision. Another advantage of the proposed system is its portability. It can be moved to another location without the need for re-calibration. On the other hand, due to the use of a tablet display, scalability to other surfaces is limited. Agarwal et al. propose that other interactive surfaces than display devices may be used but in that case the image segmentation approach needs to be revised. Like other techniques based on Support Vector Machines, training is very important. Only if the training data is significant, good results can be expected. Once more, computational cost is an issue. According to [AICB07] the system provides 20 fps on a 3.4 GHz processor.

3.7.3 Letessier and Bérard

Letessier and Bérard presented their novel approach for visual tracking of bare fingers at the 17th annual ACM Symposium on User Interface Software and Technology in 2004 [LB04]. In contrast to the previously described vision-based techniques, the work by Letessier and Bérard does not propose an explicit physical prototype but a collection of straightforward computer vision algorithms to be used in a simple vision hardware setup. According to [LB04], off-the-shelf and inexpensive computer cameras, a stan-

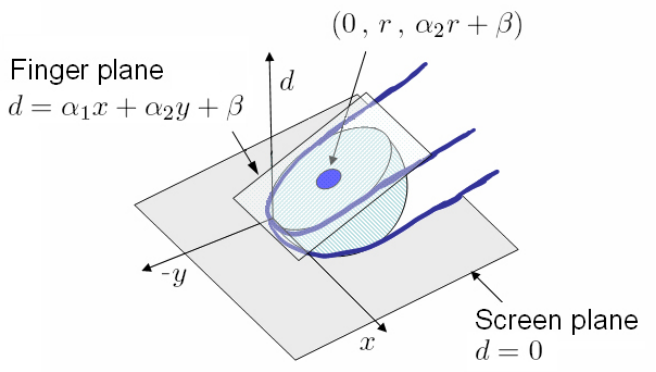


Figure 3.13: The figure illustrates the geometrical approximation of the fingertip by establishing a plane that slices the finger. The height of the ellipsoid's centroid formed by the fingertip shape is regarded for touch and hover detection. Courtesy of A. Agarwal et el. [AICB07].

dard projector device and a recent computer system (e.g. a portable laptop computer) are sufficient. It is recommended to assemble both camera and projector in a specific unit (similar to PlayAnywhere), so that the system can be mounted on the ceiling, for example. In the following, the prime stages (i.e. foreground extraction, automatic thresholding, shape filtering and event association) of the main algorithm are explained:

1. *Foreground extraction.* At first, similar to the other vision-based techniques, the camera image is segmented in order to provide an extraction of the desired foreground (i.e. hands) and remove distracting background from the scene. For this purpose, the approach makes use of an Image Differencing Segmentation (IDS) technique. IDS generally obtains a similarity map that yields a certain value of similarity for each of the pixels, when comparing the actual camera image (or frame) with a predefined (still) model of the background. If the similarity between two compared pixels is low, the probability that the actual pixel belongs to the foreground is high. The proposed foreground extraction procedure contains three steps:

    - *Background model.* For setting up the background model, typically pixel variations are regarded. Since relevant variations are mostly due to camera and lighting noise, Letessier and Bérard propose to model the background as an average image, where each of the pixels is the mean over time of recent pixel values measured at the corresponding position.
    - *Comparison metric.* In order to calculate the similarity of two pixels, an appropriate comparison metric need to be used. The approach suggests the use

of the Euclidean distance between the pixels in the (r, g)-normalized chromaticity plane, named the Chrominance Euclidean Distance (CED). According to [LB04], this should effectively remove shadows from the scene and save computational performance at the same time, due to the metric's mathematical simplicity. The segmentation of shadows is crucial, since hand shapes could be extended and no longer used for accurate shape filtering in the following [LB04]. The Euclidean distance d of the pixel p in the current frame and p' in the background model is computed as follows:

$$p = [R, G, B] \tag{3.1}$$

$$[r, g] = [R/(R + G + B), G/(R + G + B)] \tag{3.2}$$

$$d(p, p') = \|[r, g] - [r', g']\| \tag{3.3}$$

Figure 3.14 shows a typical similarity map obtained by the CED. It has to be said that the projected image is removed from the scene by causing overexposure on the camera image during the segmentation procedure.

- *Background maintenance.* Since the background of the scene may change over time (e.g. when objects pause in the view of the camera for a longer period of time), the background model needs to be updated constantly. As mentioned before, the pixels of the background image (Bg) are computed by averaging recent pixel values. The model at the time $t + 1$ is calculated according to:

$$Bg^{t+1}_{(x,y)} = \alpha^t_{(x,y)} \cdot Im^t_{(x,y)} + (1 - \alpha^t_{(x,y)}) \cdot Bg^t_{(x,y)} \tag{3.4}$$

Where Im denotes the current image and the value of α (i.e. the learning rate) is constantly updated during the computations and being influenced by the results of the IDS procedure. If a pixel is successfully detected as foreground in the actual IDS step, it influences the learning rate to be comparatively high in the following computation of the background model.

2. *Automatic thresholding.* To use the similarity map for further shape filtering it is converted to a binary map. Since manual thresholding is not desired, this is performed by an automatic thresholding approach. Here, a certain threshold θ^t is defined by regarding the similarity map's grayscale histogram. The histogram typically exhibits two modes. According to [LB04], the lower mode contains approximately 80% of the pixels, corresponding to background noise. Basically, θ^t should be chosen at each step to eliminate this mode, while preserving the rest of the data. This is achieved by approximating the median m_0 and the median

absolute deviation m_1 of the corresponding mode:

$$m_0 = median_{(x,y)} d^t(x,y) \tag{3.5}$$

$$m_1 = median_{(x,y)} \left| m_0 - d^t(x,y) \right| \tag{3.6}$$

$$\theta^t = m_0 + 4 \cdot m_1 \tag{3.7}$$

The automatic threshold is said to be stable and empirically close to manually chosen thresholds [LB04]. The binary map is then handed over to a shape filtering approach.

3. *Shape filtering.* The main idea here is to define a simple geometric model that describes the appearance of a typical fingertip (cp. figure 3.15). This model is basically presented as a set of characteristics (or criteria): $c_1, ..., c_n$, where $c_k(x,y) \in 0, 1$. In order to get accepted as being part of a fingertip, each pixel has to verify all of the characteristics, one after the other. Generally, these criteria are ordered according to their cost in performance and number of pixels they sort out. Low-cost characteristics that sort out many pixels are likely to be positioned at front of the serial shape filtering procedure in order to minimize the necessary computational effort. If a pixel does not verify a certain criteria, it is rejected. Hence, the procedure is called Fast Rejection Filter (FRF). The basic function of the FRF algorithm is further illustrated in pseudocode 3.1. Letessier and Bérard propose the following characteristics. To be accepted as part of a fingertip, a pixel p needs to be:

   - c1: classified as foreground in the binary map,
   - c2: within a region of connected pixels that is large enough to fit the dimensions of a hand (approx. 20 cm^2),
   - c3: surrounded by a fully segmented disc of 9 mm,
   - c4: part of exactly one connected component, while scanning the contour C and
   - c5: the distance AB has to be coherent with the width of a finger (9–20 mm).

   Figure 3.16 shows a typical detection image. Pixels that are not rejected are clustered into connected components and the centroid of each cluster is stored in a list of coordinates, which simply corresponds to an array of fingertip positions.

4. *Association.* For recognizing touches on the surface, the actual fingertip positions from the previous step are handed over to an association (or tracking) routine. Here, an ID is generated for every detected fingertip position, whereas a simple closest neighbor algorithm checks for three possible events:

- *Appearance.* If no fingertip has been detected in the previous frame, within a certain range of the current fingertip position (using a threshold), an appear event is recognized and a new ID is associated with the position.
- *Motion.* If the current fingertip position is further away to a previously detected position than a fixed motion threshold, a motion event is detected and the ID is associated with the new position.
- *Disappearance.* IDs that cannot be matched to any of the new positions generate a disappear event.

Since the approach is not able to detect actual surface touches, a spatiotemporal filter is included. If a fingertip does not move for a certain period of time (e.g. 300 ms), a touch (e.g. click) event is recognized.

Basically, the approach provides the software side of a state-of-the-art computer vision multi-touch interface and the possibility to easily build up a compact and portable interaction system. The final system is meant to be set up in a small amount of time and in almost any natural indoor environment, such as an office or meeting room, for example.

The proposed system mostly satisfies the requirements of HCI in tabletop environments. It allows about 4 users at the same time to interact at a frame rate of 25 fps on a 1.4 GHz portable Apple Macintosh computer. Finger detection is said to be stable, while robustness strictly depends on environmental lighting conditions, which have to be controlled. Latency of the system is 80 ms with a deviation of approximately 18 ms. This is not ideal, considering the initially mentioned maximum value of 50 ms. Furthermore, the overhead projection produces occlusions of the projected image by the hands of the users. [LB04]

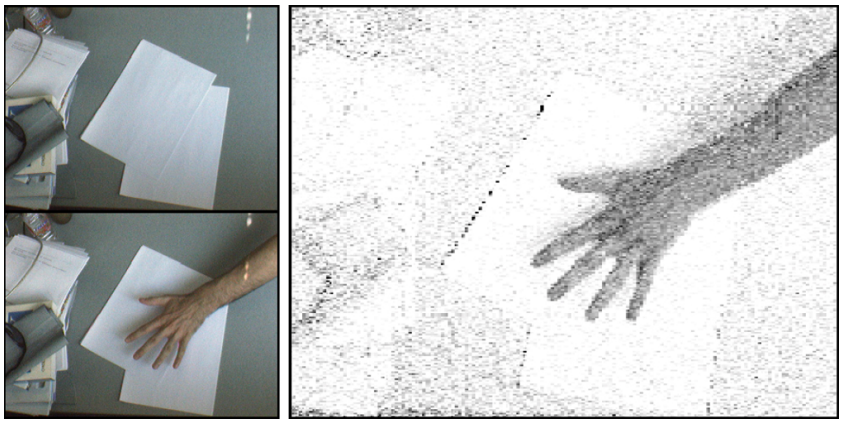


Figure 3.14: Typical segmentation result of the CED metric. The background model can be seen in the top left image, whereas the bottom left image shows the actual frame. The similarity map on the right hand side indicates a lot of visual noise but provides high resolution hand segmentation. Courtesy of J. Letessier and F. Bérard [LB04].

