Integrating Pointing Gesture Detection for enhancing Brainstorming Meetings using Kinect and PixelSense

Andreas Kunz[1], Ali Alavi[1], Philipp Sinn[1]

[1] Innovation Center Virtual Reality (ICVR), ETH Zurich, 8092 Zurich, Switzerland, E-Mail: kunz@iwf.mavt.ethz.ch

Abstract

Microsoft PixelSense is able to detect multitouch input and tagged objects as well, which makes it suitable to be used in net-based brainstorming sessions within small teams. However, any gestures above the table cannot be detected, which makes net-based brainstorming sessions less intuitive. Thus, we present a solution how Kinect can be used together with PixelSense to overcome this limitation without interference between the two devices.

Keywords:

Kinect; Deictic Gestures; Computer Supported Collaborative Work

1 INTRODUCTION

Nowadays, many big ideas incept from a team of individual designers, sitting together in a brainstorming meeting. This collaborative approach uses the collective knowledge and creativity of the team, and is beneficial to the major stakeholders of the meeting (the designers, the organization, and the customers) [1]. The advent of computers and digital media, and increased availability of high speed networks, enabled remote brainstorming sessions. Thus, members of a group, distributed among different locations, may participate in such meetings and effectively participate in the ideation phase of developing a product. This has two main advantages:

Firstly, the need of commuting between different locations can be eliminated. This is particularly interesting for international firms with offices located around the globe.

Secondly, there are reduced social inhibitions among group members. It has been seen that many team members, particularly in the presence of a senior member, will withhold commenting for fear of criticism or negative evaluation, a behavior that depresses the ideative efficacy [2]. Moreover, in Electronic Brainstorming Systems (EBS), “because participants do not see each other (even if they are in the same room), attention is essentially paid to ideas […] helping to reduce redundancy [of ideas] and improve task performance” [3].

In spite of these advantages, net-based brainstorming has certain disadvantages, mainly due to the fact that available EBSs are not capable of transmitting different forms of interaction: as depicted in Figure 1, these interactions happen between two types of entities, and in two different spaces (see Figure 1):

Interaction between humans and the digital media happens on the “task space”, which can be a tabletop computer, and above it, for example pointing to the contents on the table. This is where the generated artifacts (forms of mindmaps, sketches or written notes) belong to.

Interaction between team members, which according to [3] consists of verbal (words), vocal (intonation), and visual (body language) elements, takes place in communication space.

For an efficient net-based brainstorming, we should decide on a subset of these communication elements to be captured, aligned, and correctly transferred to the remote side.

While most EBSs are capable of transmitting the content of the task space and also verbal and vocal elements of the communication space, they come short in a proper transmission of visual parts of human-to-human and human-to-content communication (facial expressions, hand gestures, nodding, shrugging, and so forth).

Many approaches try to overcome this problem by delivering this visual content using video-conferencing, which is not efficient. Firstly, many researches in social psychology show that lack of social cues in brainstorming meetings leads the group members to focus on the task instead of on the people, thus improving the task performance [3], [5]. In other words, delivering all the social content of the meeting is not ideal. As a result, video-conferencing may increase social inhibition among group members. Moreover, in order to view remote collaborators, a big portion of the digital media screen needs to be dedicated to showing other collaborators faces and bodies, leading to either a very small task space, or the need of additional screens. Both adversely affect the quality of the meeting. Hence, we are interested in transmitting only the essential visual communication elements which contribute to the meeting while not adding much social inhibition. Additionally, it does not occupy significant visual resources of the meeting or distracts the participants’ attention.

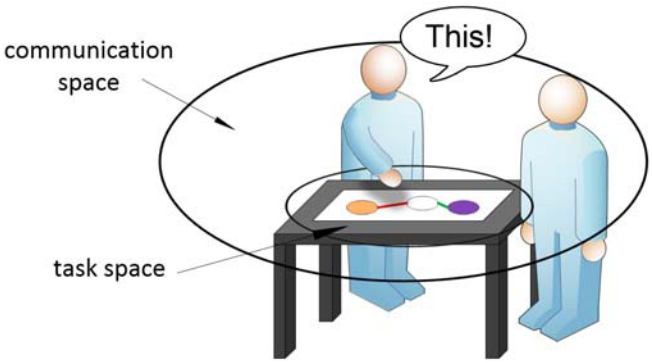

Figure 1: Brainstorming integrates communication- and task-space.

In the following paper, we introduce a new technology for detecting pointing gestures, relating them to the content of the task, and informing the remote partners about it. The main part of this process is to detect and track the pointing gestures. Since pointing gestures are typically performed in the free space (communication space) above the interactive table, PixelSense's sensors cannot detect them anymore with a sufficient resolution. Thus, an additional tracking system is required. Although a marker-based tracking system can accomplish this task, we are interested in a solution that is less intrusive to the user, because wearing the markers might neither be possible nor desirable for some of the participants. Moreover, detecting hands and fingers with color-based cameras is not an option neither, because of the variety of skin tones in a large group of people, and also a large range of colors available in the task space. Hence, we try a different approach by tracking hands using a depth camera. Even though tracking hands using depth data does not have the mentioned problems of other tracking systems, a depth camera cannot be easily used in presence of a PixelSense touch screen, since they both work with infrared light, thus interfere with each other. This paper offers a solution to overcome this interference problem between the Microsoft Kinect depth camera and the Microsoft PixelSense touch screen. Once the pointing gesture's orientation is detected, the remainder of the paper will describe, how the target of the pointing gesture will be displayed to the remote partners.

