– WHITE PAPER –

Integrating a Depth Camera in a Tabletop Setup for Gestural Input On and Above the Surface

Nadia Haubner, *Johannes Luderschmidt*, *Simon Lehmann*, *Ulrich Schwanecke*, *Ralf Dörner*
RheinMain University of Applied Sciences
Unter den Eichen 5, 65195 Wiesbaden, Germany
{nadia.haubner, johannes.luderschmidt, simon.lehmann, ulrich.schwanecke, ralf.doerner}@hs-rm.de

We present a system for the detection of gestural input above a tabletop environment employing a depth sensing camera that is mounted over the surface. We discuss the challenges when combining both systems and offer according solutions. We employ computer vision methods to actually detect gestural input in the camera's depth images. Our detection currently works with two depth sensing techniques: low-cost structured light and high-end time-of-flight. We have built a combined setup by mounting a depth camera above an existing interactive tabletop system based on infrared (IR) light illumination. A depth camera that employs IR illumination interferes with tabletop systems that also use IR illumination. One result is that only the structured light based depth camera works smoothly with our tabletop system. It has the potential to enhance other tabletop setups based on IR illumination techniques. Our system complements tangible interaction on tabletops with gestural input above and around the surface. The presented solution paves the way for new hybrid gestural interaction techniques. We exemplify a few such techniques like hybrid hovering, proxemic interaction or multi-user interaction.

1 INTRODUCTION

With the advent of the depth camera Microsoft® Kinect[1], depth sensing technology based on infrared (IR) illumination has become widely available at low prices. Since a range of novel interaction techniques will be possible, it is a well invested effort to deal with the question how such a camera can be employed. As there are many professional and home-made tabletop systems based on low-cost IR light illumination like frustrated total internal reflection (FTIR) or diffused illumination (DI), it is also of interest if and how both low-cost systems can be combined and which kind of new interaction techniques this combination offers.

So far, in the field of human computer interaction, depth cameras have amongst others been employed to recognize gestural user input in games and in the field of interactive displays. Games like those for the Kinect system use the whole body of a player as an input device. For the interaction with displays, users can, for instance, control applications with gestures like in [5]. For such applications, the user faces the camera (front-view) to detect input of hands, arms and of the whole body. Depth cameras are also applicable in tabletop environments where they have for instance been used to create height maps of objects on the surface [10, 14] or to track touches [15]. Contrary to front-view setups, most of the existing tabletop approaches mount the depth camera above the tabletop facing the surface (top-view). In a top-view setup, a camera can monitor the whole tabletop environment and users dispersed around the table do not occlude each other.

However, the existing tabletop environments with integrated top-view depth camera only allow for a small range of gestural input. To detect gestural input in a top-view setup, a different algorithmic approach is necessary as compared to a front-view setup to segment, detect and track arms above the surface. Additionally, so far IR illumination depth cameras have not been combined in a top-view setup with tabletop environments that are based on IR illumination techniques like FTIR or DI. Therefore, the integration of such a camera into tabletop environments poses a technical challenge since interference may lead to undesirable effects.

The contribution of this paper is the discussion of all major issues when a depth-camera is combined with a tabletop environment: Firstly, we propose a computer vision approach to segment, detect and track gestural input with top-view depth cameras. Secondly, this paper presents a top-view setup of a depth camera with a table-

[1] http://www.xbox.com/kinect

top environment based on DI. For the purpose of comparison with the low-cost Kinect, we have also tested the high-end time-of-flight [9] depth camera PMD[vision]® CamCube 2.0 [2], which turns out to be inferior in comparison to the Kinect in our setup. Thirdly, we explain technical challenges and appropriate solutions when a depth camera is integrated in an IR illuminated tabletop environment. Fourthly, this paper gives examples for interaction techniques in such a combined setup.

1.1 Outline

We give a brief overview of relevant literature in section *Related Work*. Section *Setup* describes our tabletop environment that integrates the depth camera. Section *Detecting User Input* describes how we detect gestural input on and above the tabletop. Afterwards, section *Integration* discusses problems and solutions related to the IR illumination of both systems. Additionally, it illustrates how the setup can be calibrated and describes the communication of the tracking systems with applications. Before the final section *Conclusion and Future Work*, section *Interaction* exemplifies how tracked body parts in combination with tangible interaction can be employed in a tabletop environment.

2 RELATED WORK

In [6], Hilliges et al present two different rear projection-vision tabletop setups to detect interaction above the surface: Firstly, they detect hands from inside of the table with a standard camera above the switchable diffuser from SecondLight [7]. The height of hands is approximated by their brightness in the image. Secondly, they use a depth sensing camera from behind a holographic screen that forms the surface of the tabletop. The image on the screen is rear-projected and the depth camera can monitor through the screen. Therefore, the camera can detect the exact height of hands behind the surface. Additionally, Hilliges et al consider various forms of interaction in the air and use shadow rendering to give a feedback about hand and arm position. Both setups have the drawback that they require special hardware: For the first setup the rather small SecondLight screen is mandatory and for the second setup a holographic screen is necessary. It offers low display quality that depends heavily on the viewing angle. The first setup can only approximate the height, the second setup cannot detect touches on the surface reliably. In [13], Takeoka et al introduce Z-touch that combines multi-touch and above the table interaction. Their system senses the approximate posture of fingers in the proximity of the surface using multi-layered infrared laser planes that are synchronized with shutter signals from a high-speed camera. Their system needs an extensive hardware setup and can detect interaction only in the area where the laser planes are installed.