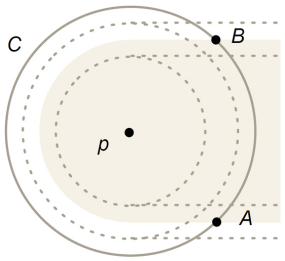
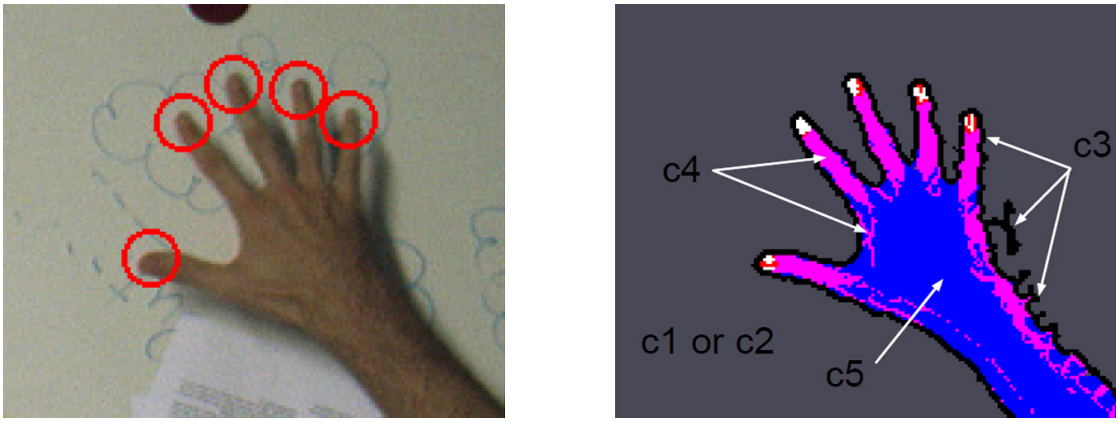


Figure 3.15: The diagram shows the simple geometric model of a regular fingertip. The observed finger shape (gray) is limited by the smallest and largest possible finger widths (dashed circles), spanning between A and B. C indicates the scanned contour. Courtesy of J. Letessier and F. Bérard [LB04].

Pseudocode 3.1 Fast Rejection Filter. Courtesy of J. Letessier and F. Bérard [LB04].

```
1:
2:  for each pixel (x, y) do
3:     for each k do
4:        if c_k = 0 then
5:           skip pixel
6:        end if
7:     end for
8:     mark pixel as a candidate
9:  end for
10: return  map of candidates
11:
```



(a) (b)

Figure 3.16: A typical detection image (a) of the approach with the detailed output (b) of the Fast Rejection Filter. (b) indicates what pixels have been rejected by which filter characteristic. Grey pixels have either been rejected by c1 or c2, black by criteria c3, pink by c4 and blue by c5. Pixels that have not been rejected are marked with white pixels, forming the detection clusters. Courtesy of J. Letessier and F. Bérard [LB04].

CHAPTER 4

Implementation Details & Practical Results

This chapter focuses on presenting the proposed vision-based system for the detection of fingertips on variable and dynamic interactive surfaces. In the following, important implementation details as well as practical results are explained.

The underlying approach is based on a specific pipeline of computer vision software algorithms processing images obtained by a video camera. Similar to related finger tracking approaches, such as [Wil05], [AICB07] and [LB04], the presented system is capable of recognizing fingertips that hover above a predefined surface. Generally, the system is meant to perform under regular indoor lighting conditions, where the camera is viewing the scene from an overhead-like position. Custom-made hardware components are not required. Basically, the intended configuration is designed to be preferably simple, light and cheap. The system should be easily portable to various locations and it should be possible to arrange the setup in a small amount of time. In contrast to the hardware-based touch technologies described at the beginning of chapter 3, the presented system supports variably sized surfaces, which should improve scalability. Furthermore, in addition to similar vision-based approaches, the system is capable of tracking the interactive surface continuously, so that the touch area does not necessarily have to be fixated in any kind of way and can be moved dynamically during runtime. Altogether, this should allow more flexible interfaces, where the interactive surface may be replaced easily with other types of different size or even shape to support multiple applications with the same basic system (e.g. supporting large tabletop interactions and small control panel applications by simply switching the interactive surface). General hardware and software design issues are presented in section 4.1.

It has to be mentioned that this work does not propose an explicit physical hardware prototype, as it is in [Wil05] and [AICB07]. Instead, it focuses on the selection of ap-

propriate computer vision algorithms and their implementation on the basis of a simple, cross-platform software application that is presented in section 4.2. The so-called multi-touch application is designed to take the video stream of the camera and automatically perform the main algorithm on the single frames. Moreover, each step can be applied separately and the corresponding results are visualized in an expressive manner.

The main algorithm of the presented approach consists of a series of computer vision methods and algorithms that are applied to the input images of the camera or the output of a preceding step, respectively. The three main parts of the algorithm are:

1. surface tracking based on rectangle detection and a vertex tracking algorithm (also called vertex tracker),

2. image rectification and calibration and

3. fingertip detection, including foreground extraction and shape filtering.

Initially, the interactive surface is detected in the scene by a geometrical rectangle detection technique, relying on straightforward edge detection and a Hough transform, similar to the page tracking functionality in [Wil05]. In short, straight lines in the camera view's edge image are transformed into a Hough space, which is then searched for pairs of parallel lines that match certain characteristics defining the geometrical shape of the interactive surface (section 4.3.1). The calculated surface vertices are further processed by a vertex tracker in order to obtain stable output before being handed over to image rectification and calibration routines that are explained in 4.3.2. Here, the image is rectified by removing the perspective distortion, resulting from the angle enclosed by the surface plane and the view of the camera, and narrowed to the region of the surface. The rectified image region is then calibrated according to the dimensions of the actual interactive surface. The fingertip detection algorithm, which is described in 4.3.3, takes the rectified image region from the previous step and performs a series of sub algorithms on the pixels. Initially, a foreground extraction technique is applied to the region to remove distracting background from the scene. The foreground is then processed by a shape filtering approach, similar to the one proposed in [LB04], in order to detect pixels that are likely to belong to a fingertip. Fingertip pixels are finally clustered into groups and their centroid is stored for further processing. In this approach, event detection (i.e. touch detection) is optional. In 4.3.4, some potential techniques for detecting physical surface touches are briefly presented to support future work.

At the end of this chapter, important experimental and practical results are explained in section 4.4. The section provides a basic overview of the prime contributions of this work to the field of computer vision-based surface tracking and fingertip detection in natural indoor environments.

4.1 Design Considerations

Generally, this work addresses both hardware and software needs of a modern multi-touch interface. The presented system basically consists of a recommended hardware configuration and a software component, managing the hardware input and providing a GUI to the user.

Concerning the hardware side, only off-the-shelf computer vision components are used for viewing the scene. Additional illumination is not intended and the interactive surface itself does not have to be specially prepared in any kind of way. Specific recommendations and hardware preliminaries are provided in the following. The software side of the interface is implemented using state-of-the-art open source programming libraries to provide cross-platform software scalability and portability.

4.1.1 Hardware Preliminaries

The proposed system uses commonly available hardware components, which makes the setup easy to configure. Figure 4.1 shows an image of a typical hardware setup. Basically, a standard camera device, an optional stand, a regular computer system and an everyday surface are sufficient. It has to be mentioned that the proposed system does not provide visual feedback that is superimposing the interactive surface. Thus, no projector device or additional LCD display is required in the first place. Visual feedback is exclusively provided at the GUI of the later described multi-touch application. Due to the fact that the system is meant to perform under indoor lighting conditions, the basic configuration does neither require special light sources, such as IR illuminants, nor infrared-pass filters for the camera.

Interactive Surface

For the interactive surface, basically any flat panel (also a tabletop) may be used. Due to the use of image rectification and a surface tracking approach, the panel does not have to be fixated and can be moved (e.g. translated, tilted, etc.) almost freely within the view of the camera. The freedom of movement is limited by the surface tracking. For example, the surface cannot be tilted for more than a certain amount of degrees. More information on that is provided in subsection 4.3.1. Moreover, since the surface tracking algorithm is actually expecting rectangular shapes, the interactive surface needs to provide a rectangular appearance of certain proportions (i.e. DIN format). However, the algorithm can be easily extended to support other than DIN proportions as well. Dimensions do not really matter, as long as at least one long enough part of each surface edge is visible within the camera view.

Generally, it is recommended to use a high contrast, homogeneous surface that can be easily detected in a natural environment. In case of a smaller, dynamically mov-

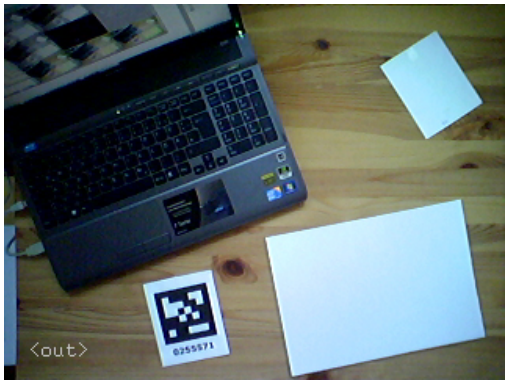


Figure 4.1: A typical hardware setup for the proposed vision-based system on the basis of the multi-touch application. A standard webcam is viewing the scene from an overhead-like position. The interactive surface, in that case, consists of a simple sheet of paper bonded to a panel made of cardboard that prevents from bending. Visual feedback is provided at the GUI of the multi-touch application.

ing surface (e.g. a control panel), a simple white sheet of paper bonded to a piece of cardboard that prevents from bending is sufficient. When working on a table, either the tabletop surface itself (e.g. if the camera view contains high contrast background) or a bright or dark panel can be used, respectively. The recommended color depends on the background of the scene. It has to be mentioned that the presented foreground extraction technique (during fingertip detection) regards the color of skin. Thus, the interactive surface should feature different color.

Webcam

For viewing the scene, a single color video camera, such as a regular webcam, is sufficient. The proposed technique has been developed and tested with a standard 1.3 mega pixels TerraTec TerraCam X2 (USB 2.0), which generally provides a video frame rate of 15 fps at 1280-by-1024 pixels. In order to sustain a decent frame rate during the main algorithm and keeping computational efforts low, images of 320-by-240 pixels at a rate of approximately 60 fps were used for testing and evaluation. Nevertheless, this resolution is common in similar vision-based finger tracking approaches [LB04].