2 RELATED WORK

The importance of aligning the layers from Figure 1 was already stated in earlier work. According to Ishii et al. [6], people feel it difficult to communicate if they cannot tell whether the remote partner is listening carefully or not. An early example of a shared workspace was given by Krueger [7]. However, in his setup the shared workspace was more a shared view space, since it was not possible to interact with the artifacts. In 1990, Tang and Minneman [8] introduced VideoDraw, a device that allows the partners to share a drawing surface. It consists of video cameras aiming at the screen, whereby each camera is connected to a monitor on the other side. As both partners draw with whiteboard markers on the screen, the video camera captures these markers and the accompanying hand gestures, which are then transferred to the other side. VideoWhiteboard [9] features rear-projection of the shared task space, and a camera which is also placed behind a 90'' projection screen. The partners see the complete image, real and video marks, as well as the shadows of their remote partners' gestures and actions. Bly [10] conducted an exploratory study to investigate the use of a drawing surface in design sessions. In one of her settings, two designers were geographically separated and connected via video tools. Bly observed that in the sessions that provided visual contacts, "gestures constituted a significant portion of the drawing actions that took place". In order to allow designers to work remotely by sharing a drawing surface, Bly and Minneman developed Commune [11]. This system provides two separate horizontal writing surfaces, each consisting of a horizontally mounted CRT monitor, and a transparent digitizing tablet mounted directly on top of the screen. On each writing surface, collaborators can gesture and make marks by using a stylus. However, the remote partner was not captured, but his gestures were restricted to a telepointer that was transferred to the remote site.

With ClearBoard [6], Ishii et al. bring together task space and communication space, since the system allows keeping eye contact while working on an interactive surface. Kirk et al. [12] [13] underlined in their study the importance of hand gestures that are in correct relation to the task space. Stotts et al. [14] suggested using remote collaboration groupware that displays live video embodiments situated within the shared workspace. With "The Vis-a-Vid (VAV) Transparent Video Facetop", they presented a respective user interface. It has cameras that acquire live video embodiments showing the collaborators' faces. The local live video embodiment is displayed as visual feedback for controlling where the hand is placed. Further, pointing gestures are detected to allow controlling the computer's mouse pointer. Due to the camera positions, gestures must be performed in free space, making VAV less useful for on-screen interaction.

Wellner presented the Digital Desk [15] and the Double Digital Desk [16]. The Digital Desk consists of a normal office desk with a projector and a camera above it, both pointing to the desk's surface. The projector allows superimposing digital artifacts on physical ones lying on the desk. The user can interact with the system using a mouse, a digitizing tablet and a stylus, or by pointing with his bare finger that is tracked through image processing of the acquired camera images. Agora [17] is a remote collaboration groupware system that supports shared desktop artifact activities and remote gesturing. However, the task space is shared as video only, entailing all associated drawbacks. VideoArms [18] [19] is an elaborate conferencing system that supports collaborators' natural use of hand gestures. The system acquires people interacting on shared task spaces by means of a camera that is on-axis with the display device. Therefore, the context of hand gestures is preserved, e.g. deictic gestures pointing out a shared artifact can be correctly interpreted by the remote collaborator. To improve the recognition of users, CollaBoard [20][21], exploiting polarized LC light emission, uses polarizing filters in front of the camera in order to segment a person in front of a highly dynamic background on an LC-screen.

Like with the CollaBoard, many of the systems mentioned before use a tracking system in order to detect the interaction devices' positions. For a vision-based capturing of hand-gestures in the work space, resistive or inductive touch screens for interactive devices in the task space do not interfere with the camera. Even IR-based touch screens can be used in the task space, as long as the IR-emission is not in the camera's viewing direction. However, today's active tables such as PixelSense [22] actively emit non-polarized IR-light. Moreover, cameras are usually very sensitive to infrared light and thus are driven into saturation, resulting in the fact that RGB-cameras can hardly see different colors anymore. Thus, the technology proposed in CollaBoard cannot be used anymore. In addition, all setups so far only detect a rough pointing direction (in regard to screen coordinates x and y), neglecting that deictic gestures also have an orientation, which requires additional depth information.

To further improve the close coupling between task and work space, we propose a solution which uses PixelSense together with

a Kinect's depth sensing camera to reliably track gestures above the table.

3 APARATUS AND ENVIRONMENT

Our study focuses on brainstorming meetings happening around a tabletop computing system. Microsoft PixelSense is such a system, with a wide set of features, which enables the users to run the extended set of Microsoft Windows software and applications. The table is also capable of detecting multi-touch interaction, enabling the users to interact with the computer system in a natural way. We use PixelSense because of its wide availability, and also because of the ease of software development on this platform.

We decided to use Kinect also because of its availability, reasonable pricing and its available programming libraries and frameworks which facilitates realization of the experiments.

Figure 2: Brainstorming around PixelSense. Most gestures happen above the screen.

