

Advanced Interaction Techniques For Remote Collaboration Systems Using An RGBD Camera

BARBARA J. SCHWANKL



MASTERARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Interactive Media

in Hagenberg

im September 2015

© Copyright 2015 Barbara J. Schwankl

This work is published under the conditions of the *Creative Commons License Attribution–NonCommercial–NoDerivatives* (CC BY-NC-ND)—see <http://creativecommons.org/licenses/by-nc-nd/3.0/>.

Declaration

I hereby declare and confirm that this thesis is entirely the result of my own original work. Where other sources of information have been used, they have been indicated as such and properly acknowledged. I further declare that this or similar work has not been submitted for credit elsewhere.

Hagenberg, September 18, 2015

Barbara J. Schwankl

Contents

Declaration	iii
Vorwort	vi
Kurzfassung	vii
Abstract	viii
1 Introduction	1
1.1 Motivation	1
1.2 Outline	3
2 Related work	5
2.1 State of the art	5
2.2 Analysis of existing work	10
3 Methodology	13
3.1 Concept	13
3.2 Setup	13
3.2.1 User caption with the MS Kinect v2 sensor	14
3.3 Remote visualization	18
4 Prototype implementation	21
4.1 Pipeline	21
4.1.1 Data acquisition	21
4.1.2 Frame transformation	24
4.1.3 Rendering	26
4.2 Implementation details	26
4.2.1 Body detection	27
4.2.2 Arm transformation	31
4.2.3 Determination of accurate boundary vectors	36
4.2.4 Overlapping body parts	37
4.3 Limitations	38

5 Conclusion	40
5.1 Future work	40
A System requirements	42
References	43
Literature	43
Online sources	45

Vorwort

platz für danksagungen

Kurzfassung

Diese Arbeit beschäftigt sich mit einem Kollaborationssystem, welches Videokonferenzsysteme ergänzt und bei welchem ein digitales Whiteboard zur Interaktion, und eine RGBD Kamera als Sensor verwendet werden. Ziel ist es, ein System zur Verfügung zu stellen, bei welchem der Gesprächspartner scheinbar hinter dem transparenten Whiteboard steht. Dadurch kann direkt und persönlich mit dem Gesprächspartner kommuniziert werden, wobei zusätzlich zur Sprache auch Mimik und Gestik zu besserer Zusammenarbeit und einer erfolgreichen Kommunikation beitragen. Diese zusätzlichen Eindrücke auf der Meta-Ebene ermöglichen den Aufbau einer tieferen Verbundenheit zum Kommunikationspartner als es über eine reine auf Ton basierende Kommunikation möglich wäre.

Für die Visualisierung des Gesprächspartners wurden drei verschiedene Varianten in einem Prototypen verwirklicht, um die perspektivischen Verzerrungen, welche durch die Positionierung der Kamera verursacht werden, zu kompensieren. Zusätzlich wurden effiziente Algorithmen verwendet, um das System in einem herkömmlichen Büro ohne zusätzliche Aufwände einsetzen zu können.

Abstract

This work presents an enhanced system supporting remote collaboration using a digital whiteboard and an RGBD camera. Showing the remote partner behind the transparent collaboration surface imitates a face-to-face meeting, revealing significant mimics, gestures and interactivity of the partner. Thus, a personalized feeling for the users is created. Three different gesture visualization scenarios are introduced in order to compensate for the perspective distortion that is due to the camera positioning. Several seed fill algorithms are compared to increase the performance of the system, considering the amount of data that has to be processed to render the motion images. A prototype has been developed to investigate the relevance and adaptability of the system within a workaday life.

