# An Extensible Architecture for Multitouch & Pen Interactive Tables

Erwin Aitenbichler
Telecooperation Lab
Darmstadt University of Technology
Hochschulstrasse 10
64289 Darmstadt, Germany
erwin@informatik.tu-darmstadt.de

Dirk Schnelle-Walka
Telecooperation Lab
Darmstadt University of Technology
Hochschulstrasse 10
64289 Darmstadt, Germany
dirk@tk.informatik.tu-darmstadt.de

## ABSTRACT
Augmented multitouch interaction provides a better user experience, but requires the integration of other input sources, e.g., a digital pen, each featuring an own API. Our approach to a multitouch table is realized as a service-oriented architecture combining these different aspects under the hood of a common API. Their use make our table work with a variety of input devices and offer the opportunity of extending the capabilities by adding new services.

## Keywords
Multitouch, Anoto, Object detection

## INTRODUCTION
Gesture based interfaces have been around for years. Starting from light pens that appeared in the late 1950s up to Tablet PCs that can be bought off-the-shelf, today. Unfortunately, most of these devices, including the newer ones, restrict the user to poking on standard interfaces of windows, icons, menus and pointers. These techniques are still far from a seamless interaction between a human and a computer. The vision, as shown in the famous movie "The Minority Report" in 2002 is still not reached for the public. There are some first approaches like the iPhone, but most of what is done, especially for larger displays, happens at research institutes.

In this paper we introduce our approach towards this vision. One of the goals of the development was to build a table that is suitable to develop concepts for such a seamless interaction. But there are also other interaction techniques accompanying multitouch interaction. One of them is the use of pen and paper to avoid media discontinuity, e.g., when taking notes. Hence, note taking should also be usable on a table. A suitable tool for writing and annotating digitally is the Anoto Pen [2]. It is designed to work on physical documents but offers a high potential to be also used with digital documents. This is the goal of several paper toolkits like CoScribe [10] that combine the interfaces pen, paper, and PC. CoScribe is developed at our research institute enabling us to benefit from this framework directly. Its strength is the unified interaction on paper and screens, thereby providing equal support for physical and digital documents.

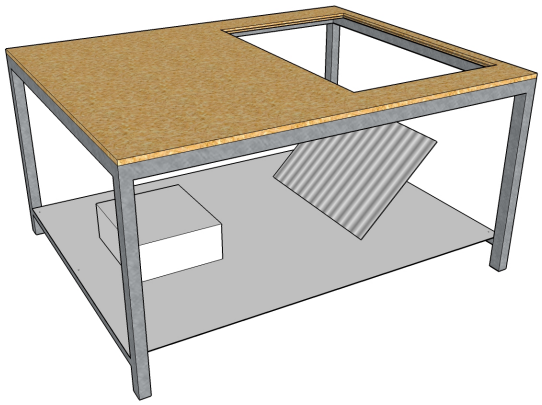

**Figure 1: Schematic overview of the table construction**

In the following sections, we describe an extensible and flexible architecture for touch and pen interactive tables. We show three different scenarios which we can support with our own image processing software: touch detection (finger and palm), pen detection, and marker detection. As application scenarios we describe an Annotated Photo interactive demo and the integration of DiamondTouch. The latter is supported by a TUIO [5] bridge translating our custom protocol to the TUIO protocol. This way, a large amount of existing applications relying on this de-facto standard can be run on our table.

The variety of the demos show that our table can be used with different input devices which are easy to integrate because of the extensible architectural design of our software.