As depicted in Figure 2, users employ different forms of hand gestures around the surface. Because of the physical positioning of the users around the table, the pointing gestures happen directly above the table's screen. This is confirmed by a preliminary user study we performed, which showed that most of the relevant gestures above the table are in a height of up to 465 mm and within the screen area. The Kinect has to have a bird's eye view onto the PixelSense in order to have an unobstructed view of the scene regardless of the number of users. Taking the height of the PixelSense table and Kinect's aperture into account results in the following setup (see Figure 3). The setup is designed in such a way that Kinect's field of view exactly covers the interactive region of PixelSense.

This setup causes frequent unwanted touches appearing on PixelSense, completely disturbing the normal interaction of users with PixelSense. This is due to the fact that both PixelSense and Kinect use infrared emitters and sensors, working with similar wavelengths (830 nm). Thus, the interference between these two devices is a major problem. This interference has less impact on Kinect than on PixelSense, meaning that depth sensing by the Kinect still works, while the PixelSense cannot be used anymore for touch detection.

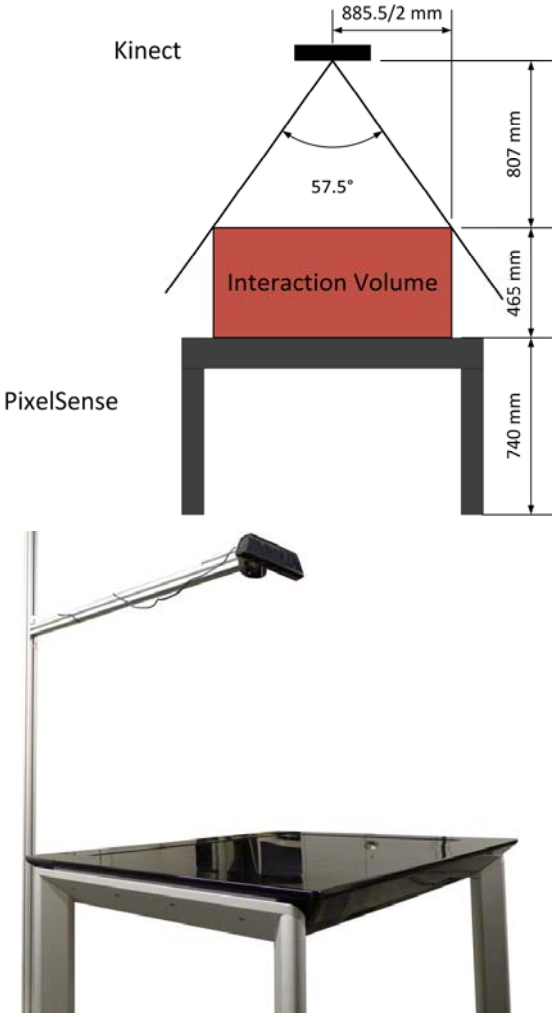

Figure 3: Realized setup for the Kinect & PixelSense tracking system.

The Kinect's depth sensing is based on a structured light approach. Its IR projector emits a dot pattern into the room, while its IR camera observes the scene and compares the disparity between the dot pattern and a reference. Thus, the depth at each dot's location can be determined. Depth detection fails if the Kinect cannot see its dots anymore. Common causes are reflective surfaces, which cause disturbances, too large distances or high levels of IR intensity in the scene. In the latter case, the dots are insignificantly brighter than the rest of the environment (low contrast) and cannot be detected in the IR image reliably.

PixelSense features an array of IR sensors in the screen for touch detection. Additionally, IR light is homogeneously emitted through the entire screen. In the IR image of the Kinect, the PixelSense screen appears as a bright rectangle due to its IR emission (see Figure 4, top). Consequently, the Kinect cannot see its own pattern anymore (contrast issue) in this region and returns the depth value "-1", which means "error" (errors are output as black in the depth map). See Figure 4, bottom.

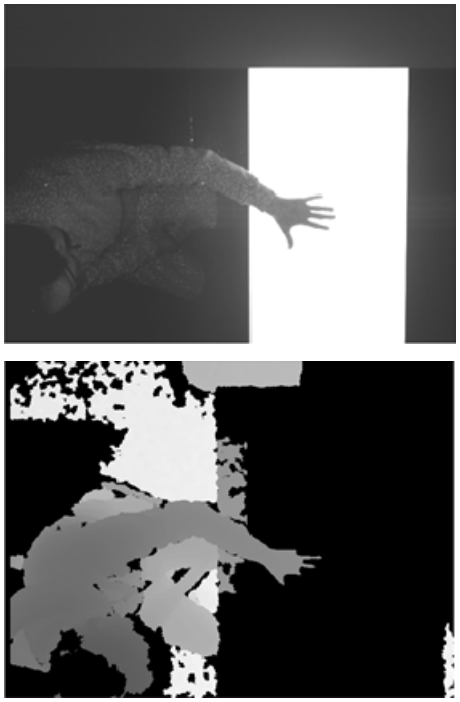

Figure 4: Kinect's view (bird's eye view) on the PixelSense with a user sitting left of it and stretching his hand out above the screen. Top: Kinect's IR image. The Kinect's dot pattern can be seen on the user and faintly on the floor. Bottom: Kinect's calculated depth image. The PixelSense's screen and glossy frame cannot be detected (black pixels) but the user and partially the floor are detected.