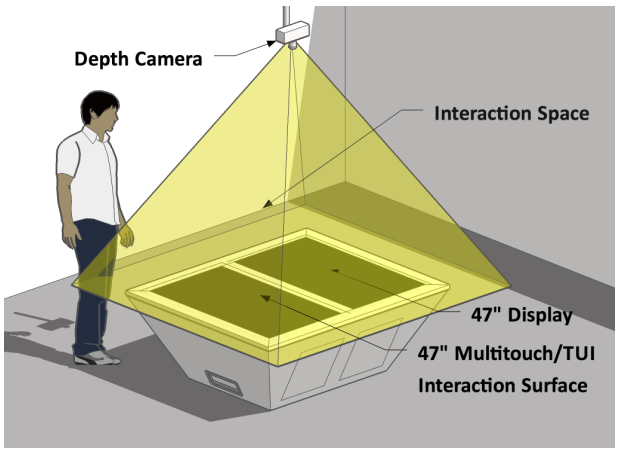


Figure 2: Schematic overview of our setup.

Several approaches use cameras to detect interaction above the surface: In [1], Agarwal et al detect high-precision multi-touch input on an arbitrary surface employing an overhead stereo camera system. However, their system is susceptible to strong reflections of the screen surface and does not take gestural input into consideration. In [12], Schick et al observe user interaction with large vertical displays from above using regular cameras for 3D reconstruction. Their approach allows a user to seamlessly switch between touch and pointing to enable interaction with remote areas of the display via pointing. However, their research does not consider interaction with horizontal surfaces making it not suitable for tabletop environments.

In [4], Dohse et al enhance an FTIR multi-touch tabletop environment with arm-tracking employing a top-view approach with an RGB camera to assign touches to users and to improve touch detection reliability. They detect arms using skin detection. Hence, the system works only with bare arms and it does not provide depth information.

Depth cameras have been employed in top-view settings in combination with tabletop environments for several purposes. In [14], Wilson uses a depth camera to create a height map of objects on an arbitrary surface. A driving simulation game allows players to drive a projected virtual car over real objects placed on the table using a game controller. Although users can hold their hands and arms in the camera view to be used as obstacles in the driving game, gestural input above the table is not further considered. Additionally, the tabletop itself is an arbitrary surface that cannot detect interaction on its own. A similar approach is employed to create the IncreTable [10]: Leitner et al use a depth camera over a tabletop onto which an image is rear- and front-projected. The depth camera creates a height map of objects on the table. The IncreTable allows playing a mixed reality tabletop game

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

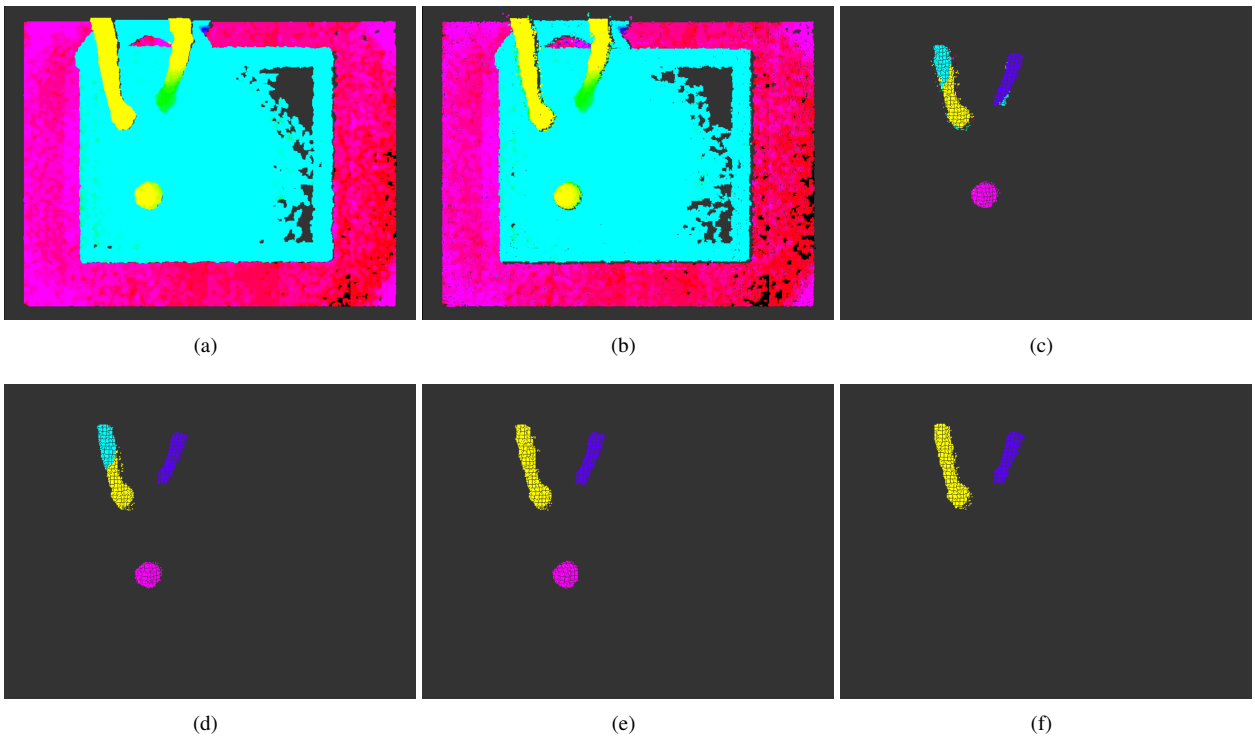


Figure 1: Results at different stages of the arm detection process with the Kinect: (a) raw depth, (b) after preprocessing, (c) after initial segmentation, (d) after cleaning segments, (e) after merging segments, and (f) the detected arms.