## TABLE CONSTRUCTION
The table that we constructed is a rear projection table as shown in Figure 1. A Full HD data projector is used for displaying the image from the PC, via a mirror. For this application, a thin foil mirror has to be used, because conven-

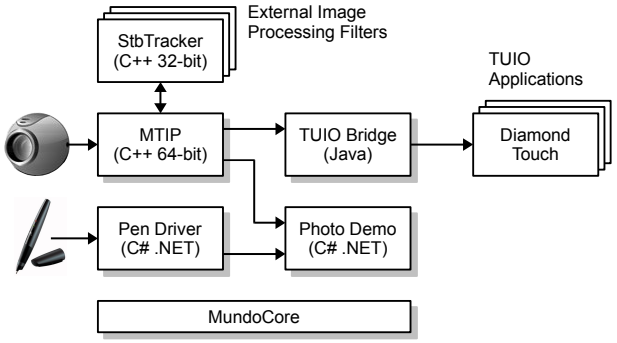


**Figure 2: Software Architecture**

tional mirrors produce duplicated ghost images. The image is mirrored to the display surface of the table. The display has a size of 100cm x 56cm with a 16:9 ratio and Full HD resolution of 1920x1080 pixels. The finger touches are recorded by a PointGrey infrared (IR) camera which is mounted close to the bottom right corner of the mirror. This camera position was chosen to avoid the disturbing reflections caused by the projector's lens. The camera itself is equipped with an IR pass filter and a wide-angle lens. The wide-angle lens allows for a short distance to the display surface. The software to process the images is explained in the sections below.

The display surface consists of four layers. Starting from the top, there are the following layers, each having a special purpose:

1. Acryl glass protection layer (1mm)
2. Transparent foil with Anoto pattern
3. Back projection foil
4. Acryl glass base plate (5mm)

The first acryl glass layer shields the underlying Anoto foil from finger touches and ink writings with the Anoto pen. It allows for a more comfortable interaction when dragging with the fingers for a longer time. The Anoto foil has a printed Anoto streaming pattern that is needed to detect the pen strokes. The third foil displays the image coming from the projector. The acryl glass at the bottom has a stabilization function.

The following section gives insight how this setting is used for multitouch and other interaction techniques.

# ARCHITECTURE

Figure 2 shows the architecture of our interactive table software. The system is implemented as a set of loosely-coupled services that are implemented using different programming languages. All communication between components is performed via MundoCore [1].

MundoCore is a communication middleware that was designed for Ubiquitous Computing to integrate software components in a heterogeneous environment. It provides a common set of APIs for different programming languages (e.g., Java, C++, Python, C#.NET) on a wide range of different devices. The architectural model addresses the need for proper language bindings, different communication abstractions (publish/subscribe, distributed object computing, and streaming), peer-to-peer overlays, different transport protocols, different invocation protocols, and automatic peer discovery. MundoCore is open source and available from the homepage of our lab under the Mozilla Public License.

MundoCore builds on *publish/subscribe* as basic abstraction. This scheme provides channels, which interconnect output ports (publishers) of services with input ports (subscribers) of other services. Channels are a suitable means to transport messages, such as touch events, pen events, or video images. MundoCore is well-optimized and facilities low-latency and high-bandwidth communication. Over Gigabit Ethernet, the transmission of an event between two services takes 0.5ms and video streams can be transmitted with up to 600 MBit/sec. In comparison to that, the latency caused by camera framerate limitations or by the Bluetooth connection of an Anoto pen will always be higher.

For our interactive table system, MundoCore gives us great freedom and allows us to choose the language and runtime system best suited for the implementation of a specific service or application. The demo scenario described in this paper uses four different kinds of processes: 32-bit C++, 64-bit C++, C#.NET, and Java.

The Mundo Touch Image Processor (MTIP) as the core service in this architecture processes the camera images. The camera is mounted inside the table and records the touch surface from below. MTIP implements all image processing steps and generates touch events that are published to a MundoCore channel. MTIP is implemented in C++ for optimal performance. Because the computer hosting MTIP runs 64-bit Windows and the PointGrey API is used to acquire images from the camera, MTIP has to be a 64-bit application as well.

Most image filters are directly integrated into MTIP. However, it is also possible to implement filters as external MundoCore services. In this case, MTIP sends a video stream to the external service and receives an output video stream or vector data in return. An example for this is the integration of the used marker tracking library, we use the (commercial) Studierstube [7] Tracker library. Currently, we only have a 32-bit version of this library, what makes it impossible to integrate it directly into MTIP. Hence, we decided to implement marker tracking as an external filter service.