In addition, the frame around the screen is glossy. This causes specular reflections which disturb Kinect as well. Essentially, neither the PixelSense screen nor its frame can be detected by the Kinect. However, detecting gestures above the PixelSense is not a problem. Any object, e.g. a hand and forearm, on or above the PixelSense can still be detected reliably, since the dots are visible there (see Figure 4, top). However, due to Kinect's resolution, it is not possible to detect individual fingers continuously (see Figure 5).

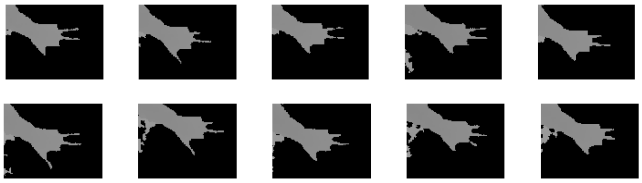

Figure 5: Ten consecutive frames of PixelSense depth data based on Figure 3 (cropped to only show hands). A stretched out hand can be seen from above. The fingers are irregularly detected.

While Kinect is still able to detect objects above PixelSense without any modification, it causes noticeable distortions on the PixelSense. This is due to the PixelSense's touch detection principle. It relies on the IR light emitted through the screen to detect inputs. If objects are close to the screen, the IR light is reflected into the sensors. Based on this data, PixelSense detects touches, tags, and blobs.

The Kinect projects its IR dots (which have several different diameters due to the tracking procedure of Kinect) onto the PixelSense (see Figure 6).

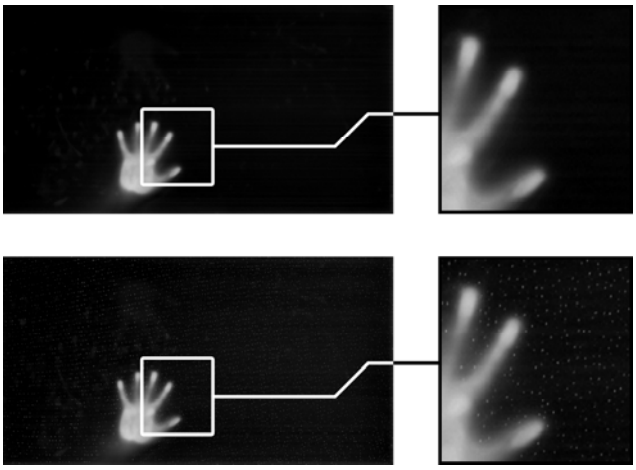

Figure 6: Raw sensor data of a hand on the PixelSense screen, with a part magnified. Top: Kinect off. Only the hand and some smudges are visible (the smudges are not interpreted). Bottom: Kinect on. Many small dots are visible in the raw data.

These dots are above PixelSense's detection threshold and are thus erroneously detected as inputs. Depending on the size of the dots, they are either interpreted as blobs or as touches. While blobs can be easily filtered out by the PixelSense's software, touch inputs should not be filtered, since they are required for the interaction. However, this results in unwanted click events. Typically such an event is active as long as the object rests on the screen. It is independent of time, changes in size and position (as long as the change in position is smooth). Most of the Kinect's click events have durations below 50 ms, but can go up to a few seconds in rare cases (see Figure 7).

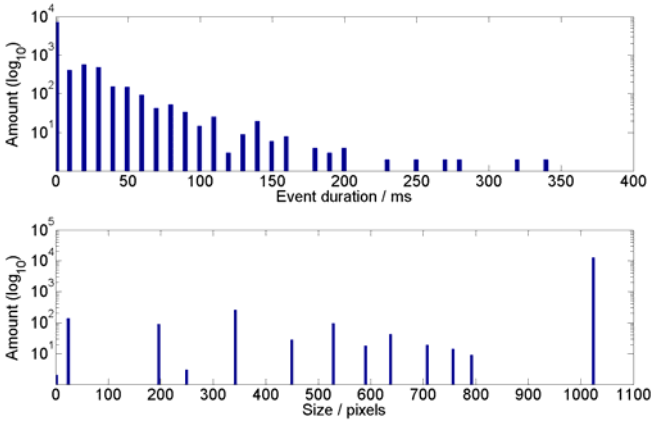

Figure 7: Top: Histogram of the duration of individual events (please note: events active during one timeframe have duration 0). Bottom: Histogram of registered sizes per timeframe independent of event.

Figure 7 shows the unwanted touch events occurring in a 15-minutes observation interval. In total, 9361 touch events were triggered by the Kinect, which corresponds to 624 Kinect touches per minute. These numbers show that without any additional modification of the system, the PixelSense cannot be used anymore. These touch events appear to be the same size independent of the room illumination, i.e. whether the fluorescent lights were switched on or off, and Kinect's distance to PixelSense.

Another peculiarity is that the dots are not detected continuously, hence the short durations, despite a static pattern. Also, most events grew bigger in the sensor readings in discrete steps or remained at one specific size.

To be able to design more sophisticated filters, 'real' finger touches were also recorded (see Figure 8). They consisted of clicking, double clicking and dragging actions with one or more fingers.