Just as the interactive surface, the webcam does not have to be fixated and can be moved and positioned freely during runtime, as long as the interactive surface stays within the camera view. It is recommended to mount the webcam on a stand (e.g. a regular tripod) in order to provide a wide angle view of the scene. The used webcam does not obtain barrel distortion, so that initial camera calibration is not required.

Optional Hardware

The recommended hardware setup can be extended in several ways. In the following, a couple of ideas for optional hardware components are presented:

- A FireWire camera can be used [LB04] instead of the USB 2.0 device to improve data transfer rates. If required, higher resolution images can be processed (e.g. in large tabletop environments, where the proposed resolution of 320-by-240 pixels is not accurate enough).
- A wide angle camera can be used to support large tabletop environments. In that case, probably initial camera calibration is required.
- In order to improve the performance of the computer system, a graphic card with Computer Unified Device Architecture (CUDA) support (cp. subsection 4.1.2) can be used. In that case, the program code needs to be adapted to use CUDA functions.

4.1.2 Implementation

The software side of the proposed interface is implemented using the object-oriented programming language C++. This includes the GUI of the multi-touch application as well as the implementation and integration of the required computer vision methods and algorithms. For rapid developing and cross-platform portability, the GUI is entirely based on openFrameworks, a novel open source toolkit for scalable and portable computer software [Ope10b]. OpenCV is utilized for fast computer vision-related image processing (e.g. image conversion, filtering, edge detection, Hough transform, etc.) [Ope10a].

OpenFrameworks

OpenFrameworks is a novel C++ programming library by Zach Lieberman, Theodore Watson and Arturo Castro that provides a coherent software interface to several multimedia APIs such as OpenCV, OpenGL [Ope10c], RtAudio [Sca10], FreeType [Lem10], FreeImage [Dro10] and more. Generally, openFrameworks provides a simple and intuitive base for experimentation, creative coding and rapid software prototyping. It is open source software, so that any computer programming enthusiast may contribute to the community by implementing and sending in own extensions (i.e. add-ons) that are usually carrying the preamble `ofx`. The framework is strongly inspired by the Processing development environment, which is based on the Java programming language. OpenFrameworks generally implements the basic working principle of Processing in

C++. Here, applications basically consist of three main methods that are executed serially:

- `Setup()`
- `Update()`
- `Draw()`

`Setup()` is only run once, at the time when the application is started. Here, all of the initialization work should be done. `Update()` and `Draw()` are infinitely looped until the program is terminated. It is common to do processing during the `Update()` routine, which is executed one step before `Draw()`. The `Draw()` method is simply used to show the results on the screen. [Ope10b]

The general abilities of the openFrameworks environment are limited and cannot be compared to optimized C++ program code. Nevertheless, the library provides functions for fast and robust GUI implementation and supports various useful extensions, including a simple built-in video grabbing tool.

OpenCV

OpenCV describes a well-known open source programming library for real-time computer vision-related image processing. Generally, OpenCV is based on the proprietary Image Processing Library by Intel. The free library contains hundreds of optimized algorithms for, amongst others, image processing and machine learning. [Ope10a, BK08] OpenCV code is by default included in openFrameworks (i.e. `ofxOpenCV`), so that corresponding functions may be used out-of-the-box within the program code.

Optional Software

Since geometrical calculations, such as the later presented surface tracking approach, take a lot of computational performance, the requirements to the computer system are relatively high, especially when using high resolution in real-time processing. For that reason, it is generally preferable to source out these calculations to an additional processing unit. Computer Unified Device Architecture (CUDA) describes a novel technique that provides a special graphical processing unit situated at the computer graphic card [Nvi10]. Concerning real-time computer vision applications, CUDA is relevant for speeding up computations.

4.2 The Multi-Touch Application

A prime contribution of this work is the implementation of a multi-touch application, which provides a simple and beneficial experimental basis for successively applying related computer vision algorithms to a stream of live images obtained by one or more computer video cameras.

Generally, the application is designed to be easily portable across multiple platforms, including Microsoft Windows, Apple Macintosh and Linux distributions. The algorithmic part of the application is sourced out to form an openFrameworks add-on, named `ofxMultitouchRecognition`. Here, all of the image processing is performed by combining self-implemented algorithms and standard OpenCV functions. The application's GUI is implemented separately to provide a useful demonstration tool to other programmers that are trying to reuse and enhance the software. Figure 4.2 shows the main window of the GUI. In the following simple user manual, the basic functions are described briefly.

User Manual

The unprocessed webcam image (<*in*>) is shown on the bottom left of the window. The output of the actually selected algorithm is shown in the middle image on the bottom (<*out*>). The large and small workspaces above the two images and on the bottom right of the window are used to display intermediate steps of the single algorithm parts. On the right hand side, the user can make a couple of adjustments to the program by pressing the corresponding key on the keyboard. Key *a, A* sets the application mode to automatic, which means that the main algorithm is automatically performed on the input images of the webcam and the output is displayed in the middle image on the bottom of the main window. By pressing *s, S* on the keyboard, the user can set the actual size of the interactive surface, which is important for calibration and shape filtering. The user can choose between a number of standard DIN proportions (*A0–A5*). Key *d, D* adjusts the general self-estimated brightness level of the input images. Here, the user may choose between *bright*, *regular* and *dark*. The selected level is used within the code to adjust the parameters of the single algorithms to the actual lighting conditions. By pressing and holding *F2*, the user can set points for image rectification in the input image (<*in*>). This is requested to manually mark the vertices of the interactive surface and to tell the surface tracker the default vertices to follow. Pressing and holding *F3* simply augments the mouse cursor with a color picker function that is executed on the output image (<*out*>). By pressing *m, M* on the keyboard, the input image can be mirrored as an initial image processing step, so that overhead viewing issues may be corrected. Each single step of the main algorithm may be executed by pressing the corresponding key (*1–8*). It has to be mentioned that most of the steps require that the previous step

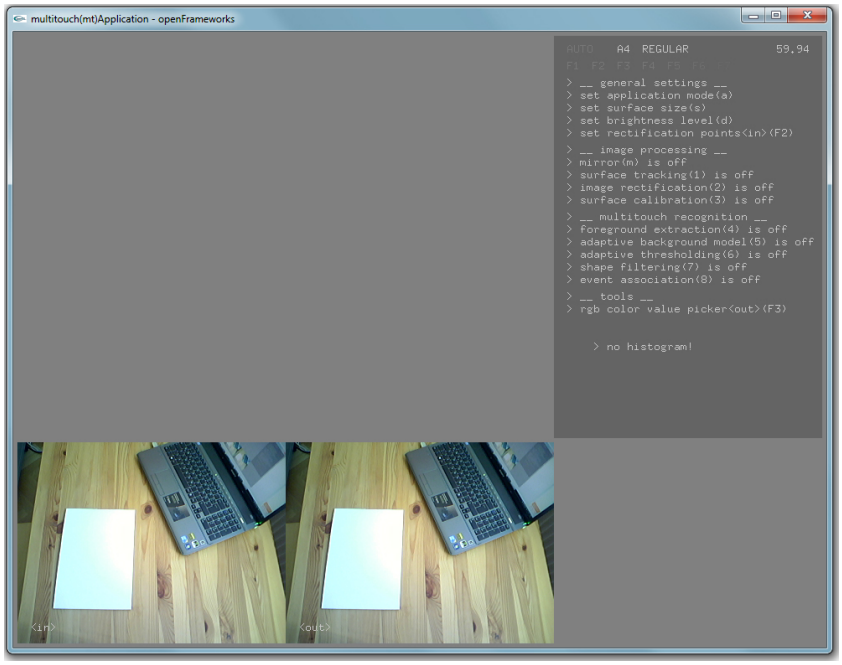


Figure 4.2: The main window of the multi-touch application GUI is split up in three parts. On the bottom of the window, the input and processed output image streams are displayed. In workspaces above and next to the images (i.e. empty gray spaces), intermediate results are provided. On the right hand side, the user can make various adjustments to the application.

is enabled. For instance, all further image processing steps can only be applied if the surface is rectified and calibrated first.

4.3 Computer Vision Pipeline

In the following, the computer vision pipeline of the main algorithm is explained in detail and the working principle of the single components is demonstrated by pseudocodes as well as illustrated by example images. All of the images have been established by the previously described multi-touch application.

4.3.1 Surface Tracking

The vision-based tracking of objects through a series of natural images is a challenging task. Typically, the process involves detecting object features in a single frame and following them to their new positions in the subsequent frame. The tracking proce-

dure is heavily influenced by the quality of the features, which means in this case their robustness against varying lighting conditions and visual occlusions. [LA97]

Since the interactive surface provides a specific rectangular shape, it is suitable to use geometrical information as a main feature for tracking. A major advantage here is that geometrical constraints usually do not change under varying illumination, in contrast to other object features like color or texture information, for example. Furthermore, occlusions are indeed a major problem here. When the interactive surface is used for manipulation, it is partially occluded by the users' hands, which makes the tracking of the surface itself difficult.

To address these problems, a number of rectangle tracking techniques, such as [Wil05], [HPM03] and [JS04], propose to use edge images together with the Hough transform for feature tracking. Typically, straight lines are found in edge images produced by a straightforward edge detection algorithm (e.g. Canny, Sobel, etc.). The lines are basically represented by their offset (i.e. distance from the origin) and angle in the two-dimensional image plane. The Hough transform establishes an accumulator plane in which for each combination of the two parameters an individual value is stored. The value represents the probability of the presence of a single feature that matches the parameters of the cell [HPM03]. The accumulator plane is commonly referred to as Hough space. This Hough space is later searched for lines that fit certain criteria that match the appearance of the desired shape. In the case of a rectangular surface to be searched, criteria are the edge lengths and angles between two pairs of parallel lines, for example. Generally, the Hough transform is computationally expensive and its use for real-time computer vision processing is critical. An optimized version of the Hough transform is included in the OpenCV library.

This approach is not based on matching criteria directly in the accumulator plane, as it is proposed in [HPM03] and [JS04]. Because for this purpose, the Hough transform needs to be implemented manually and this did not seem feasible in the terms of this work. Instead, the pre-implemented OpenCV version of the Hough transform is used to obtain Hough lines from the edge images and using these lines for a geometrical line and rectangle detection technique, similar to [LA97]. In the following, the proposed surface tracking approach, including a geometrical rectangle detection technique and a vertex tracker, is explained in detail.