The TUIO bridge receives touch events from MTIP and translates them to TUIO messages. Hence, arbitrary multitouch applications can also be used on our table.

To access the Anoto pen, we use the (commercial) Interactive Surface Development Kit [6]. Anoto pens have built-in cameras to read a special pattern printed on the table's surface and send coordinate information to a computer via Bluetooth. Since the SDK is provided as a .NET assembly, the *Pen Driver* service was implemented in C#. The service records pen events and publishes them to a MundoCore channel.

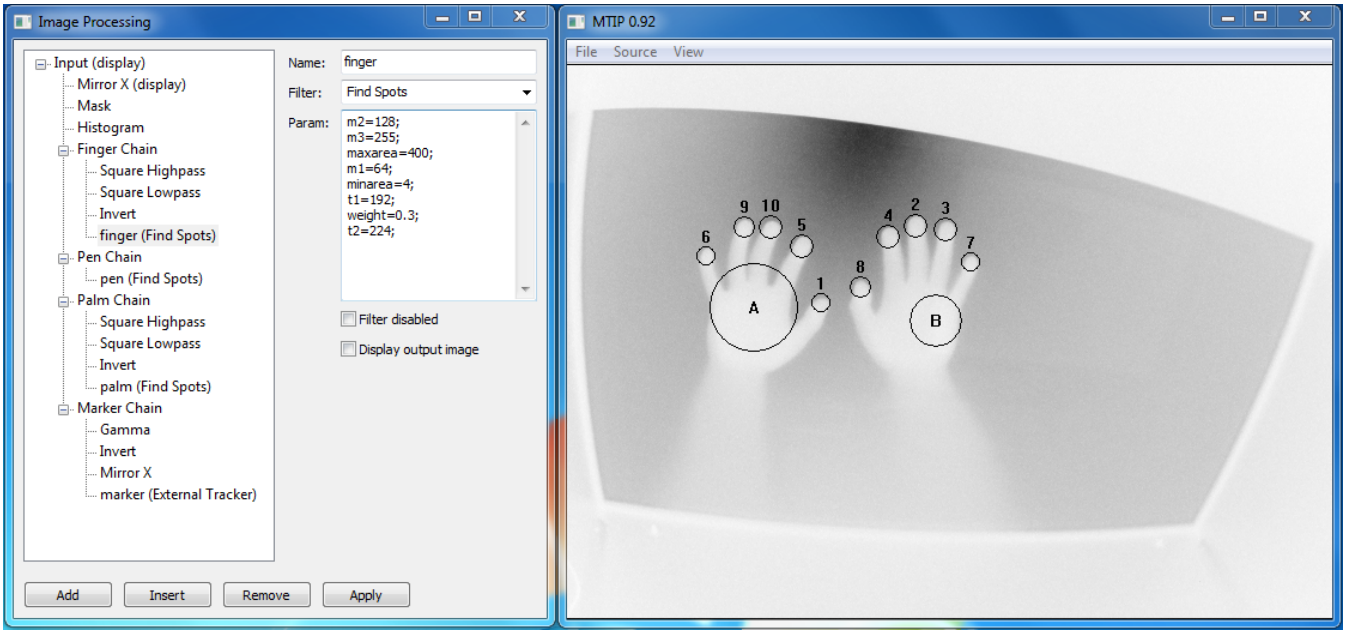


**Figure 3: Screenshot of the Mundo Touch Image Processor (MTIP)**

The Photo Demo which is described later on shows the combined use of Anoto events and MTIP events.