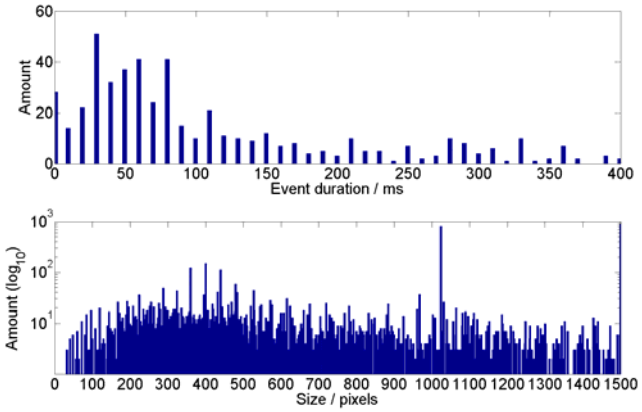

Figure 8: The touch events caused by typical finger touches. Please note that only the bottom histogram has a logarithmic y-scale.

Comparing the diagrams from Figure 7 and Figure 8, it becomes obvious that both – Kinect "touches" and real touch events – have the same duration and thus cannot be distinguished in this manner.

Another obvious approach would be to filter out touch events regarding their size. As it can be seen from Figure 7, most of the touch events are triggered by dots of 32 x 32 (1024) pixels in size. However, such a simple size filtering is not feasible, because many 'real' touch events from human interaction generate inputs of the same size (see Figure 8).

More complex filtering approaches (e.g. based on "growth" of the touch resulting from increasing touch pressure) could not be applied either because some 'real' and some Kinect touches had the same size for the entire event duration, which made them essentially indistinguishable.

Within a research work by Butler et al. [23], the interference by multiple overlapping Kinect patterns was reduced by shaking the Kinects. A similar approach was realized in our setup, hoping that due to the Kinect's moving dots on PixelSense's sensors, the sensors' exposure time would be too slow and thus no event would be triggered. However, shaking the Kinect did not deliver the expected results, as shown in Figure 9.

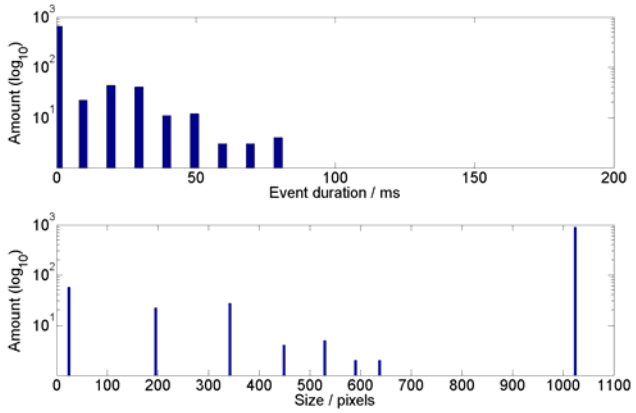

Figure 9: The touch events triggered by the shaking Kinect.

The data in Figure 9 was acquired in a 120-seconds time interval. It shows that the most prominent size of 32 x 32 pixels remains but the duration of all touches is below 100 ms. However, touch events are generally more frequent and still indistinguishable from 'real' touches based on size and duration. Thus, shaking is no feasible solution to reduce the amount of misinterpretations.

Within another approach, we take benefit from the fact that Kinect and PixelSense have different sensitivity levels for IR light. Since Kinect also has to detect reflected dot patterns in 3 m distance, its sensor is more sensitive than the ones from PixelSense. This means, if the Kinect's IR projector intensity could be reduced, the projected dots' intensities would be below the detection threshold of PixelSense, while Kinect could still detect objects in shorter distance, which is the case in our setup (see Figure 3).

Since it was the goal to avoid any internal modification of Kinect or PixelSense, only external optical IR-attenuators were evaluated. Several optical filters, such as diffusion films, LC-matrices, and Plexiglas were tested. They either provided insignificant attenuation, deformed the dot pattern structure, or blocked the IR light completely. The only good results were achieved with a linear polarization filter for visible light, which was attached in front of Kinect's IR projector (see Figure 10).

Figure 10: A linear polarization filter is used to attenuate the Kinect's IR-light. A piece of cardboard holds it in place.

The linear polarization filter successfully reduced the intensity without detrimental effects on depth data. Furthermore, the filter allowed some fine tuning of the attenuation by turning it, since Kinect's IR light seems to be polarized already.

In addition to the correct orientation of the filter in front of the IR projector, Kinect and PixelSense must also be aligned correctly. Since the sensors in PixelSense are behind the linear polarization filter of the screen, they are also sensitive to the polarization of the incoming IR-light. If the long side of the Kinect is parallel to the long side of PixelSense, the attenuation is maximal (see Figure 11).

Figure 11: The long sides of Kinect and PixelSense have to be aligned for the filter to work correctly.

With the filter position set to cause best possible attenuation, measurements were taken again regarding touch events triggered by the Kinect. For a 15-minutes time interval, the following results were achieved (see Figure 12).