inspired by The Incredible Machine: A user can create and play a game with virtual and real objects. As with Wilson's approach, the table does not allow for touch or gestural input. However, users can employ an interactive pen to create the game boards. In [15], Wilson uses a Kinect that is mounted above an arbitrary surface to detect touches. This is achieved by looking only at pixels of the depth image that lie in an interval just above the surface. However, complementary gestural input to touch detection is not considered. In LightSpace [16], Wilson and Benko employ multiple depth cameras and projectors to enable interaction on arbitrary horizontal and vertical uninstrumented surfaces onto which images can be projected. LightSpace allows for various forms of interaction: To create a connection between displays and points, a user touches two different surfaces at once. Using another form of interaction, user may hold their hand next to the surface to pick up an object from the table and vice versa. Touches can be detected by employing the approach from [15]. In the LightSpace setup, the surfaces are uninstrumented and not capable of tracking tangible interaction on their own. Hence, LightSpace needs an extensive installation and does not allow for the enhancement of existing tabletop setups. Additionally, rather than employing sophisticated tracking algorithms, LightSpace works on the raw 3D mesh obtained from the depth cameras. Thus, LightSpace does not allow to detect high-level properties like exact arm positions, which are crucial for the detection of gestural input above the surface.

To conclude, there are various approaches that extend tabletop systems with in the air interaction. However, either they do not provide an interactive surface or they need extensive hardware setups. In the presented approaches, no system combines a standard IR illuminated tabletop setup with a depth camera that allows for on and above the surface gestural interaction.

3 SETUP

This section introduces our setup consisting of two systems: We employ an interactive tabletop surface for two-dimensional touch and tangible object interaction. For the detection of three-dimensional gestural input above and around the tabletop, we use a depth camera that faces the surface giving a top-view on the tabletop. Figure 2 illustrates the setup in a schematic overview and figure 7 shows a photo of the setup.

The tabletop system is custom built. The screen is divided in halves and houses two display technologies: One half is rear-projected and the other half uses an LCD panel. Both halves have a size of 47" and a resolution of 1920x1080 pixels. Currently, we only use the rear-projected panel for interaction. To allow for multi-

touch input and marker-based object recognition on the rear-projected part, the tabletop employs DI and a camera inside the table. Any object capable of reflecting infrared light resting on the surface shows up in the captured greyscale image of the camera and can be detected with computer vision methods. With a height of 55 cm, the table has approximately the height of a coffee table.

We have tested two different depth cameras based on active IR illumination independently of each other: the low-cost Kinect which costs approximately and the high-end camera CamCube that costs approximately 70 times more. The Kinect projects a pattern of IR light on the scene and calculates the distances based on a disparity map. Its depth image has a resolution of 640 x 480 pixels. The CamCube is an example for a camera that is based on the time-of-flight principle [9]. The CamCube illuminates the surroundings with strobes of IR light and calculates the pixel-wise depth image by evaluating the time that reflected light needs to return to the sensor. The CamCube's sensor has a resolution of 204 x 204 pixels. The distance between the surface and the camera is about 2 m, which results in a field of view that covers all of the surface with a surrounding of approximately 30 cm margin (see figure 2).

The detection of touches and tangible objects on the surface of the table is performed by our software framework *Actracktive*. Our software *SPIRITED*[3] detects three-dimensional user input above and around the tabletop. Although both systems – the tabletop environment and the depth camera – work independently, their IR illumination techniques interfere. Therefore, our software has to address several challenges. Section *Detecting User Input* introduces the independent functionality of Actracktive and SPIRITED and section *Integration* describes the challenges for the software in a combined setup and proposes appropriate solutions.

4 DETECTING USER INPUT

In our tabletop setup, users are most likely sitting around the table and reaching towards the center of the interactive surface when interacting with the system. In order to allow for rich interaction and to obtain information about the scene using this setup, we developed and implemented algorithms to detect and track user input on and above the surface. The input comprises touch, tangible objects and interaction above the surface.

4.1 Touches and Tangible Objects

On the table's surface, touches and marker-based tangible objects are allowed for interaction. While there are software packages freely available for each individual task and some also try to integrate both, we use our software framework *Actracktive* for flexible integration of computer vision based algorithms for input detection.

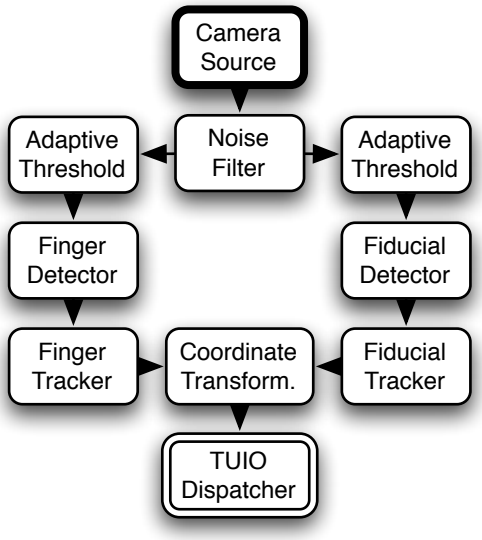


Figure 3: Overview of the processing steps used for detection and tracking of touches and fiducials on the table's surface.