4.3.1.1 Rectangle Detection

The proposed rectangle detection is based on the following single steps:

1. *Image conversion and blur.* The unprocessed color image obtained by the webcam is initially converted to grayscales, using `cvCvtColor()`, and blurred with a Gaussian filter, using `cvSmooth()`, to remove visual noise. This is very important for the later edge detection and Hough transform. The more the image is

blurred, the viewer edges are detected and the less computational expensive the Hough transform gets. On the other hand, it is important not to eliminate too much edges in the image. The size of the filter kernel needs to depend on the global image luminosity level (i.e. general image brightness). As explained before, the actual level needs to be estimated by the user at runtime and is entered within the GUI before being handed over to the surface tracking routine. If the image is generally bright, the filter kernel needs to be comparatively large to remove enough noise (i.e. weak edges). On the other hand, a dark image requires the filter kernel to be small, not to loose too much edge information. Generally, filter kernels between 13-by-13 and 21-by-21 have proven to suit regular indoor environments. Figure 4.3 shows the starting point of the surface tracking approach: a typical input image and the result of the image conversion. The blurred image can be seen in figure 4.4(a).

2. *Edge detection.* The well-known Canny algorithm is used for edge detection instead of the Sobel operator that is used in [HPM03]. Empirical experimentation showed that in this approach Canny simply removes noisy edges more successful than the Sobel algorithm. The parameters of the OpenCV function `cvCanny()` need to be adapted to the choice of the previously used Gaussian filter kernel. More information on that is provided during the later presentation of practical results in section 4.4. Figure 4.4(b) shows a typical result of the edge detection step. The output is a binary image, where detected lines (i.e. strong edges) are presented as white pixels.

3. *Hough transform.* The Hough transform, which is executed by calling the corresponding OpenCV function `cvHoughLines2()`, is applied to the binary edge image from the previous step to retrieve an array of detected Hough lines. For each of the lines, the orthogonal distance to a parallel line, intersecting the origin, length and slope are stored in a separate feature space. Figure 4.5 shows the input image with detected Hough lines. The second thin line in image 4.5(b) indicates the orthogonal distance of the line to the origin, which is situated at the top left corner of the image.

4. *Geometrical rectangle detection.* The feature space of straight Hough lines is then searched for pairs of almost parallel lines. Therefore, the angle distance of the two lines needs to be smaller than a certain threshold in order to get accepted as a parallel pair. The recommended threshold of $30°$, as proposed in [LA97], is further restricted to $10°$ to eliminate more false positive detections. Nevertheless, this restricts the tilting of the interactive surface as well. Furthermore, the two lines must not be closer than a manually predefined threshold of 10 pixels in order to reduce the amount of possible pairs. The parallel line pairs are further

sorted into another feature space, which is then searched for two almost orthogonal pairs. Here, the angle of the two pairs does not have to be lower or greater than a specified deviation (i.e. $5°$) from the optimal value of $90°$. A third feature space of possible rectangle candidates is filled with the detected orthogonal line pairs. Pseudocode 4.1 furthermore demonstrates the working principle of the geometrical approach and the corresponding result is shown in figure 4.6(a). Here, white circles indicate possible surface vertices that belong to a set of two orthogonal line pairs, matching the previously described criteria. For each of the candidates a probability value is established. The candidates are further analyzed according to certain characteristics that define the interactive surface. Proposed characteristics are as follows:

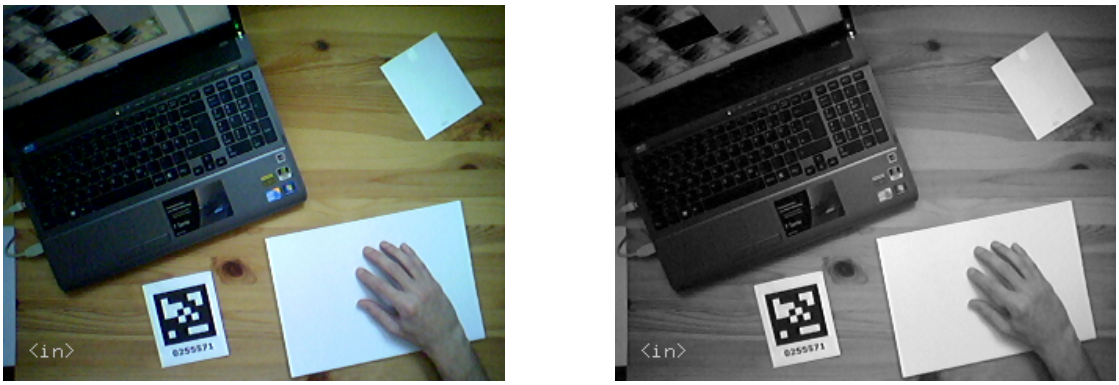
- The rectangle enclosed by the two pairs of parallel lines needs to provide a certain aspect ratio (i.e. DIN format).
- The length difference of two parallel edges has to be smaller than a predefined threshold of 10 pixels.
- The rectangle area has to be greater than a predefined value of 5 percent of the image area.

The output of the geometrical rectangle detection technique is defined by the vertices of the one rectangle with the highest probability. It has to be mentioned that, due to the many possible combinations of parallel line pairs, the algorithm obtains duplicate rectangle detections that are at the almost same position and basically overlap each other.

The first found surface vertices are finally handed over to the vertex tracking algorithm that is described in the following.

4.3.1.2 Vertex Tracker

The vertex tracker should provide smooth tracking without flickering, even if the previous detection algorithm fails to detect the surface for one or more frames. Here, at each frame, the actual vertices are compared to a combination of two separate buffers. If a surface is successfully detected, its vertices are stored in a primary buffer (in relation to previously buffered vertices). If the surface is currently not detected, the vertices from the primary buffer are used. For convenient results, the desired surface needs to be initially marked by the user via the multi-touch application GUI. The tracker stores these vertices in a secondary buffer so that later detections may be analyzed according to those. New detections are only considered if they are spatially near the predefined vertices. Pseudocode 4.2 demonstrates the exact working principle of the tracking procedure.



(a) (b)

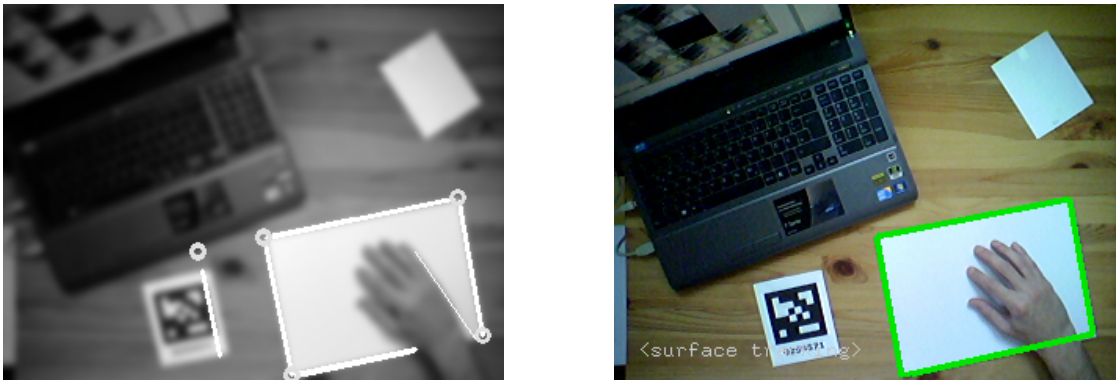
Figure 4.3: The figure shows a typical input image (a) and the initial conversion to grayscales (b).

(a) (b)

Figure 4.4: Result of the Gaussian blur filter with a filter kernel of 13-by-13 (a) and the application of the Canny edge detector on the blur image (b).

(a) (b)

Figure 4.5: The result of the Hough transform on the edge image (a). A single line example is depicted in (b). The shorter line indicates the orthogonal distance to the image origin (i.e. top left image corner).



(a) (b)

Figure 4.6: The figure shows the general output of the geometrical rectangle detection approach (a). The pairs of parallel lines are indicated by thick white lines and the sets of possible rectangle candidates are presented as thick white circles. The final result of the vertex tracker is indicated by the green rectangle, which is overlaying the initial input image (b).

Pseudocode 4.1 Geometrical rectangle detection.

```
 1:
 2: fspace_lines, fspace_pairs, fspace_orthogonal pairs {line, pair, orthogonal pair feature spaces}
 3:
 4: for each line in Hough space do
 5:    store orthogonal distance, length and slope
 6: end for
 7: return fspace_lines {containing straight lines}
 8:
 9: for each line in fspace_lines do
10:    find parallel lines
11: end for
12: return fspace_pairs {containing pairs of parallel lines}
13:
14: for each pair of parallel lines in fspace_pairs do
15:    find orthogonal pairs of parallel lines
16: end for
17: return fspace_orthogonal pairs {containing sets of two orthogonal line pairs}
18:
19: for each element in fspace_orthogonal pairs do
20:    calculate probability
21: end for
22: return list of possible rectangles
23:
```

Figure 4.6(b) shows the output of the vertex tracker. It can be seen that the interactive surface is successfully recognized in a natural environment. The surface vertices and the corresponding image region are further processed by the following rectification and calibration procedures.

4.3.2 Image Rectification & Calibration

Generally, image rectification is required to bring spatial two-dimensional coordinates into correspondence, as it has been already mentioned in subsection 3.7.1 to the explanation of the vision-based approach by Wilson.

Basically, there are two image planes to cope with. The plane in which the physical interactive surface resides and the image plane of the scene established by the camera view. Points within the coordinate system of the interactive surface need to be mapped to their scaled equivalents in the on-screen coordinate system. Usually, this procedure is referred to as homography and the mapping can be expressed by a 3-by-3 orthogonal

Pseudocode 4.2 Vertex tracker.

```
1:
2: vertices_actual, vertices_predefined, vertices_default, vertices_tracker {actual, predefined, default, tracker vertices}
3: buffer_1, buffer_2 {vertex buffers}
4:
5: if surface has been predefined then
6:    buffer_2 = vertices_predefined
7: end if
8:
9: if surface is currently detected then
10:   if surface has been detected during the previous frames then
11:      if surface has been predefined then
12:         if surface is near predefined surface then
13:            buffer_1 = buffer_2 = vertices_actual
14:         else
15:            buffer_1 = buffer_2
16:         end if
17:      else
18:         buffer_1 = vertices_actual
19:      end if
20:   else
21:      if surface has been predefined then
22:         if surface is near predefined surface then
23:            buffer_1 = buffer_2 = vertices_actual
24:         else
25:            buffer_1 = buffer_2
26:         end if
27:      else
28:         buffer_1 = vertices_actual
29:      end if
30:   end if
31: else
32:   if surface was visible during the previous frames then
33:      vertices_tracker = buffer_1
34:   else
35:      if surface has been predefined then
36:         vertices_tracker = buffer_2
37:      else
38:         vertices_tracker = vertices_default
39:      end if
40:   end if
41: end if
42:
```

matrix, also described as warp matrix [BK08].