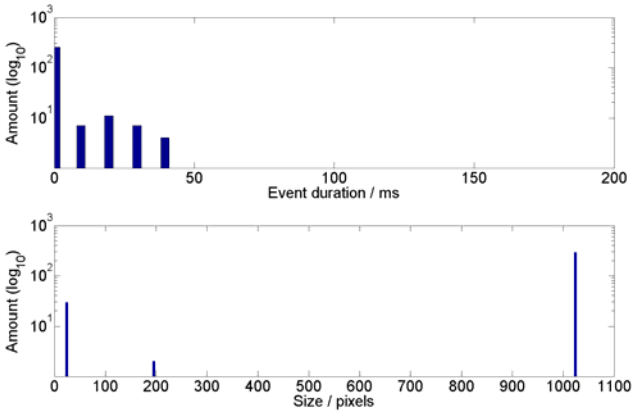

Figure 12: Kinect-triggered touch events with the attenuated IR-projector.

Within 15 minutes, there was only a total of 281 triggered touch events, which corresponds to 18 touch events per minute. These touch events only occurred when the fluorescent ambient illumination in the lab was still switched on, which resulted in increased total illumination of the sensors (see Figure 13).

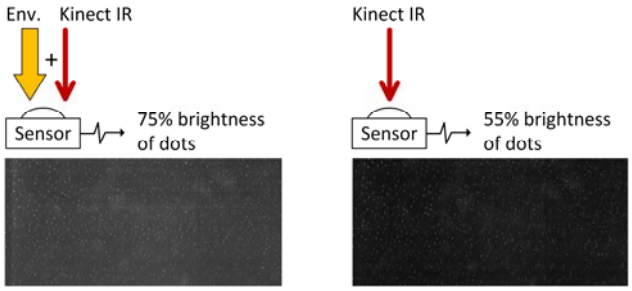

Figure 13: With the room lights switched on, the PixelSense's sensors receive a higher amount of IR.

When switching off the room illumination or by using LED-based light sources, no touch events were triggered anymore by the Kinect. Alternatively, the PixelSense's sensitivity can be recalibrated (but is undesirable because it decreases responsiveness).

Within a next step, it had to be verified whether objects in maximum interaction distance to the Kinect – which is on the tabletop – could still be detected reliably. For this, a thin paperback book was placed on the PixelSense's screen and the visual as well as the IR-image were captured (see Figure 14).

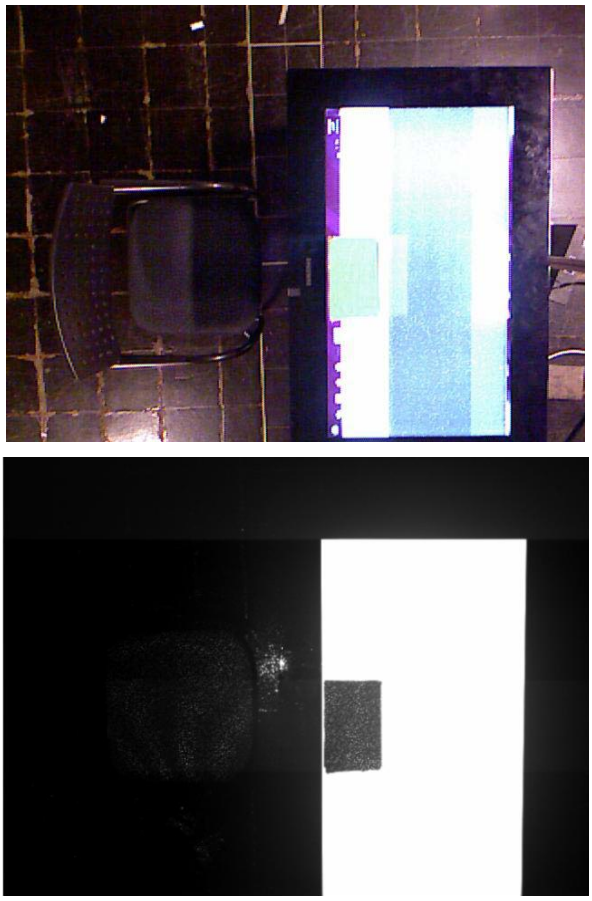

Figure 14: Top: The RGB image of Kinect looking at the PixelSense. Bottom: Corresponding IR-image. The IR-dots can be clearly seen on the booklet.

With this setup, the depth data was recorded for 100 frames with the following four settings:

- Filter off, room lights on
- Filter off, room lights off
- Filter on, room lights on
- Filter on, room lights off

The depth values were examined near all four corners of the booklet. For each of the above conditions, 100 frames were recorded (see Figure 15).

As it can be seen from Figure 15, the Kinect can still reliably track objects on the table, while PixelSense keeps fully operational. Fluctuations in the depth data between the four different conditions can be attributed to normal fluctuations inherent in the system.