Chapter 1

Introduction

Teleconferencing technologies enable users to work collaboratively without the need to share the same room. Collaborators may come from different cities or different countries, have a different mother-language and a different cultural background. Therefore, it is important that remote collaboration concepts set an increased focus on communication and interaction. Video supported collaboration offers multiple advantages as the user can transport information through gestures and mimics without interrupting the currently talking user [8]. Previous studies showed that in video conferencing systems, body language availability and the field-of-view of the transmitted data influences the users' communicative attitude [16, 17]. The subjective feeling of dominance within a conversation depends on the users' gender. Men feel more dominant using body language while women feel more dominant with their body language not being transmitted. Furthermore, eye-contact and gaze awareness are essential to conceive meta-information [3]. The conceived distance to the communication partner influences how comfortable the participants feel. Hall [4] describes these so-called proxemics in his work. Digital collaboration tools, such as interactive whiteboards, can break the barriers of proxemics [20].

The main aim of this thesis is to find a way to imitate face-to-face collaboration for remote participants and enhance gesture visualizations with a simple hardware setup.

1.1 Motivation

The ideal scenario for a remote collaboration tool is a digital screen on which the collaborative content is shared and edited by all users simultaneously. On the screen and behind the shared content, the remote collaborator is seen like through a transparent surface. Figure 1.1 shows a setup that endorses this vision. However, there are several drawbacks linked to that approach. The camera that is placed behind the screen captures the user and the shared

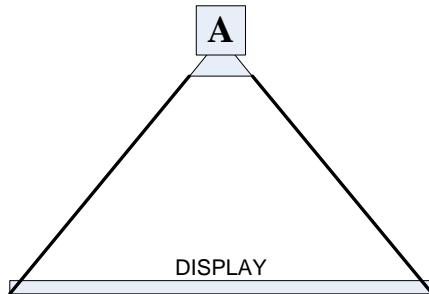


Figure 1.1: The scenario shows a camera behind a transparent collaborative wall that captures the user from the front. In this way, a face-to-face communication is imitated.

content. Written content appears mirrored and would have to be excluded from the video stream first, then mirrored, and subsequently combined with the remote user's image again.

Furthermore, this setup requires an unreasonable amount of space for conventional offices. The screen has to be set up in the middle of the room to leave enough space to place the camera behind. Thereby, a suitable distance from the camera to the screen is required to be able to capture the user when standing still as well as on moving.

A prototype was created that surpasses the weaknesses of the approaches presented in Chapter 2. The demands for the system are

- (global) availability,
- affordability,
- maximum visible user projection, and
- transferability.

A fundamental idea behind elaborated remote collaboration is availability of the system for all participating users, disregarding their business site location and their financial capabilities. Remote collaboration often includes people from foreign countries, who may encounter difficulties to access to the latest technology. Furthermore, a maximum of the remote user's appearance with a focus on the mimics and gestures should be perceived. Additionally, the system should be applicable to different people, not being customized for one person, but rather be suitable for multiple users with different profiles, habits and different body heights.

Multiuser support is an essential requirement for collaborative environments. The prospective system requires support for multiple users within a conference, one user per system setup. All collaborators should use a common surface to sketch, draw and write down their desired content. Indicating each user with a personalized mouse cursor is often not sufficient to quickly link the writing user with the written content, especially if several users pro-

duce content at the same time. Therefore, a focus shall be put on the linkage between user and content. Transmitting the whole body of a user including the writing hand helps to understand the relation between user and writing. Opposed to mouse cursors, projecting users behind the collaborative surface imitates a face-to-face communication and creates an interpersonal relationship. Furthermore, using human body parts to point helps observing the remote collaborator and the shared content simultaneously. Additionally, the content may be emphasized through the remote user's mimics and gestures.

The system is targeted at conventional offices, adapting to the environmental conditions. It should possess

- a screen that is able to detect touch input,
- an RGBD camera to get synchronized color and depth information, and
- an existing extensible software framework to extract data out of the sensor.

Market investigation narrowed down the possible systems to

- a large touch display (such as the Microsoft Surface Hub [25]), or
- a projection screen with pen pairing (such as the CASIO interactive whiteboard [21] or the we-inspire systems [22]), combined with
- a Microsoft Kinect v2 sensor.