Due to the freedom of movement of both the interactive surface and the camera, the angle between the camera lense and the surface changes over time, so that the image rectification procedure needs to be continuously applied to the input image at each frame. Generally, known points in the image are established and the orthogonal matrix is computed for the later transformation of the whole image.

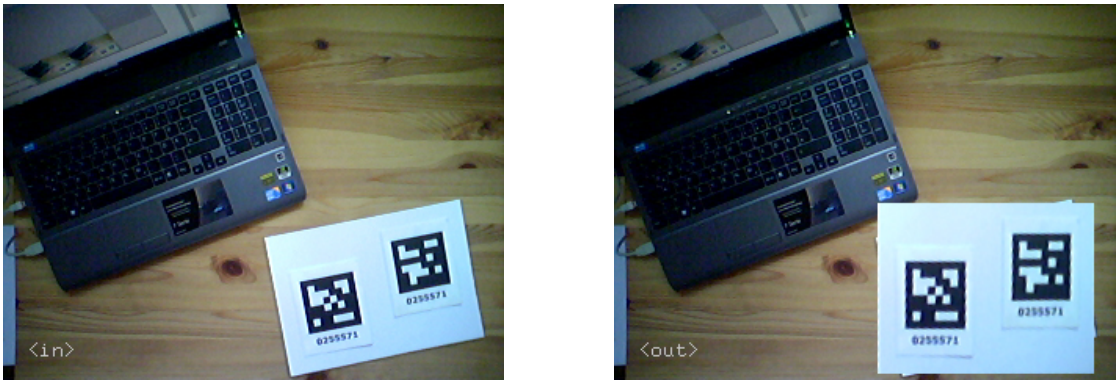
In this approach, the vertices of the interactive surface are used for image rectification. These can either be manually set points, directly obtained by the user within the GUI, or automatically determined vertices resulting from the previous surface tracking. Initially, the top left, top right, bottom left and bottom right points are determined by a straightforward algorithm. After that, the warp matrix is calculated by using the `cvGetPerspectiveTransform()` function that matches the original vertices in the surface plane to their new positions in the on-screen image plane. The matrix is then used for calling `cvWarpPerspective()` in order to rectify the whole image according to the calculated perspective transform. Figure 4.7 shows the result of the image rectification procedure on simple test patterns. The figure shows that only the area of the patterns (i.e. in that case, the interactive surface) is visualized, which is overlapping the original image. The area is internally also called the region of interest (ROI) and used for further processing. Any later algorithms are exclusively processing this ROI in order to save computation time and to remove visual noise, residing alongside the physical interactive surface.

For calibrating the system, the user has to simply choose the current size of the used surface from a selection of common DIN formats. The conversion factors between the actual dimensions of the surface and the ROI are obtained by a straightforward calculation and stored in a conversion table for later usage.

4.3.3 Fingertip Detection

The fingertip detection algorithm presented in this work focuses on analyzing geometrical shapes in binary images. For that purpose, the rectified ROI provided by the previous surface tracking and image rectification is initially segmented into desired foreground (i.e. hands) and distracting background. As mentioned before, the segmentation process is critical, as it predetermines the results of the further procedures.

In the following, a suitable technique based on skin detection is presented and compared to the results of the adaptive background model proposed in [LB04]. The foreground pixels are later processed by a specific shape filtering technique that obtains fingertip pixel clusters.



(a) (b)

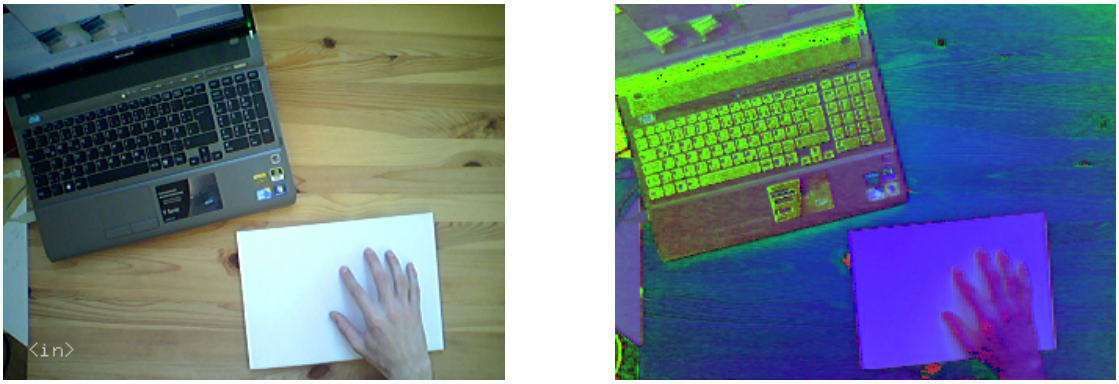
Figure 4.7: The figure shows an example input image with two test patterns (a) and the corresponding output of the image rectification approach (b). The image region detected by the previous surface tracking approach is rectified and used for further processing.

4.3.3.1 Foreground Extraction

Foreground extraction here refers to the process of segmenting pixels that belong to the hands of the user(s) from the rest of the scene. The background generally includes other objects that are residing on the interactive surface as well as visual feedback that may be projected onto the surface. In this approach, visual feedback is not present and therefore not interfering with the segmentation process. In a real application setup, projections can be removed from the scene by causing overexposure, as it is proposed in [LB04].

The main problem throughout segmentation are shadows resulting from the ambient lighting. Shadows may extend hand shapes and make these inappropriate for the later shape filtering. That is why the required foreground extraction technique needs to be robust against shadows. The suggestion of Letessier and Bérard to use an adaptive background model has been the starting point of this work. Unfortunately, the self-implemented version of the IDS technique did not show the same results as proposed in [LB04]. Figure 4.14(b) shows the application of the algorithm to the input image shown in figure 4.8(a). As it can be seen, only the outer shape of the hand region is detected as foreground, leaving the inner side without enough texture information for the later shape filtering. Moreover, the segmentation is noisy and fails to detect the whole outer shape. This may be lead back to an erroneous implementation or inappropriate lighting conditions. More on the performance of IDS may be found in the following subsection 4.4.2.

For that reason, another extraction technique that is naturally robust against shadows has been chosen, namely the detection of skin. As proposed in [ATD01], it is suitable to



(a) (b)

Figure 4.8: A typical input image (a) and a visualization of its conversion to the HSV color space (b).

perform skin detection in the HSV color space. In comparison to RGB, the HSV space provides hue, saturation and value (i.e. lightness) for each pixel. The range of elements in the image that are likely to be equivalent to the color of skin can easily be detected by the standard OpenCV function `cvInRangeS()`, as proposed in [And10]. The function compares each pixel to a three-dimensional range of values. If the desired pixel color lies in the range of skin color, the function returns a white pixel at the position. Otherwise, a black pixel is returned. This leads to a binary image where regions that show similar color to skin are segmented. Figure 4.9(a) shows the application of the skin detection technique to the typical input image from 4.8(a). A visualization of the corresponding HSV image is shown in 4.8(b). Appropriate parameters for the OpenCV function have been chosen similar to the ones proposed in [And10] and [ATD01]. Nevertheless, the parameters strictly depend on the actual lighting situation and need to be adapted to the globally estimated luminosity level as well.

To remove holes and obtain a dense segmentation, a small median filter for smoothing (i.e. 3-by-3 filter kernel) is further applied to the ROI, using the `cvSmooth()` function. Figure 4.9(b) shows the result of the hand segmentation after median smoothing. The output image is handed over to the shape filtering algorithm.

4.3.3.2 Shape Filtering

The proposed shape filtering technique is very similar to the approach by Letessier and Bérard but uses different filter characteristics and another working principle. The implementation is exclusively based on standard OpenCV functions.

Generally, the shape filtering approach uses a specific dual template matching technique. Template matching basically refers to an image processing method, where a cer-

(a) (b)

Figure 4.9: The application of the skin detection technique to the detected and rectified image region (a). The final image segmentation result after applying a 3-by-3 median filter for smoothing is shown in (b).

tain shape (i.e. a template or image patch) is moved over the input image and matched with the underlying image region, according to a comparison metric. The process results in a grayscale map, where for each pixel a value is given that describes the similarity between the region around the pixel with the predefined image patch. [BK08]

In this approach, the two two-dimensional templates are designed to match the smallest and largest possible fingertip sizes (12.5–22.5 mm) in the binary segmentation image. The shapes are continuously rescaled before each computation step (i.e. at each frame), as the actual conversion factor from physical to on-screen coordinates may change at runtime, when the interactive surface moves further away from or nearer to the camera. It can be said that the size of the two template matching shapes depends on the predefined thresholds as well as the actual size of the interactive surface. The shape filtering procedure is described in the following:

- *Initial connected-component analysis.* At first, the segmentation image is processed by a connected-component analysis that searches for connected groups of pixels by applying `cvFindContours()`. The area of the connected region needs to be within low and high thresholds, corresponding to the actual size of a human hand, which is approximately preset between 10–25 cm^2.

- *Dual template matching.* For each of the found components (i.e. hand regions), the dual template matching is performed, using the standard OpenCV function `cvMatchTemplate()`. Initially, the region is matched with the small shape and after that with the large shape. The process leads to two different template matching maps. Examples are shown in figure 4.10. As it can be seen from figure

4.10(a), the matching with the smaller shape segments the outer hand shape. In fact, it produces large values in the matching map for all of the pixels that are not further to the edge of the hand shape than the half of the smallest possible fingertip size (i.e. the size of the smaller template matching shape). Furthermore, the figure shows the effect of a small hole in the previous segmentation image to the template matching map. Figure 4.10(b) shows the application of the larger shape. Here, especially the finger regions themselves return large values in the matching map, as they almost exactly match the larger shape.

- *Shape filtering.* In order to get accepted as a fingertip pixel, a pixel needs to verify all of the following characteristics. The pixel needs to:
  - be detected as foreground in the binary segmentation image,
  - provide a strong value in the first template matching map and
  - provide a very strong value in the second template matching map.