## MTIP

The Mundo Touch Image Processor (MTIP) is a customizable application that allows to detect various touch events based on image processing. In general, MTIP performs the following processing steps:

1. The image is acquired from the camera.
2. The image is processed in one or multiple filter chains. Each chain ends with a component that extracts features from the image, such as geometric coordinates, or ID information.
3. In the geometry processing step, inter-frame and inter-chain aspects are handled. For example, a detected contact is labeled with an ID that remains constant until it is released. It is also possible to combine the results of multiple image processing chains, e.g, masking of nearby finger contacts if a pen contact is detected.
4. Event notifications are generated and sent to the applications using MundoCore channels.

## Image Processing

Image processing is based on a filter graph. The use of only unary operations make it possible to display this graph as a tree. Image processing starts with a set of common preprocessing filters that are needed by all following operations. Then the data flow splits up into several dedicated filter chains. In the following, we describe the recognition of finger touches, palm touches, pen touches, and marker detection. The expanded filter tree is shown in Figure 3.

### *Preprocessing*

The head of the filter list contains three filters that are responsible for the common preprocessing steps.

| Filter | Parameters |
|---|---|
| Mirror X | |
| Mask | (mask polygon) |
| Histogram | |

First, the image is mirrored horizontally, because the recording camera is mounted inside the table. Alternatively, the mirror operation could automatically be handled later on by a geometric operation, based on the calibration data, but displaying images when, e.g., calibrating the table is more intuitive to the user when it is not left-right reversed. Next, the mask filter removes regions around the surface that might contain disturbing spots.

The histogram filter calculates a histogram of the image as an input for the *auto exposure* function of MTIP. Most cameras have built-in auto exposure functions, however, these often do not perform well in tracking applications. The auto exposure function of MTIP provides better results mainly because of two reasons. First, the histogram is calculated after cropping the insignificant regions of the image. Hence, regions close to the interactive surface are not taken into account for auto exposure. Second, MTIP ignores small saturated white spots in the image without changing the camera's exposure parameter. Such saturated spots can be caused by IR emitting pens on the surface or by reflecting markers, resulting in a degradation of the finger detection. In such cases, the camera exposure must not be changed.

### *Finger processing*

The following filter chain was designed for shadow tracking to detect finger contacts on the surface. These contact points

will be visible as dark spots.

| Filter | Parameters |
|---|---|
| Highpass | size=6, amp=10 |
| Lowpass | size=2 |
| Invert | |
| FindSpots | minarea=4, maxarea=400, weight=0.3 |

First, we apply a highpass filter to eliminate the uneven lighting of the surface and to extract spots with the size of a finger. The size parameter of the filter is set such that only small spots, as they are caused by a finger, pass through the filter. Since the contrast of fingers on the surface is only a few levels of gray, the filter multiplies the result with an amplification factor.

As a side-effect, the highpass filter also amplifies noise caused, e.g., by the analog parts of the camera. To remove high frequency noise, a lowpass filter is applied next. The filter's size paramter should be as small as possible, but it should be big enough to suppress the detection of contacts that would be recognized because of noise.

Because shadow tracking is used, the image is inverted next. Finally, a FindSpots filter is used to extract the center coordinates of all bright spots. Important parameters of this filter are minarea, maxarea, and weight. minarea specifies the minimum permitted area of the bright spot in pixels. This margin determines the sensitivity of the filter, but also its susceptibility to noise. maxarea specifies the maximum permitted area of the spot in pixels. weight defines how "full" or "hollow" a spot may be. It allows to mask, e.g., the effects caused by the edges of objects lying on the surface.

It is noteworthy that we do not use a "classical" background removal step. Such a background removal filter records a reference image and then subtracts the reference image from the acquired camera image. Our tests with this filter, which is also supported by MTIP, proved that such a filter has a bad performance. The question is when such a reference image should be taken. Clearly, the answer would be: when the surface is not touched. However, this cannot be easily detected at this processing stage. Also, if the user remains touching the same position for an extended period of time, or a marker stays on the surface, this cannot be done. Another possible solution would be to record every other frame without IR illumination to get the reference image. However, this does not help for shadow tracking, reduces the effective frame rate, and makes the table setup more complicated.