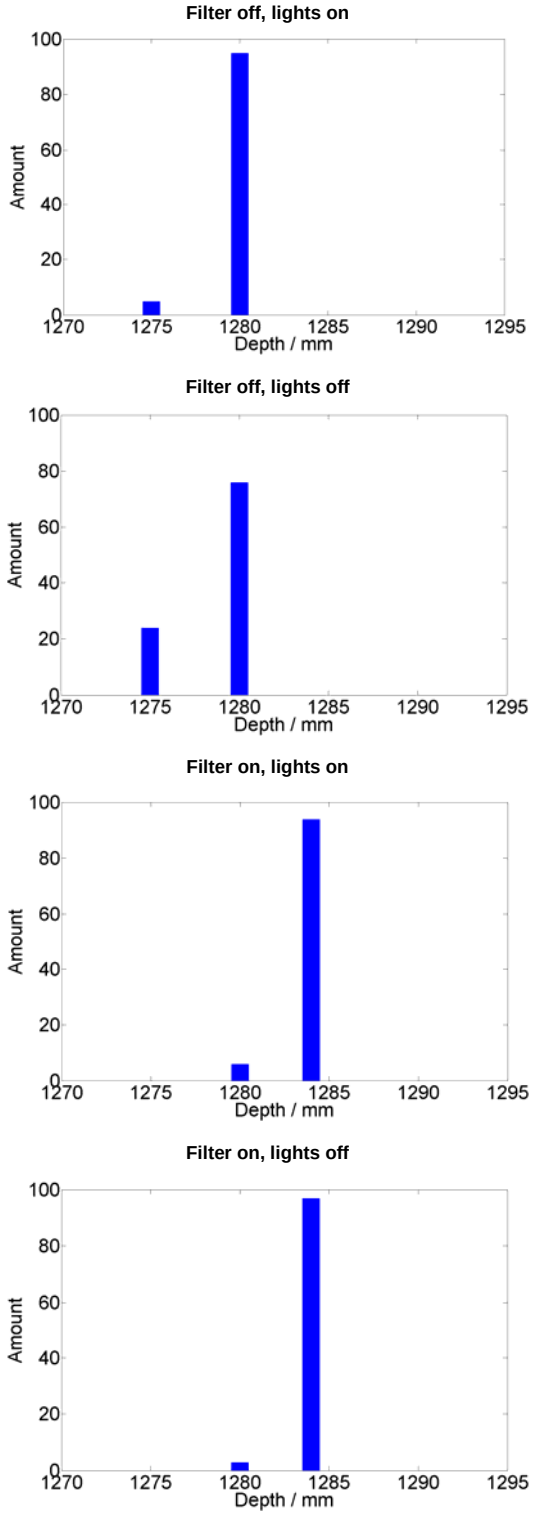

Figure 15: Histograms of the depth values for 100 frames for one corner. The differences when the filter is on and off can be attributed to normal fluctuations of the Kinect.

Lastly the effect of the filter on hand detection above the PixelSense was tested (Figure 16). Detection was largely unaffected. The finger detection is only slightly worse compared to the results without the filter. Thus, it is also possible now to detect deictic gestures in the communication space, which makes the whole setup consisting of Kinect and PixelSense suitable for net-based brainstorming sessions in small workgroups.

Figure 16: Ten consecutive frames of Kinect depth data similar to Figure 4 but with the filter. The arm and hand are reliably detected, fingers are not.

4 REPRESENTATION OF POINTING GESTURES

With the depth data available, the hand orientation, relative to the PixelSense screen, can be determined. Thus, calculating the intersection of the pointing gesture's vector with the screen is possible. We use this intersection point to calculate the nearest possible object in the task space. As a visual feedback, this object starts to be highlighted on the screen. The initiator of the gesture can correct its target by changing the pointing direction. If a user points on a target for more than 1 second, the object is fully highlighted, and the information regarding the pointing gesture is also transmitted to the remote location, where the target object gets highlighted also.

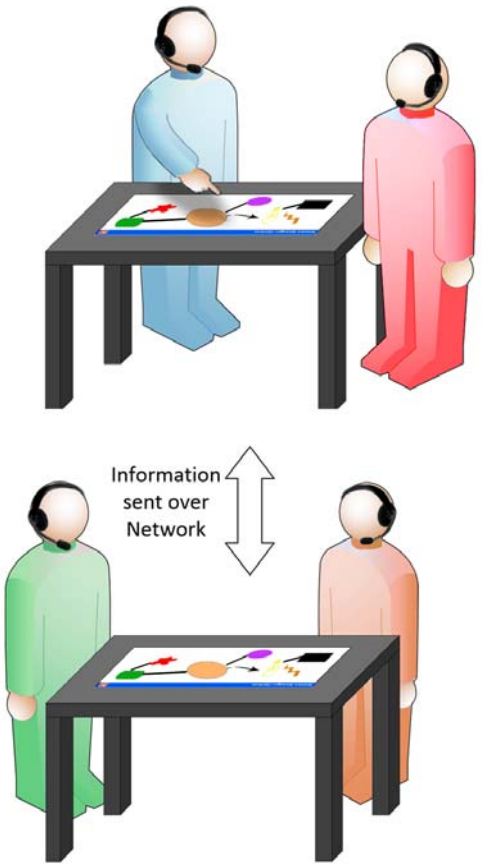

Figure 17: Representation of pointing gestures in a net-based collaboration: The pointing gesture is detected at one side and the corresponding target is highlighted on the other side.

5 CONCLUSION AND FUTURE WORK

This paper showed how a Kinect depth sensing camera could be used for tracking gestures in the communication space above the PixelSense tabletop computer. The main technical challenge, i.e. the interference between two devices' infrared sensors and emitters, is solved using a linear polarization filter in front of the Kinect's IR-emitter. Using this technique, the interference on the PixelSense could be avoided while the Kinect still can detect any gesture above the table. Thus, no electrical modifications of the devices are necessary, which allows an easy adaptation of the existing equipment. Moreover, the measurements showed that Kinect-generated fake touch events could be completely eliminated, if an LED room illumination was used.