Actracktive allows for a freely configurable graph of processing components that are executed continuously. The detection algorithms we use are based on the freely available software CCV [4] for touches and the fiducial tracking library developed by Bencina et al [2] for tangible objects. We implemented appropriate components to use these algorithms in our framework. The configuration we use is shown in figure 3. It consists of two distinct pipelines, one for each detection algorithm. Both pipelines access the already noise reduced camera image and proceed to process it with different parameters according to the needs of the detection algorithm. Touches are generally larger than the fiducials' components, and thus different thresholding parameters are needed.

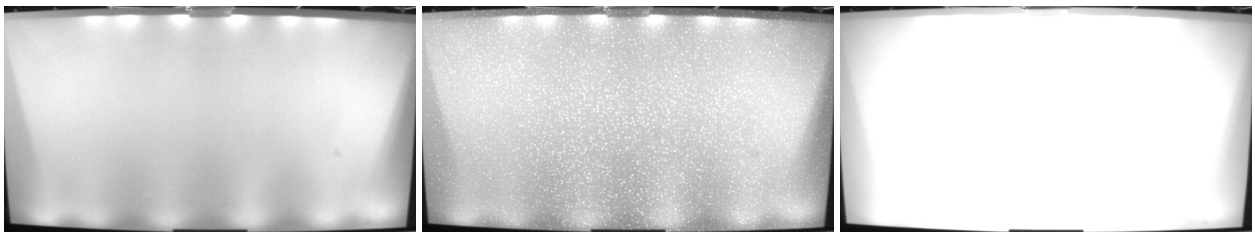
The detection of touches results in a logical representation that contains the two-dimensional position, speed and acceleration of the fingertip. Fiducials have the same representation, but with additional information about the rotation.

4.2 Interaction Above the Surface

In order to detect interaction above the surface, we developed an algorithm that is optimized for the detection of arms, which reach into a volume above the interactive tabletop by analyzing the height map of the depth camera.

[3] System for Presence and Interaction Recognition In Tabletop Environments using Depth

[4] http://ccv.nuigroup.com



(a) (b) (c)

Figure 4: Infrared light captured by the camera inside the tabletop system with (a) no depth camera, (b) the Kinect, and (c) the CamCube.

It is composed of a preprocessing on the raw depth data, followed by a segmentation and the detection of arms. Afterwards a simple arm tracking algorithm is performed.

4.2.1 Preprocessing

We obtain frames, where each consists of 640 $\times$ 480 pixels with the Kinect sensor and 204 $\times$ 204 pixels with the CamCube respectively. To reduce computational costs in further processing steps of the algorithm and to compensate for the distortion in the raw depth data, e.g., caused by reflection and quick movement, we apply one step of a Gaussian pyramid to compress and filter the image. Thus, we get an image composed of 320 $\times$ 240 pixels for the Kinect and 102 $\times$ 102 pixels for the CamCube.

4.2.2 Segmentation and Detection

Our segmentation algorithm is similar to an approach presented in [11]. Here, the segmentation of arms is composed of three steps. First, an initial segmentation is conducted by sequentially scanning the depth image once and assigning each pixel to a segment depending on the euclidean distances in the depth value. In the second step, small segments are merged into larger ones by minimizing the inter-cluster variance. Finally, a hierarchical merging of adjacent segments is performed using the minimal total scatter of the combined cluster as criterion.

Our approach performs an initial segmentation similar to the algorithm described in [11]. In this step, adjacencies of segments that are useful for merging segments are stored in a matrix. Afterwards, segments consisting of less pixels than an empirically found threshold are removed to compensate for segments arising from distortion in the depth data. The following merging step is where our approach differs essentially from [11]. In their approach, both merging steps weight all three dimensions equally to calculate the inter-cluster variance and total scatter. Since arms have an elongated shape, two segments of the same arm, e.g., forearm and upper arm have a rather high inter-cluster variance and total scatter. In comparison, both measures are smaller for two segments coming from different crossing forearms. As a consequence two crossing arms would be more likely merged than two segments, which belong to the same arm. To limit the distance using depth is not an option to avoid this problem as users might not always be holding their arms in parallel to the xy plane, i.e. the table surface.