If a Frustrated Total Internal Reflection (FTIR) setup or if an infrared backlight is used, then finger contacts will be visible as bright areas. In this case, the same considerations as above apply with only little effects on the setup of the filter chain. The Invert filter has to be disabled or removed and the parameters of the highpass filter can be tuned for optimal results.

### *Palm processing*

The following filter chain was designed to recognize the hand or palm.

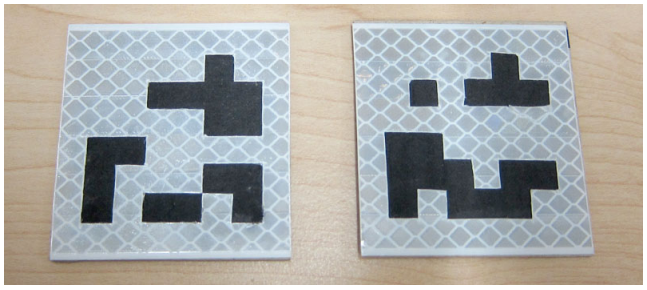

**Figure 4: Reflective ID Markers**

| Filter | Parameters |
|---|---|
| Highpass | size=40, amp=10 |
| Lowpass | size=20 |
| Invert | |
| FindSpots | minarea=100, maxarea=10000, weight=0.3 |

This filter chain also uses a highpass filter. This filter again removes the uneven lighting of the surface, which has a very low frequency. The size parameter has a relatively large value and therefore allows to pass the spots caused by a hand or palm. After that, the lowpass filter blurs individual fingers. Again, the image is inverted next because of shadow tracking. Similar to the finger processing chain, a FindSpots filter is used to extract the center coordinate of the bright spots.

### *Marker processing*

We use the Studierstube Tracker [7] for marker tracking. For AR applications such markers are typically printed on paper. In our case we want to be able to recognize markers *through* the table surface containing a back-projection foil, hence we have constructed our own markers.

The black regions of a marker are realized by 3M reflective foil reflecting IR light coming from under the table. The white parts of the marker are implemented by cutting out the corresponding parts from the reflective foil. The foil is then sticked to a dark surface. Finally, the marker can be attached to a physical object that should be tracked on the table. Two such markers are shown in Figure 4.

| Filter | Parameters |
|---|---|
| Gamma | low=128, high=255, gamma=1.0 |
| Invert | |
| Mirror X | |
| ExtFilter | channel=stbracker |

For image processing, the contrast between the black and white levels is increased using a gamma filter that reduces the input range. Next, the image is inverted and mirrored back to its original orientation. The resulting video stream is then published to the channel stbtracker.image and the filter receives marker information on stbtracker.coord. The actual marker detection is performed in an external MundoCore service.

## Spot Detection

The field of mathematical morphology [9] provides the theoretical foundations and building blocks for the necessary

image processing operations. The FindSpots filter performs the following steps to determine the coordinates of bright spots in the image:

1. First, the grayscale input image is converted to a binary image with a hysteresis threshold operator. This method proved to be quite robust against the sampling noise of the camera.
2. Next, the algebraic area opening is calculated and subtracted from the image. This operation takes a pixel count as parameter and eliminates all bright areas larger than this threshold.
3. Finally, the center coordinates of each bright spot are determined. This is achieved by calculating the mean value of all bright pixel coordinates of a spot. Consequently, the resulting pixel coordinates have sub-pixel accuracy.

## Vector Processing

After the feature extraction from the image, the features are passed to the *tracker* components in the vector processing stage. The FindSpots filters in the different chains only determine the center locations of bright spots on a per-frame basis. Hence, they do not provide enough information to distinguish if a finger was moved, a new finger contact appeared or disappeared. Therefore, the tracker components also consider inter-frame aspects. If a spot is reported for the first time, then the tracker assigns a new ID to the spot and generates a *down* event. If a spot is no longer reported, then the tracker generates an *up* event. The tracker analyzes each two consecutive frames and tracks all spots on a maximum likelihood basis. Hence, IDs stay assigned with the spots.