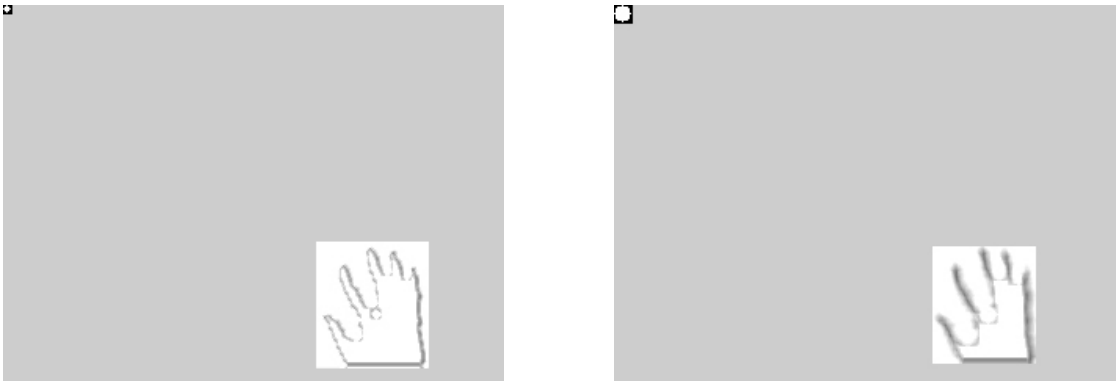
  The accepted values are defined by manually selected thresholds that have been obtained by empirical testing. Finally, the shape filtering approach leads to a so-called detection map in which possible fingertip pixels are presented. Figure 4.11(a) shows the detection map of the actual example. If a pixel verifies all of the conditions, a white pixel is returned at the position. Otherwise, a black pixel is returned. As it can be seen in the figure, there are some noisy pixels on the right side of the hand. To remove such noise, another connected-component analysis is performed on the detection map.

- *Final connected-component analysis.* Another connected-component analysis is used to find actual fingertip pixel clusters (i.e. pixel clusters that are large enough to belong to a fingertip). The centroid of each cluster (cp. figure 4.11(b)) is returned in a list of fingertip positions.

The accuracy of the detected positions is estimated at millimeter level (i.e. approximately 2–10 mm). The results of the technique primarily depend on the image segmentation and the correct adaptation of the template sizes. Holes in the segmentation may produce many FPs. If two hands overlap each other or strive near each other, they are commonly grouped into one connected-component. This does not affect the shape filtering approach as long as the two hand shapes are clearly segmented.

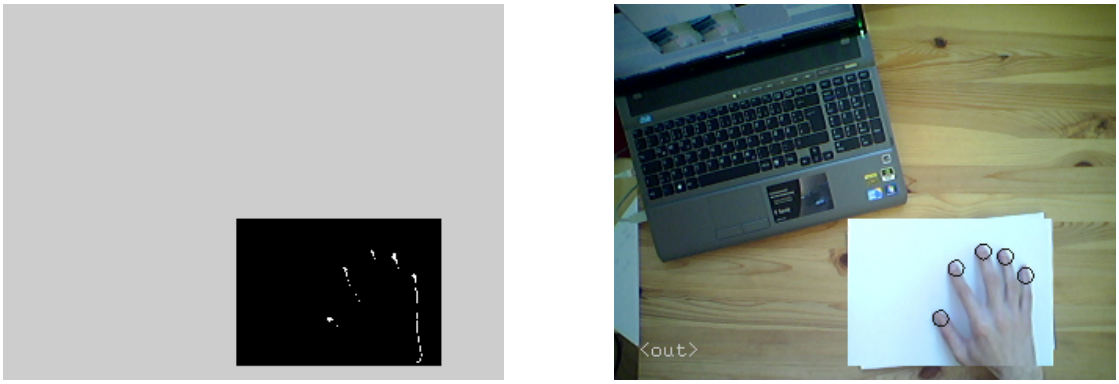
4.3.4 Event Detection – Potential Techniques

As initially mentioned, this work does not propose an explicit approach for event detection, referring to the detection of physical surface touches. In the following, a couple of



(a) (b)

Figure 4.10: The shape filtering approach is based on a dual template matching technique, including both small and large image patches that are roughly representing the minimum and maximum sizes of typical fingertips. The resulting template matching maps are shown in (a) and (b), whereas the corresponding patches are displayed in the top left of the two images.



(a) (b)

Figure 4.11: The detection map consists of fingertip pixel clusters (a). The cluster centroids are augmenting the rectified input image (b).

suitable techniques are presented that would basically fit the constraints of the proposed vision-based system:

- *Shadow shape analysis.* A technique similar to the one proposed in [Wil05] and described in 3.7.1 could be used for touch detection. However, the original approach by Wilson uses infrared lighting conditions, so its adaption to natural lighting conditions is not straightforward, because the orientation of shadows typically changes in natural environments. Moreover, shadow intensities may change as well. An appropriate technique would be to mount a specific light source to the camera, so that the angle of the incoming light and the shadow intensities are almost constant during runtime. The angle of light could be calculated according to the previously determined viewing angle, for instance during calibration.

- *Stereo matching.* Regarding a very accurate vision-based detection of physical surface touches, a technique similar to [AICB07] that uses a stereo vision pair could be suitable. In that case, the surface tracking approach would have to be revised. Under stereo vision conditions, the surface tracking needs to be performed in one of the camera views and the corresponding surface vertices in the second camera view are found by using the epipolar geometry. As the interactive surface is movable and the epipolar geometry changes continuously, the reconstruction of the three-dimensional fingertip position for surface touch detection would not be straightforward to implement. Altogether, such an approach would make the system more complicated to implement, bulky and expensive.

- *Depth analysis.* If available, a TOF camera could be a potential alternative to the previously described touch detection approaches, since both surface tracking and fingertip detection could be managed by a single device. In that case, the depth image obtained by the TOF camera could be searched for a certain area of equal distance to the camera, which indicates the existence and position of the interactive surface. Moreover, the difference in distance from the surface and the users' hands could be used for obtaining touch positions. Nevertheless, the segmentation of the hand region would not be trivial. As TOF cameras are non-standard computer hardware, the system would get more expensive and bulky as well. Moreover, TOF devices provide comparatively poor resolution (i.e. mostly lower than 320-by-240 pixels) on the one hand but very high frame rates on the other hand. However, the latter would be practical for real-time image processing.

4.4 Experimental & Practical Results

This section provides essential experimental and practical results that have been established throughout the experimentation with computer vision algorithms in the terms of

tracking rectangular shapes and detecting fingertips in natural indoor environments.

Generally, the presented multi-touch application performs at a frame rate of approximately 20–40 fps, when both surface tracking and fingertip detection are enabled. For testing and evaluation, a portable Sony Vaio laptop computer has been used, which is based on a quad-core Intel i7 1.73 GHz processor with 6 GB of RAM. The used operating system is Windows 7 64-bit. The frame rate mainly depends on the number of lines that are considered during the rectangle detection technique and the number of connected components that need to be processed by the shape filtering approach, which corresponds to the number of visible hands hovering the interactive surface. Generally, the performance of the system is similar to the approaches by Letessier and Bérard and Agarwal et al. that function at 25 and 20 fps, respectively. It has to be mentioned that the rate alludes to the selected region of interest and not the whole image of 320-by-240 pixels. The performance can partly be optimized, considering that the rendering of the GUI takes some computation time as well. However, the maintained frame rate is at the bottom end of appropriate values for good usability. In the following, important results are categorized according to the corresponding part of the main algorithm.

4.4.1 Surface Tracking

Regarding the surface tracking approach, experimentation showed that the standard OpenCV function for the Hough transform obtains robust detection of straight lines in the edge images. The function can be widely adapted to the specific purpose by choosing the number of points in the accumulator plane cell that a certain line needs to provide to be returned by the function. This is used to effectively filter out very short or very long lines in order to suppress unneeded lines for the later search for parallel line pairs. Generally, the search for orthogonal pairs of parallel lines is rather straightforward but produces many almost duplicate orthogonal line pairs. It can be the case that several orthogonal line pairs are detected at almost the same position, since usually a couple of parallel line pairs are found in a certain region. Those produce possible rectangle matches with exactly the same probability of being a good match, as appropriate filter characteristics for rectangles are of course limited. This results in slight flickering of the detected rectangle, as it may jump from one of the possible rectangles to the other during the frames. At each step, simply the first found rectangle is used, since it is difficult to sort out almost duplicate detections at this stage. To this end, the proposed algorithm does not include elimination of duplicates. A suitable technique here would be to include a nearest neighbor search, considering the previous frame, so that only the nearest rectangle with the highest probability is regarded.

Generally, the proposed rectangle detection is a tradeoff between adjusting the Hough transform to returning shorter or longer lines. Short lines will massively increase the necessary computational effort during later comparison and will produce a lot of detection noise, as the number of possible rectangles increases massively. Accepting long

lines will reduce the systems ability to detect partly overlapped interactive surfaces, as surface edges are indeed disconnected by the hands of the user(s). For this reason, the initial blurring needs to be adapted accordingly to eliminate weaker edges in the image before the Hough transform is applied. The choice of the filter kernel for Gaussian filtering is critical, because it mainly depends on the current lighting condition. If the image is generally dark, the filter kernel needs to be small not to eliminate strong edges (i.e. high contrast changes) in the image. On the other hand, if the image is generally bright, the filter kernel needs to be large in order to suppress weak edges (i.e. small contrast changes). Figure 4.12 shows two more examples of blurring the input image from figure 4.3(b) with different Gaussian filter kernels. The corresponding edge images in figure 4.13 show how various edges are lost, due to the stronger initial blur. It has to be said that the optimal Gaussian filter kernel in the case of the presented example is 13-by-13. Figures 4.12(a) and (b) have been blurred with kernel sizes of 17-by-17 and 21-by-21.