In our approach, only one merging step is conducted instead of two steps as proposed in [11]. We use the euclidean distances to the common regression plane of two segments as the merging criterion. The detailed procedure is described in the following. Considering a set of segments $S = \{s_i | i = 1...n\}$ where $s_i = \{p_k^i = (x_k, y_k, z_k) \in \mathbb{R}^3 | k = 1...m_i\}$ resulting from the initial segmentation step, for each s_i a set of adjacent segments $A_i = \{s_j | s_j \in S$ and s_j adjacent to $s_i\}$ has been determined in the same step. Two segments are defined as adjacent if they lie in a k-neighbourhood to each other in the depth image raster. Let $s_{ij} = s_i \bigcup s_j$ be the combined segment of each pair of s_i and s_j and $R = (r, \vec{n})$ the regression plane with r being a point on R and $\vec{n}$ being the unit normal vector of R. The average distance d of s_{ij} to R_{ij} is

$$d(s_{ij}, (r, \vec{n})) := \frac{1}{|s_{ij}|} \cdot \sum_{k=1}^{|s_{ij}|} \big((s_{ij}^k - r) \cdot \vec{n} \big).$$

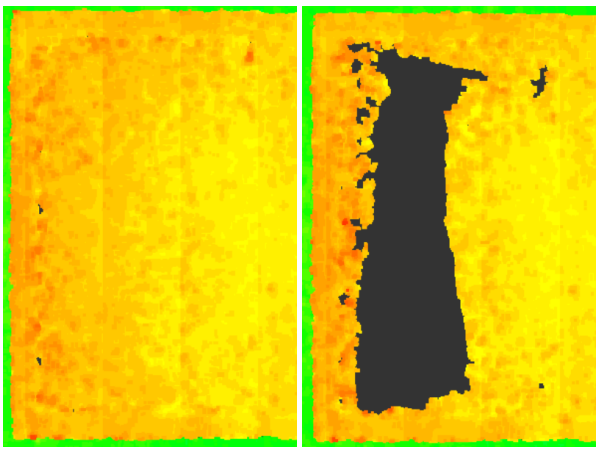
The best adjacent segment s_i^{best} is determined as

$$s_i^{best} = \begin{cases} \underset{s_j \in A_i}{\operatorname{argmin}} \big(d(s_{ij}, R_{ij}) \big), & \text{if } d(s_{ij}, R_{ij}) < T_S, \\ \emptyset, & \text{else} \end{cases}$$

where T_S is a predefined threshold that has been found empirically. Finally, two segments are merged as $s_i \bigcup s_i^{best}$ until no more matches are found.

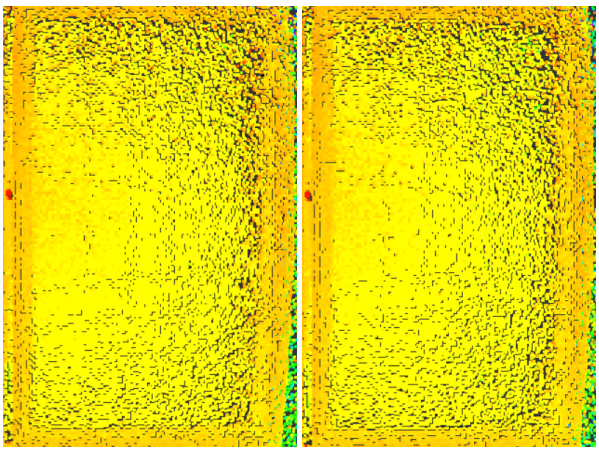
The detection of arms is again based on the approach presented in [11]. We make use of the elongated shape of arms and use the eccentricity $e = \left(\frac{\lambda_0 - \lambda_1}{\lambda_0 + \lambda_1} \right)^2 \in [0, 1]$ with λ_0 and λ_1 being the length of the two largest principal components of a segment and $\lambda_0 > \lambda_1$. The larger e, the more elongated is a segment and we classify a segment as arm if e is larger than an empirically found threshold.

An arm is represented by an ellipsoid E_a resulting from a principal component analysis of the arm's segment. The



(a) (b)

Figure 5: Depth data acquired by the Kinect (a) without and (b) with enabled direct illumination from the tabletop system.



(a) (b)

Figure 6: Depth data acquired by the CamCube (a) without and (b) with enabled direct illumination from the tabletop system.

three components a, b and c and the centroid d are represented in a matrix $A = [a, b, c, d] \in \mathbb{R}^{3\times4}$, which is passed to the application.

4.2.3 Tracking arms

A simple tracking algorithm based on the euclidean distance between the centroids of extremities in the previous frame and the centroids of extremities in the current frame is implemented. Two segments s_1 and s_2 in sequential frames I_{i-1} and I_i with $i = 1...n$ have the same tracking id if $|c_1 - c_2| < T_A$, c_1 and c_2 being the centroids of s_1 and s_2 respectively and T_A being an empirically found threshold.

4.2.4 Implementation and Discussion

To retrieve depth data, we use the SDK provided by PMDTec for the CamCube, and the Open Source library libfreenect provided by the OpenKinect project [5].

The implemented algorithm works at interactive frame rates and is able to detect and track arms reliably. As the algorithm detects all elongated shapes, it cannot discriminate elongated objects from arms. Moreover, if a user is holding an object in the hand, the object cannot be separated from the arm. Figure 1 shows the results of the different steps of the algorithm exemplarily. We also realized first applications using SPIRITED, e.g., for hybrid hover interaction as shown in figure 9.

[5] http://openkinect.org/wiki/Main_Page

5 INTEGRATION

When not used in a combined setup, the input detection software Actracktive and SPIRITED works reliably. However, as we aim to integrate both systems, we have to address the challenges caused by the usage of infrared light by both systems. Additionally, it is mandatory to bring all detected objects into a common coordinate system and also to define a format to use for sending the input to the application. In the following, we discuss the necessary steps to achieve an integrated setup.