Future work will focus on improving the detection quality of gestures in the workspace by improving the current technology as well as employing other technologies such as IR-stereovision or IR-shadow casting.

6 ACKNOWLEDGMENTS

This work was part of the research project "Computer Support for Brainstorming Meetings with Blind and Seeing Persons" and funded by the Swiss National Science Foundation (SNF).

7 REFERENCES

[1] Sutton, R. I.; Hargadon, A. (1996): "Brainstorming Groups in Context: Effectiveness in a Product Design Firm", Administrative Science Quarterly, Vol. 41, No. 4, p. 685-718.

[2] Osborn, A.F.: "Applied imagination: Principles and procedures of creative problem solving (Third Revised Edition)"; Charles Scribner's Sons.

[3] Michinov, N. (2012): "Is Electronic Brainstorming or Brainwriting the Best Way to Improve Creative Performance in Groups? An Overlooked Comparison of Two Idea-Generation Techniques", Journal of Applied Social Psychology 42.S1, p. E222-E243.

[4] Mehrabian, A.; Wiener, M. (1967). "Decoding of Inconsistent Communications". Journal of Personality and Social Psychology 6 (1): p. 109–114.

[5] Walther, J. B.; Anderson, J. F.; Park, D. W.: "Interpersonal effects in computer-mediated interaction a meta-analysis of social and antisocial communication" Communication Research 21.4 (1994): 460-487.

[6] Ishii, H.; Kobayashi, M. (1992): "Clearboard: A Seamless Medium for Shared Drawing and Conversation with Eye Contact"; in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '92; p. 525 – 532;

[7] Krueger, M.W. (1983): "Artificial Reality"; Addison Wesley.

[8] Tang, J.; Minneman, S. (1990): "VideoDraw: A Video Interface for Collaborative Drawing"; in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems Empowering People; p. 313 – 320.

[9] Tang, J.; Minneman, S. (1991): "VideoWhiteboard: Video Shadows to Support Remote Collaboration"; in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Reaching Through Technology; p. 315 – 322.

[10] Bly, S. A. (1988): "A Use of Drawing Surfaces in Different Collaborative Settings"; in: Proceedings of CSCW; p. 250 – 258; ACM Press.

[11] Bly, S. A.; Minneman, S. L. (1990): "Commune: A Shared Drawing Surface"; in: Proceedings of OIS; p. 184 – 192; ACM Press.

[12] Kirk, D.; Stanton Fraser, D. (2006): "Comparing Remote Gesture Technologies for Supporting Collaborative Physical Tasks"; in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '06; p. 1191 – 1200.

[13] Kirk, D.; Rodden, T.; Fraser, D. (2007): "Turn it this way: grounding collaborative action with remote gestures"; in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '07; p. 1039 – 1048.

[14] Stotts, D.; Smith, J.; Jen, D. (2003): "The Vis-a-Vid Transparent Video Facetop"; in: Proceedings of the UIST 2003; p. 57 – 58; ACM Press.

[15] Wellner, P. (1993): "Interacting with Paper on the Digital Desk"; in: Communications of the ACM 36(7); p. 87 – 96.

[16] Wellner, P.; Freeman, S. (1993): "The Double Digital Desk: Shared Editing of Paper Documents"; XEROX Euro PARC Technical Report EPC-93-108; Cambridge/UK; Xerox Corporation.

[17] Kuzuoka, H.; Yamashita, J.; Yamazaki, K.; Yamazaki, A. (1999): "Agora: A Remote Collaboration System that Enables Mutual Monitoring"; in: Proceedings of CHI 1999; p. 190 – 191; ACM Press.

[18] Tang, A.; Neustaedter, C.; Greenberg, S. (2006): "VideoArms: Embodiments for Mixed Presence Groupware"; in: Proceedings of HCI 2006; p. 85 – 102; ACM Press.

[19] Tang, A.; Neustaedter, C.; Greenberg, S. (2004): "VideoArms: Supporting Remote Embodiment in Groupware"; in: Video Proceedings CSCW 2004; ACM Press.

[20] Kunz, A.; Nescher, T.; Küchler, M. (2010): "CollaBoard: A Novel Interactive Whiteboard for Remote Collaboration with People on Content"; in: Proceedings of the 2010 International Conference on Cyberworlds, CW 2010; p. 430 – 437.

[21] Nescher, T.; Kunz, A. (2011): "An interactive whiteboard for immersive telecollaboration"; in: The Visual Computer: International Journal of Computer Graphics, Vol. 27, No. 4, p. 311 – 320.

[22] Microsoft PixelSense; http://www.microsoft.com/en-us/PixelSense/default.aspx; accessed 23.10.2013

[23] Butler, A.; Izadi, S.; Hilliges, O.; Molyneaux, D.; Hodges, S.; Kim, D. (2012): "Shake'n'Sense: Reducing Interference for Overlapping Structured Light Depth Cameras"; in: Proceedings of CHI 2012; p. 1933 – 1936; ACM.