Trackers use a FindSpots filter as their main source of information. Optionally, a second FindSpots filter can be specified in the exclude parameter. The exclude option allows to remove spots from the main source. This is useful, e.g., if a disturbing effect is detected with another FindSpots filter in parallel. This way, falsely detected contacts can be removed in this stage.

Finally, the tracker converts the coordinates to normalized screen coordinates. This is necessary to take the properties of the camera and its placement into account. The necessary transformation parameters are determined during calibration which is described next.

## Calibration

MTIP supports the calibration process in its user interface. The application displays a point grid and the user has to touch one point after the other. This allows the application to know the relationship between several points in the camera's coordinate system and the points in the projector's coordinate system. MTIP uses these calibration points together with 2D spline interpolation to transform between coordinate systems.

The calibration procedure can also be done with the Anoto pen. In this case, for each contact, three pieces of information can be recorded: The location of the spot in screen

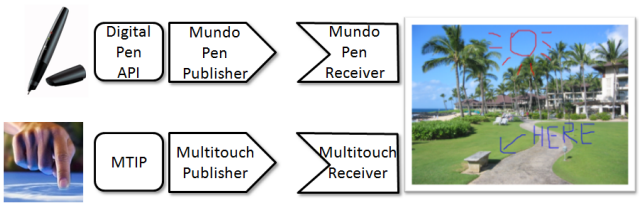


**Figure 5: Architecture of the annotated photo demo**

(projector) coordinates, the position of the contact in the camera's coordinate system, and the position of the pen in Anoto pattern coordinates. This also allows to make the calibration process more precise in general, because the pen can be placed more accurately than a finger. Moreover, the pen also provides a force parameter which is exploited to explicitly confirm a calibration point by pressing the pen down firmly.

## PHOTO DEMO

As a proof of concept we implemented a demo that shows the potential of our approach. Imagine a group of users viewing photos. They are standing around the multitouch table where some photos are displayed on the screen. The users can modify these photos at their finger tips by moving them around and resizing them. Additionally, they can take a digital pen to annotate the photos. They can do this as they would do it with physical objects. The pen can draw with different colors. The color is changed by simply clicking on a colored area at the top left corner of the screen.

The architecture of the demo is shown in Figure 5. The Photo Demo was implemented in C# using WPF. We chose .NET for this application, because WPF provides a fast and powerful framework for creating user interfaces. The Pen Driver acquires data from the digital pens and publishes the resulting stroke streams to a MundoCore channel. Touch events are detected by MTIP, which publishes these events to another MundoCore channel. The annotated photo demo receives both event streams and integrates them into the user interface.

The demo shows the ease of extension to incorporate other input sources using the same API by means of the underlying Mundo middleware.

## TUIO BRIDGE

TUIO [5] defines a protocol on the basis of OSC and an API for tangible multitouch surfaces that has become a well-known de-facto standard. Echtler states in [4] that "although TUIO has been designed with tracking of tangible objects in mind, it has become a de-facto standard for multitouch data". It features several implementations in different programming languages and is used in a variety of projects. The main goal of these projects is to support higher-level interaction. Most of them are based on DiamondTouch [3]. A prominent example is the DiamondSpin Framework [8]. DiamondSpin is a toolkit written in the Java programming language. It allows the continuous rotation of windows and controlling the desktop using multiple touch points. If this protocol is supported, all the applications that are developed using this technology will be usable.