5.1 Technical Aspects/Challenges

As the whole setup is based on sensing infrared light of roughly the same wavelength (near-infrared at about 850 nm), the separation of the different light sources is the main challenge of an integration. In general, the detection of touches on the surface will be potentially disturbed when the area around the finger is lit by any light with the same or higher intensity than that used by the table itself. The Kinect faces the same problem and will generally be unable to measure depth in areas where the projected pattern hits a surface illuminated by some other infrared light source. The CamCube, however, can hardly be disturbed by external light sources as its own light is emitted at a high intensity and additionally is pulsed at a high frequency (around 20 MHz). Thus, it can obtain a depth image with only minor decrease in quality even if other light sources illuminate parts of the scene.

These different levels of robustness to external influences show that it is not possible to arbitrarily combine the different devices. In the following, we show the results of the device combinations and how they influence each other.

Figure 4 shows how the camera of the tabletop system reacts to the presence of the depth cameras. In comparison, it becomes clear that the two kinds of depth cameras have a drastically different impact on the tabletop system. The Kinect causes bright but small dots to appear, which can be easily filtered out prior to the touch and object detection. The bright, flashing lights of the CamCube make it impossible to detect anything on the surface with Acktractive. Figure 4(c) is captured when the CamCube's lights were fully switched on. During operation, the brightness of the table's camera image flickers heavily. The flickering pattern depends on the capturing frequency of the CamCube and the table's camera respectively. This rules out using the CamCube for a general integration into our setup, as it would require a separation of the infrared light of the tabletop system and the CamCube. A hardware solution could be to use a wavelength for the IR illumination in the table that differs from the depth camera's illumination wavelength. An optical bandpass filter on the table's camera could then filter out the light from the depth camera. A software solution could filter out all frames that are overexposed by a flash of the CamCube and use only those that were taken between two flashes. While an implementation should be feasible, it would drastically reduce the frame rate of the tabletop system and thus result in a worse tracking performance.

Still, the Kinect has an influence on the tabletop's input detection, as the small dots might be interpreted as touches. In order to make the existing systems work together, only a slight modification of the processing components parameters are necessary. It is important to note, that the processing configuration does not structurally differ from the one used in a standalone setup. In case the system detects the Kinect's dot-pattern as touches, it is only necessary to adjust the minimum size of a fingertip in order to exclude them from touch detection. Another way to achieve this would be to filter out the Kinect's pattern based on contrast, but this is not an option with our system, as the brightness of touches is not higher than the Kinect's pattern. When other systems are used, this way of filtering should be considered, as it allows for an earlier separation and also allows touches to be the same size as the dots of the Kinect's pattern.

Even though the Kinect is better suited for integration, figures 5 and 6 show that the tabletop system influences the Kinect's image significantly in contrast to the CamCube's image. Due to the constant, homogeneous infrared light on the table's surface, the Kinect is unable to calculate the depth of the surface, but as we do not need any depth information *on* the surface, this is actually not an issue. Interaction performed above the surface occludes the tabletop's projected light and thus no modification of our arm detection algorithm is necessary.

Figure 7: Calibration objects on the tabletop system

5.2 Calibration / Combined Coordinate System

After physically combining both systems, it is necessary to define a common input coordinate system C, that allows applications a unified handling of two- and three-dimensional input. Both systems use internal calibrations that establish local coordinate systems for the objects tracked by each system. As applications running on the tabletop system use the display coordinate system for their output, it makes sense to establish the common input coordinate system in terms of the display coordinate system. Thus, we define C as a left-handed coordinate system with its origin at the upper left corner of the display. The x and y axes are defined by the display coordinate system and the z axis is defined perpendicular to the xy plane (and because of the left-handedness, it points upward from the table).

5.2.1 Touch/Props

The surface tracking system of the tabletop uses a two-dimensional coordinate system that is equivalent to the display coordinate system. Therefore, it does not need any transformation, as it is identical to the xy plane of C.

5.2.2 Depth data

From the raw depth data captured by the Kinect sensor a set of coordinates

$$P = \{p_i = (x_i, y_i, z_i) \in \mathbb{R}^3 | i = 1...n\}$$

is determined in the right-handed Kinect camera coordinate system C_K. In order to use the information provided by the depth camera in a tabletop application it is necessary to transform P such that it is in the common coordinate system C.

To obtain the affine transformation T that maps C_K on C, we need to determine twelve unknowns. To achieve this, we need to know the coordinates of at least four points in C_K and in C. For a higher precision, we establish nine point pairs to form an overdetermined linear

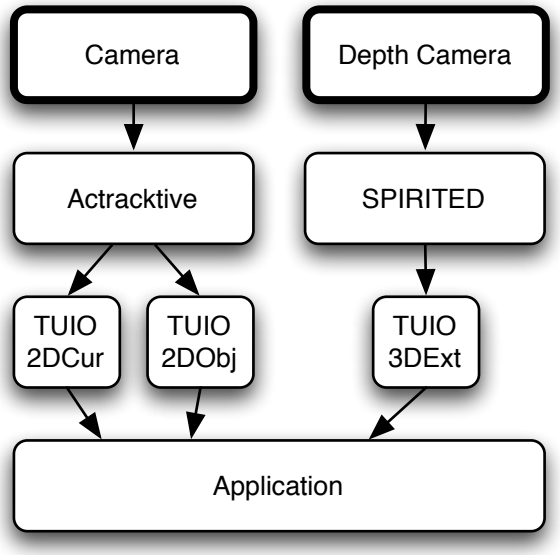


Figure 8: Overview of the user input processing in the tracking software and communication with applications.