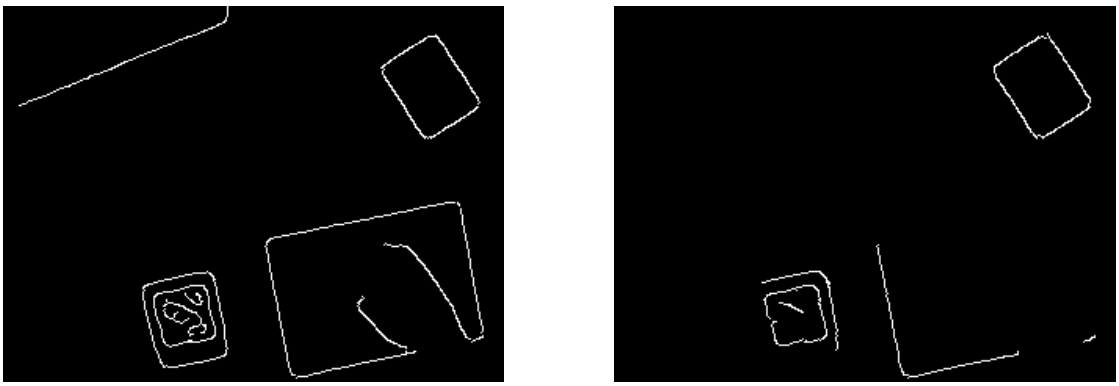
The parameters of the OpenCV Canny function need to be adapted to the Gaussian kernel size as well, respectively to the global image luminosity level. The larger the Gaussian kernel is, the lower the parameters of the Canny algorithm need to be. The function generally takes two relevant input arguments (i.e. a low and a high threshold). If the pixel gradient is below the lower threshold, it is rejected. On the other hand, if the pixel gradient is larger than the higher threshold, it is accepted as an edge pixel. If the gradient lies between the two thresholds, it is only accepted if it is connected to a pixel that is accepted as an edge pixel. According to [BK08], both should provide a ratio of 1:2 to 1:3. Experimentation showed that low thresholds of 50–100 and high thresholds of 150–200 are appropriate.

4.4.2 Fingertip Detection

Concerning fingertip detection, the results generally depend on the foreground extraction procedure (i.e. the image segmentation). Experimentation showed that the skin detection approach is efficient in producing a dense segmentation of the hand region, which is essential for the performance of the whole approach. The self-implemented version of the adaptive background model proposed by Letessier and Bérard (cp. figures 4.9(b) and 4.14(b)) did not provide comparable results by any means. Even by extensive empirical testing of various thresholds, instead of the automatic thresholding approach, the obtained similarity map could not be prepared to fit the requirements of the later shape filtering algorithm. As it can be seen from figure 4.14(b), the resulting hand segmentation is not dense and does not obtain texture information at the inner side of the hand region, which is a major problem for the later shape filtering. This is due to the fact that homogeneous regions have very low influence on the background model during the algorithm as the difference of the color values between the actual frame and the background model is not very large. Nevertheless, the adaptive background model

(a) (b)

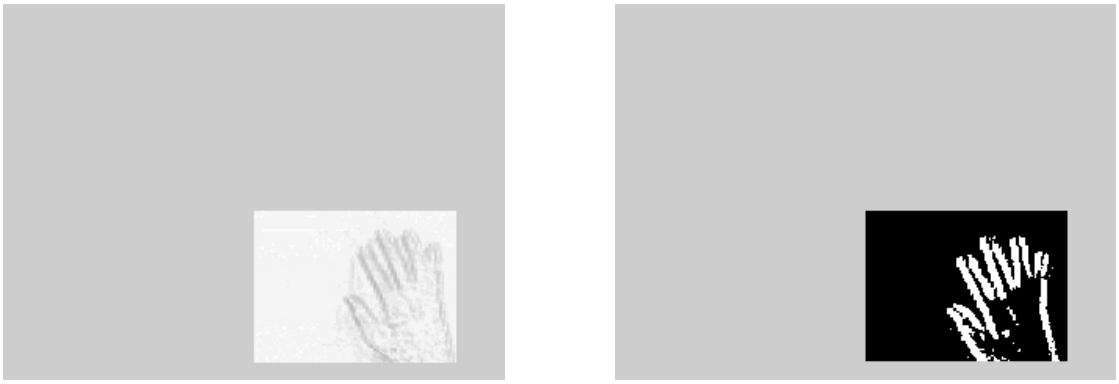
Figure 4.12: Two examples of blurring figure 4.3(b). Gaussian filter kernel sizes 17-by-17 (a) and 21-by-21 (b) have been used. The corresponding edge images are shown in figure 4.13.



(a) (b)

Figure 4.13: Edge detection examples. As it can be seen, some of the relevant edges are eliminated, due to the stronger initial blur.

is very efficient for masking objects that do not move, as they rapidly become part of the background during the adaptation of the background model. This is very suitable for interactive surfaces on tabletops, where other objects than hands are visible to the camera. Moreover, the adaptive background model suppresses the effect of varying backgrounds on interactive surfaces, such as projected images or the image of display device. On the other hand, this effect is critical for event detection, since the segmentation suddenly disappears, if the hand stays still for just a moment. To use slower adaptation of the background model by adjusting the learning rate during the algorithm, the IDS tech-



(a) (b)

Figure 4.14: The result of the self-implemented version of the adaptive background model (a) and the later thresholding (b), as proposed in [LB04]. It can be seen that only the contour of the hand is detected as foreground and that the binary image is noisy. This is inappropriate for the later shape filtering, since the inner side of the hand region does not provide any texture information. Furthermore, if the hand stays still for a moment, the segmentation completely disappears, which makes event detection critical.

nique produces ghost regions, which are inappropriate for further processing as hand shapes are widely distorted.

Median smoothing and morphological operations, such as erode and dilate with a structuring element, are suitable to improve the segmentation results (i.e. remove noise and create a more dense segmentation, respectively). Due to the use of skin detection, shadows do not interfere with the segmentation process, as long as the parameters of the skin detection approach are correctly adapted to the ambient lighting conditions.

The shape filtering algorithm, consisting of the dual template matching approach, primarily depends on the quality of the previous image segmentation. If there are holes in the hand region, the algorithm may produce many FPs. To this end, there is no method to prevent from misclassifications, as there is currently no geometrical hand model included. Here, fingertip detections should further be analyzed, if they are at plausible positions within the hand region. Therefore, the whole hand region needs to be included in the shape filtering and fingers need to be detected as well. Generally, the shape filtering approach is fast and rather straightforward. Nevertheless, it is not very robust, as false positive detections are rather frequent.

CHAPTER 5

Conclusion & Future Work

The last chapter focuses on completing this work by discussing the application of computer vision for multi-touch recognition in tabletop environments. The actual performance of the presented software component and its underlying computer vision approach is once more pointed out and summarized. Main contributions and important final conclusions of this work are provided in the following section 5.1. Further ideas for future work are presented at the end of this chapter in section 5.2.

5.1 Discussion & Final Conclusions

The main contribution of this work is the presentation of a pipeline of OpenCV functions and self-implemented computer vision algorithms for real-time surface tracking and fingertip detection in natural indoor environments. The pipeline includes a geometrical rectangle detection approach and a vertex tracker for detecting and following a high-contrast rectangular surface, even when it is partly overlapped or outside of the camera view, and obtaining the corresponding surface vertices. Furthermore, the pipeline contains an approach for segmenting natural images and extracting hand shapes from the distracting background. A specific shape filtering approach processes those hand shapes in order to detect fingertips that hover above the tracked surface and to provide their two-dimensional on-screen coordinates. Another major contribution of this work is the implementation of an easily extendable, cross-platform openFrameworks add-on for the fast and easy experimentation with computer vision methods and algorithms for real-time multi-touch recognition. The add-on includes the presented pipeline of algorithms as well as a simple and easy-to-understand GUI to support future programmers that are trying to reuse or enhance the add-on. A further contribution is the selection, explanation and comparison of relevant state-of-the-art technologies for multi-touch interaction

in tabletop environments. This includes optical-sensing techniques, a capacitive surface technology and novel vision-based approaches.

In fact, multi-touch interaction is increasingly common in the field of HCI research. In comparison to conventional input devices, multi-touch interfaces provide a basic advantage in usability, due to the direct and intuitive form of manipulation. Commonly used interaction systems based on touch-sensitive hardware offer accurate fingertip detections and robustness against varying lighting conditions. Nevertheless, cost and limitations in scalability and portability are prime drawbacks of such technologies. Vision-based systems are a suitable alternative, since computer vision algorithms have become more and more efficient over the past years and video hardware is widely available. Vision-based approaches generally offer more adjustable interfaces for situations in which low-cost, scalability and portability are demanded. However, one of the main issues in computer vision are varying lighting conditions, together with the adaptation of the algorithms to the specific task. Especially image segmentation is crucial. As expected, the presented system provides similar results to related vision-based attempts that can be found in the literature.

The proposed surface tracking approach generally obtains continuous detection of the interactive surface, even if it is partly overlapped by the hands of the user(s) or outside of the camera view. Here, results mainly depend on the number of lines returned by the Hough transform, the contrast between the surface and the scene background and the geometrical conditions for rectangle detection. Basically, false detections are eliminated by appropriate blurring in the beginning of the algorithm and tight geometrical conditions. The lack of removing duplicate rectangles is a problem but does not affect the approach in the first place. It results in slight flickering of the detected region of interest. On the other hand, occlusions are a major problem, since the detection may fail completely. There is no really suitable method to prevent from occlusions except from using larger interactive surfaces (i.e. A3 and greater). Moreover, lines detected on the hands or other objects in the background may interfere and produce false detections. This effect is reduced by manually defining the desired surface that should be tracked. The vertex tracker is then able to effectively prevent from false positive detections.

The presented approach for fingertip detection widely depends on the initial image segmentation procedure. Skin detection lead to better segmentation than the self-implemented version of the adaptive background model proposed by Letessier and Bérard. The segmented region is much more dense and does usually not contain large holes, which are a major problem for the later shape filtering. Small holes are filled by intermediate median filtering. However, skin detection is of course heavily affected by illumination changes. Visual artifacts, such as bright spots (i.e. reflections), are a major problem here. Moreover, the approach is limited to the Caucasian type of humans.

Altogether, the presented vision-based system cannot be considered robust in natural indoor environments, where illumination is uncontrolled to a large extent. Due to the

many static parameters, the approach is virtually unable to automatically handle varying lighting conditions. Small changes have a major effect on the general detection results, equally affecting both surface tracking and fingertip detection. The included method for manually predefining the global image luminosity level provides a minor improvement but it is simply too rough to fit the continuous range of illumination changes. The system obtains best results if lighting is initially predefined and does not change over time, as is in a shaded indoor environment, for example. Generally, overhead illumination works best, when the interactive surface resides on a tabletop and the camera is mounted overhead as well. Light reflections should generally be avoided.