As mentioned above, we use MundoCore for communication means. In order to make the TUIO protocol usable, we implemented a bridge in Java to fill this gap. The bridge subscribes to the corresponding MundoCore channel and consumes multitouch events. It is also a fully implemented TUIO server, that sends TUIO messages. We follow the version 1.1 of the specification which is the most recent one. The bridge service was implemented in Java, because this is the "default programming language" in our lab. In contrast to the events that are sent over MundoCore, the TUIO objects contain more information such as a session identifier, motion acceleration and rotation acceleration of the object or cursor and a time stamp. MTIP delivers a continuous flow of the following information: id of the touch point, the x and y coordinates of the touch point, the size of the touch point, and the state. The state can be one of down, up and move. The bridge has to take care that these missing information pieces are filled correctly. Therefore, the bridge has to maintain a cache of the received points to determine, e.g., the acceleration data. Additionally, the bridge also reduces the flow of the information flooding from MTIP by sending only messages, if there were changes in the data.

The bridge also shows the advantages of the channel-based information concept that is inherent to the Mundo middleware. It is very easy to add post-processing components that allow the use of higher level context and derived data, thus allowing for a high flexibility.

## SUMMARY

In this paper we described our approach of a multitouch table. The table can be used by multiple users at the same time. It is not restricted to touch interaction only, but can also be used, e.g., with an Anoto pen. Ongoing work exploits the use of other tangibles and their identification with markers. The underlying image processing software features flexible filter chains that enable us to support multiple input sources and different tracking setups. For instance, it is possible to use infrared back-illumination, FTIR, as well as shadow tracking.

The software is developed at our institute integrating some off-the-shelf libraries. It utilizes the MundoCore middleware that allows for a high flexibility. The use of MundoCore allows for a real service-oriented architecture where it is easy to make use of these libraries or extend the capabilities of our table by adding new services.

## VIDEO DEMONSTRATION

A video demonstration of our system can be accessed under the following URL: `http://atlas.tk.informatik.tu-darmstadt.de/Users/erwin/eics2010.html`

## ABOUT THE AUTHORS

**Dr. Erwin Aitenbichler** received his PhD in Computer Science from Darmstadt University of Technology, Germany. Currently he is a post-doctoral researcher in the Telecooperation Lab at Darmstadt University and head of the area Smart Environments. Erwin has done several years of research in the area of local positioning systems and 3D optical tracking.

**Dr. Dirk Schnelle-Walka** received his PhD in Computer Science from Darmstadt University of Technology, Germany. Currently he is a post-doctoral researcher in the Telecooperation Lab at Darmstadt University and head of the area Talk'n'Touch Interaction. Dirk has a strong background in speech and multimodal interaction.

## REFERENCES


[1] E. Aitenbichler, J. Kangasharju, and M. Mühlhäuser. MundoCore: A Light-weight Infrastructure for Pervasive Computing. *Pervasive and Mobile Computing*, pages 332–361, 2007.

[2] Anoto technology. `http://www.anoto.com`, 2010. last accessed on 13.03.2010.

[3] P. Dietz and D. 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, New York, NY, USA, 2001. ACM.

[4] F. Echtler and G. Klinker. A multitouch software architecture. In *NordiCHI '08: Proceedings of the 5th Nordic conference on Human-computer interaction*, pages 463–466, New York, NY, USA, 2008. ACM.

[5] M. Kaltenbrunner, T. Bovermann, R. Bencina, and E. Costanza. TUIO - a protocol for table based tangible user interfaces. In *Proceedings of the 6th International Workshop on Gesture in Human-Computer Interaction and Simulation (GW 2005)*, Vannes, France, 2005.

[6] M. I. Lab. Interactive Surface Development Kit. `http://mi-lab.org/products/interactive-surface-kit/`, 2010. last accessed on 15.03.2010.

[7] T. Langlotz. Studierstube Tracker. `http://studierstube.icg.tu-graz.ac.at/handheld_ar/stbtracker.php`, 2010. last accessed on 15.03.2010.

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

[9] P. Soille. *Morphologische Bildverarbeitung*. Springer, Berlin, 1998.

[10] J. Steimle, O. Brdiczka, and M. Mühlhäuser. CoScribe: Integrating Paper and Digital Documents for Collaborative Learning. *IEEE Transactions on Learning Technologies*, 2(3):174–188, 2009.