system $P_R = T \cdot P_S$ with $P_R, P_S \in \mathbb{R}^{9\times 3}$ which can be solved with a least squares method. To establish the system, we arrange nine calibration objects that consist of a pole with discs attached to both ends on the table surface as shown in figure 7. Beginning with 5 cm height, the objects differ in height by two centimeters each to assure that the centers of the upper discs do not lie in the same plane. Placing the objects on predefined locations on the surface, we know the coordinates of the center of each disc in C and define these as reference points P_R. The order is defined by row starting in the upper left corner of the tabletop display. In SPIRITED's calibration mode, we select the center of each object in the depth image obtained with the depth sensor in the same order to obtain P_S.

5.3 Communication

Figure 8 shows how the user input is processed in the combined tabletop and depth camera setup. We employ the TUIO protocol [8] to provide information about tracked touches, tangibles and body parts to applications. Touches are sent with the */tuio/2Dcur* profile and tangibles with the */tuio/2Dobj* profile.

Although there are TUIO profiles for 2.5D and 3D interaction, these are suitable to convey touches and tangible interaction but not information about body parts. Thus, we have decided to define our own TUIO profile called */tuio/_3Dext* that encodes the transformation matrix A sent from SPIRITED (see section *Detecting User Input / Interaction Above the Surface / Segmentation and Detection*).

A /tuio/_3Dext set message is composed as following:
/tuio/_3Dext set $sid\ pid\ a_x\ a_y\ a_z\ b_x\ b_y\ b_z\ c_x\ c_y\ c_z\ d_x\ d_y\ d_z$

sid is the TUIO session ID of the body part. pid may contain the id of the person to which the body part belongs.

6 INTERACTION

Our setup allows to implement interaction based on detected body parts in the tabletop environment. This interaction can be combined with touch or tangible object input on the interactive surface. Hence, it would not make sense to use above the tabletop interaction for input that can be performed more easily via touch like selection, dragging or rotation gestures.

Still, around and above the tabletop interaction can be used for a broad range of input techniques that complement touch and tangible object input. In the following, we exemplify four techniques: hover interaction, hybrid hover interaction and multi-user interaction.

6.1 Hover Interaction

An example for hover interaction would be the semantic tooltip: a user holds the hand for three seconds above an interface element on the surface in order to receive additional information. The time delay is necessary because the system has to discriminate between intentional and unintentional interaction. By moving the hand up and down, different semantic levels of detail of the additional information can be presented. For instance, if the user holds the hand for three seconds above a certain point on a map displayed on the tabletop, the address of this point is shown. If the hand is raised, more information about this address becomes available, for example elevation or current weather information. This interaction technique would not be feasible with conventional setups because in a touch setup there is no such thing as a hover gesture because a finger either touches the surface or not. Additionally, employing a depth camera allows to use the height of a hand above the surface as a parameter for the semantic level of detail interaction metaphor.

6.2 Hybrid Hover Interaction

As it could be difficult to keep a hand hovering over the same UI element while moving it up and down, it would be a solution to employ hybrid interaction with above the tabletop input in combination with touch. For instance, a user performs a hybrid two-handed gesture by touching an interface element on the surface with the right hand and hovering over the same element with the left hand for three seconds to assign the left hand

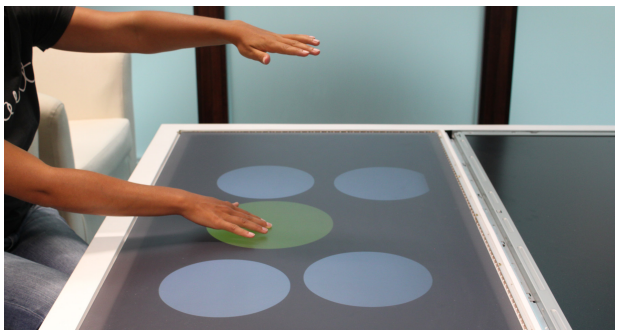


Figure 9: A user performing hybrid hover interaction by touching an object with the right hand and moving the left hand up and down.

to the element and the touch (see figure 9 and video: http://youtu.be/nPyGOceTEGo). Now, there is no need to keep the left hand hovering over it: As long as the right hand is touching the element, the left hand stays assigned to it. Now the user can easily move the left arm up and down to control the level of detail.

6.3 Multi-User Interaction

When users approach the tabletop environment, they are monitored by our tracking system. An example for multi-user interaction would be the assignment of input to a user as soon as a user touches or puts a tangible onto the surface. The assignment of touches to a user has already been considered for instance in the DiamondTouch setup [3] or in Dohse et al's top-view RGB camera setting [4]. However, with the DiamondTouch setup, a user has to stand or sit on a special pad to allow for the detection of the user, which is not necessary with our approach. Contrary to Dohse et al's approach that only detects bare arms, our system has the potential to recognize all input around and above the table. Also tangible objects can be assigned to users. This property can be employed to create an ownership metaphor: Every touched interface element or tangible object stores which user touched it the last time and can be owned by a user. Only users who own an element or tangible object may manipulate it. Additionally, an element can orient automatically towards the owner. To forward an element or tangible object, the owner touches it at first before a second user touches the element. After the owner released the object, it belongs to the second user.