5.2 Future Work

The presented vision-based system can be improved in several ways. Recommendations concerning the practical setup and optional hardware components, have already been provided in subsection 4.1.1. On the theoretical side, it is very important to find an adequate technique for automatically estimating parameters, preferably directly from the input images. It is commonly known that OpenCV is a powerful instrument for efficient computer vision processing but its algorithms are only as good as their parameters are adapted to the actual purpose and precise situation. In this work, it is essential to adapt thresholds, filter kernel sizes, etc. to the actual lighting conditions in the scene, which usually change over time in a natural indoor environment. Therefore, a major task for future work should be the development of a superior image description approach that evaluates the incoming images from the camera stream and automatically estimates relevant parameters for the specific algorithms. The approach should furthermore be able to control the camera autonomously, so that the frames can be initially prepared (e.g. automatically adapting white balance, aperture size, luminosity level, etc.) before being handed over to the computer vision routines. The appropriate preparation of the camera stream is essential.

The presented surface tracking approach should be enhanced by self-implementing the Hough transform. The matching of the shape characteristics should be performed in the Hough space, which seems to be more accurate and possibly less computational expensive as well, if the implementation is optimized. Furthermore, other than rectangular shapes can be tracked easily by simply adapting the shape characteristics.

Fingertip detection should be improved by including a geometrical finger or even hand model, so that the whole hand movement can be analyzed for more robust fingertip detection and further gesture recognition. Moreover, multiple fingertips could be assigned to different hands and users. For the foreground extraction process, a combination of the adaptive background model and an enhanced skin detection technique should be implemented to improve segmentation and to adapt the approach to various situations (i.e. different skin color, other objects with similar color).

Regarding touch detection, one of the proposed techniques that have been described in subsection 4.3.4 should be implemented. It depends on further experimentation which technique is the most suitable for the approach.

For fast real-time processing, CUDA should be considered when revising existing algorithms or implementing further techniques. CUDA optimized program code should bring a relevant speed-up in applications, where many geometrical calculations need to be performed. This is especially relevant for the geometrical rectangle detection approach and the implementation of the Hough transform.

Bibliography

[AICB07] Ankur Agarwal, Shahram Izadi, Manmohan Chandraker, and Andrew Blake. High Precision Multi-Touch Sensing on Surfaces using Overhead Cameras. In *TABLETOP '07: Second Annual IEEE International Workshop on Human-Computer Systems*, pages 197–200, 2007.

[And10] Li Andol. Andol Research. Website, 2010. Available at `http://www.andol.info/research`. Visited on October 1st, 2010.

[App10a] Apple.com. iPad. Website, 2010. Available at `http://www.apple.com/ipad`. Visited on August 12th, 2010.

[App10b] Apple.com. Magic Trackpad. Website, 2010. Available at `http://www.apple.com/magictrackpad`. Visited on August 12th, 2010.

[ATD01] Alberto Albiol, Luis Torres, and Edward J. Delp. Optimum Color Spaces for Skin Detection. In *ICIP '01: Proceedings of the 2001 International Conference on Image Processing*, pages 122–124, 2001.

[BK08] Gary R. Bradski and Adrian Kaehler. *Learning OpenCV, 1st Edition*. O'Reilly Media, Inc., 2008.

[Blo08] Ryan Block. Macbook Air Review. Website, 2008. Available at `http://www.engadget.com/2008/01/25/macbook-air-review`. Visited on August 12th, 2010.

[Bux09] Bill Buxton. Multi-Touch Systems that I Have Known and Loved. Website, 2009. Available at `http://www.billbuxton.com/multitouchOverview.html`. Visited on October 6th, 2010.

[Cir10] Circletwelve.com. DiamondTouch. Website, 2010. Available at `http://www.circletwelve.com/products/diamondtouch.html`. Visited on August 12th, 2010.

[DL01] Paul Dietz and Darren Leigh. DiamondTouch - A Multi-User Touch Technology. In *UIST '01: Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*, pages 219–226, 2001.

[Dro10] Hervé Drolon. FreeImage. Website, 2010. Available at `http://freeimage.sourceforge.net`. Visited on October 12th, 2010.

[Ges10] Gesturetek.com. Illuminate Multi-Touch Table Display Screen and Multi-Touch Surface Computer Technology. Website, 2010. Available at `http://www.gesturetek.com/illuminate/productsolutions_illuminatetable.php`. Visited on August 12th, 2010.

[Gre08] Kate Greene. A Low-Cost Multitouch Screen. Website, 2008. Available at `http://www.technologyreview.com/Infotech/20827/?a=f`. Visited on August 12th, 2010.

[Han05] Jefferson Y. Han. Low-Cost Multi-Touch Sensing through Frustrated Total Internal Reflection. In *UIST '05: Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology*, pages 115–118, 2005.

[Han06] Jefferson Y. Han. Jeff Han. Website, 2006. Available at `http://cs.nyu.edu/~jhan`. Visited on August 12th, 2010.

[Hon07] Mathew Honan. Apple Unveils iPhone. Website, 2007. Available at `http://www.macworld.com/article/54769/2007/01/iphone.html`. Visited on August 12th, 2010.

[HPM03] Mark Hills, Tony Pridmore, and Steven Mills. Object Tracking through a Hough Space. In *VIE '03: Proceedings of the 2003 International Conference on Visual Information Engineering*, pages 113–120, 2003.

[JS04] Claudio R. Jung and Rodrigo Schramm. Rectangle Detection based on a Windowed Hough Transform. In *SIBGRAPI '04: Proceedings of the 17th Brazilian Symposium on Computer Graphics and Image Processing*, pages 113–120, 2004.

[LA97] Dmitry Lagunovsky and Sergey Ablameyko. Fast Line and Rectangle Detection by Clustering and Grouping. In *CAIP '97: Proceedings of the 7th International Conference on Computer Analysis of Images and Patterns*, pages 503–510, 1997.

[LB04] Julien Letessier and François Bérard. Visual Tracking of Bare Fingers for Interactive Surfaces. In *UIST '04: Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*, pages 119–122, 2004.

[Lem10] Werner Lemberg. FreeType. Website, 2010. Available at `http://www.freetype.org/index2.html`. Visited on October 12th, 2010.

[Mer10] Merl.com. DiamondTouch. Website, 2010. Available at `http://www.merl.com/projects/DiamondTouch`. Visited on August 12th, 2010.

[Mic10] Microsoft.com. Microsoft Surface. Website, 2010. Available at `http://www.microsoft.com/surface/en/us/default.aspx`. Visited on August 12th, 2010.

[Nui10a] Nuigroup.com. Natural User Iinterface Group. Website, 2010. Available at `http://nuigroup.com/go/lite/about`. Visited on August 12th, 2010.

[Nui10b] Nuigroup.com. NUI Group Community Book – Multi-Touch Technologies. Website, 2010. Available at `http://nuigroup.com/log/nuigroup_book_1`. Visited on August 12th, 2010.

[Nui10c] Nuigroup.com. NUI Group Community Wiki. Website, 2010. Available at `http://wiki.nuigroup.com/Main_Page`. Visited on August 12th, 2010.

[Nvi10] Nvidia.com. CUDA. Website, 2010. Available at `http://www.nvidia.com/object/cuda_home_new.html`. Visited on September 9th, 2010.

[Ope10a] Opencv.willowgarage.com. OpenCV. Website, 2010. Available at `http://opencv.willowgarage.com/wiki/Welcome`. Visited on September 9th, 2010.

[Ope10b] Openframeworks.cc. OpenFrameworks. Website, 2010. Available at `http://www.openframeworks.cc`. Visited on September 9th, 2010.

[Ope10c] Opengl.org. OpenGL. Website, 2010. Available at `http://www.opengl.org`. Visited on October 12th, 2010.

[Sca10] Gray P. Scavone. RtAudio. Website, 2010. Available at `http://www.music.mcgill.ca/~gary/rtaudio`. Visited on October 12th, 2010.

[Ted06] Ted.com. Jeff Han Demos His Breakthrough Touch-Screen. Website, 2006. Available at `http://www.ted.com/talks/lang/eng/jeff_han_demos_his_breakthrough_touchscreen.html`. Visited on October 6th, 2010.

[WB94] Colin Ware and Ravin Balakrishnan. Reaching for Objects in VR Displays: Lag and Frame Rate. *TOCHI '94: ACM Transactions on Computer-Human Interaction*, 1(4):331–356, 1994.

[Wet10] Wetab.mobi. WeTab. Website, 2010. Available at `http://wetab.mobi`. Visited on August 12th, 2010.

[Wik10a] Wikipedia.org. Accuracy and Precision. Website, 2010. Available at `http://en.wikipedia.org/wiki/Accuracy_and_precision`. Visited on August 10th, 2010.

[Wik10b] Wikipedia.org. Capacitive Coupling. Website, 2010. Available at `http://en.wikipedia.org/wiki/Capacitive_coupling`. Visited on August 26th, 2010.

[Wik10c] Wikipedia.org. Human-Computer Interaction. Website, 2010. Available at `http://en.wikipedia.org/wiki/Human-computer_interaction`. Visited on August 10th, 2010.

[Wik10d] Wikipedia.org. Input Device. Website, 2010. Available at `http://en.wikipedia.org/wiki/Input_device`. Visited on August 10th, 2010.

[Wik10e] Wikipedia.org. Latency (Engineering). Website, 2010. Available at `http://en.wikipedia.org/wiki/Latency_(engineering)`. Visited on August 10th, 2010.

[Wik10f] Wikipedia.org. Refractive Index. Website, 2010. Available at `http://en.wikipedia.org/wiki/Refractive_index`. Visited on August 19th, 2010.

[Wik10g] Wikipedia.org. Robustness (Computer Science). Website, 2010. Available at `http://en.wikipedia.org/wiki/Robustness_(computer_science)`. Visited on August 10th, 2010.

[Wik10h] Wikipedia.org. Time-Of-Flight Camera. Website, 2010. Available at `http://en.wikipedia.org/wiki/Time-of-flight_camera`. Visited on September 10th, 2010.

[Wik10i] Wikipedia.org. Usability. Website, 2010. Available at `http://en.wikipedia.org/wiki/Usability`. Visited on August 10th, 2010.

[Wil05] Andrew D. Wilson. PlayAnywhere: A Compact Interactive Tabletop Projection-Vision System. In *UIST '05: Proceedings of the 18th Annual*

ACM Symposium on User Interface Software and Technology, pages 83–92, 2005.

[Wil10] Tracy V. Wilson. How the iPhone Works. Website, 2010. Available at `http://electronics.howstuffworks.com/iphone.htm`. Visited on August 26th, 2010.