7 CONCLUSION AND FUTURE WORK

7.1 Conclusion

In this paper we have presented a system that detects three-dimensional gestural input above a tabletop environment with a depth camera. The depth camera monitors the tabletop environment from above giving a top-view onto the surface. To detect the input in the top-view setup, we developed a software that employs computer vision methods. We tested two different kinds of cameras with our software: the low-cost Kinect camera and the high-end CamCube. First results show that the detection works at interactive frame rates with both cameras. However, due to a higher resolution, the detection works more robust with the Kinect.

We have combined our three-dimensional user input detection system with an interactive tabletop environment that is based on diffused illumination. The depth camera interferes with the tabletop environment as both work with IR illumination. Although the gesture input detection is not disturbed by the light of the tabletop system, we have figured out that the illumination of the CamCube interferes too heavily with the tracking of the tabletop. However, with the Kinect, the tangible interaction on the tabletop system still works well with a modification of the image processing in the tracking software. Therefore, the low-cost Kinect camera offers the potential to extend many other existing tabletop environments with interaction above the surface.

Gestural input above and around the tabletop complements tangible interaction on the surface. For instance, we suggest to use touch interaction in combination with gestural input above the surface for a hybrid hovering technique to create a semantic tooltip. Additionally, gestural input on and above the surface provides possibilities for new forms of multi-user and proxemic interaction in the field of interactive tabletop environments.

7.2 Future Work

One improvement to our three-dimensional gestural input detection would be to actually detect which body part has been detected. For this purpose, it seems to be a promising approach to employ a skeletal model. However, so far skeletal models have mainly been considered for front-view setups where the camera faces a user's front. By introducing a skeletal model for a top-view approach, we should be able to track user's body parts employing their skeletal model.

Finally, there is the whole field of exploiting the potential of the setup introduced in this paper: novel interaction techniques are to be conceived and evaluated in a cornucopia of application fields.

References

[1] A. Agarwal, S. Izadi, M. Chandraker, and A. Blake. High precision multi-touch sensing on surfaces using overhead cameras. *Horizontal Interactive*

Human-Computer Systems, International Workshop on, pages 197–200, 2007.

[2] R. Bencina, M. Kaltenbrunner, and S. Jorda. Improved topological fiducial tracking in the reactivision system. In *Computer Vision and Pattern Recognition - Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on*, page 99, 2005.

[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] K. C. Dohse, T. Dohse, J. D. Still, and D. J. Parkhurst. Enhancing multi-user interaction with multi-touch tabletop displays using hand tracking. In *Proceedings of the First International Conference on Advances in Computer-Human Interaction*, ACHI '08, pages 297–302, Washington, DC, USA, 2008. IEEE Computer Society.

[5] F. Echtler. Multitouch with hacked Kinect. http://www.youtube.com/watch?v=ho6Yhz21BJI, 2011.

[6] O. Hilliges, S. Izadi, A. D. Wilson, S. Hodges, A. Garcia-Mendoza, and A. Butz. Interactions in the air: adding further depth to interactive tabletops. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, UIST '09, pages 139–148, New York, NY, USA, 2009. ACM.

[7] S. Izadi, S. Hodges, S. Taylor, D. Rosenfeld, N. Villar, A. Butler, and J. Westhues. Going beyond the display: a surface technology with an electronically switchable diffuser. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, UIST '08, pages 269–278, New York, NY, USA, 2008. ACM.

[8] 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)*, pages 1–5, 2005.

[9] A. Kolb, E. Barth, R. Koch, and R. Larsen. Time-of-Flight Sensors in Computer Graphics. *Eurographics State of the Art Reports*, pages 119–134, 2009.

[10] J. Leitner, M. Haller, K. Yun, W. Woo, M. Sugimoto, and M. Inami. Incretable, a mixed reality tabletop game experience. In *Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology*, ACE '08, pages 9–16, New York, NY, USA, 2008. ACM.

[11] S. Malassiotis and M. G. Strintzis. Real-time hand posture recognition using range data. *Image Vision Comput.*, 26:1027–1037, 2008.

[12] A. Schick, F. van de Camp, J. Ijsselmuiden, and R. Stiefelhagen. Extending touch: towards interaction with large-scale surfaces. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ITS '09, pages 117–124, New York, NY, USA, 2009. ACM.

[13] Y. Takeoka, T. Miyaki, and J. Rekimoto. Z-touch: an infrastructure for 3d gesture interaction in the proximity of tabletop surfaces. In *ACM International Conference on Interactive Tabletops and Surfaces*, ITS '10, pages 91–94, New York, NY, USA, 2010. ACM.

[14] A. Wilson. Depth-sensing video cameras for 3d tangible tabletop interaction. In *Horizontal Interactive Human-Computer Systems, 2007. TABLETOP '07. Second Annual IEEE International Workshop on*, pages 201 –204, 2007.

[15] A. D. Wilson. Using a depth camera as a touch sensor. In *ACM International Conference on Interactive Tabletops and Surfaces*, ITS '10, pages 69–72, New York, NY, USA, 2010. ACM.

[16] A. D. Wilson and H. Benko. Combining multiple depth cameras and projectors for interactions on, above and between surfaces. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, UIST '10, pages 273–282, New York, NY, USA, 2010. ACM.