The touch displays and projection screens with pen pairing deliver reliable information about the touch coordinates on the wall. The Kinect sensor meets all desired requirements for the system, such as

- a given software development kit (SDK),
- a depth sensor to recognize the collaborator,
- fully developed hardware,
- maintained framework,
- global availability, and
- a manageable size.

1.2 Outline

This thesis covers all stages within a remote collaboration pipeline and presents the used algorithms in detail. Chapter 2 discusses related work and state of the art technology. Chapter 3 introduces the concept and the setup for a prototype aiming to enhance remote collaboration, and all components of the used RGBD sensor and their limitations are explained. The remote collaboration pipeline is covered in chapter 4. The data acquisition, frame transformation and rendering stages are described in all its detail and

the used algorithms for the prototype are elaborated. Furthermore, problems and limitations within the pipeline are discussed. Finally, future work and prospective enhancements on the prototype are outlined in chapter 5.

Chapter 2

Related work

2.1 State of the art

Tan et al. [15] present *ConnectBoard*, a see-through display with a camera and a projector mounted behind the screen respectively. The screen shows the shared media and the remote user in the background while the camera captures the local user (see Figure 1). Subsequently, the image of the local user is transmitted to the remote user. Genuine eye-contact is achieved using face detectors by automatically shifting the image of the remote user onto the path between the local user and the camera. In this way, eye-contact is always given between two users even in case of movement.

This approach is similar to the work of Ishii and Kobayashi [9] who introduced *ClearBoard*. They also used a transparent screen with a projector and a camera in the back to provide gaze awareness.

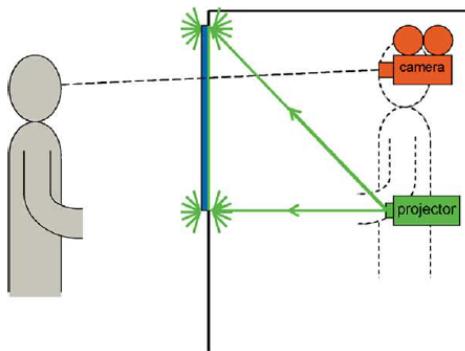


Figure 2.1: The setup of *ConnectBoard* [15]. The projector (green) projects the assembled image of the remote user with the shared content onto the screen. The camera (red) captures the local user and transmits the image back to the remote user. The transmitted image is mirrored for correct presentation.



Figure 2.2: *Dynamo* offers the users a platform to share and exchange content [10]. Furthermore, users can decide themselves what kind of content to share with whom.

Izadi et al. [10] introduced a collaborative system that enables users to share and exchange information in public spaces. Their approach requires large displays and commonly accessible mice and keyboards to interact with the system. Data is transferred via USB flash drives. Each flash drive is mapped to a user that gains an individual space on the public surface. Users exchange data by simply dragging a file onto the other users' avatar (see Figure 2.2).

IllumiRoom and *RoomAlive* are proof-of-concept prototypes developed by Jones et al. [12, 13] that augment parts of rooms. An ultra-wide field of view projector is used to display content and a depth camera is used to recognize object structures. According to the different structures, content is rendered differently (see Figure 2.3). *RoomAlive* tracks the bodies of the participants and dynamically adapts the content according to their behavior. Intentionally designed for gaming, these systems can also be adopted to virtual conferences.

Similar to these approaches, Kasahara et al. [14] present *Second surface*, an environment that allows users to create augmented collaborative content mapped to real places with mobile devices.

The *ViiBoard* [2] is a remote collaboration tool with a simple setup, using only a large touch display and an RGBD camera that is mounted next to it. The 2D mode shows the shared content next to the captured remote user. Whenever the remote user modifies the shared content, the writing hand gets scaled on the display (see Figure 2.4). In this way, occlusion of the



Figure 2.3: (a) The scene before activation of the *IllumiRoom* system [12].
(b) The *IllumiRoom* system increases the contrast of the scene through projecting color on the objects of the environment.

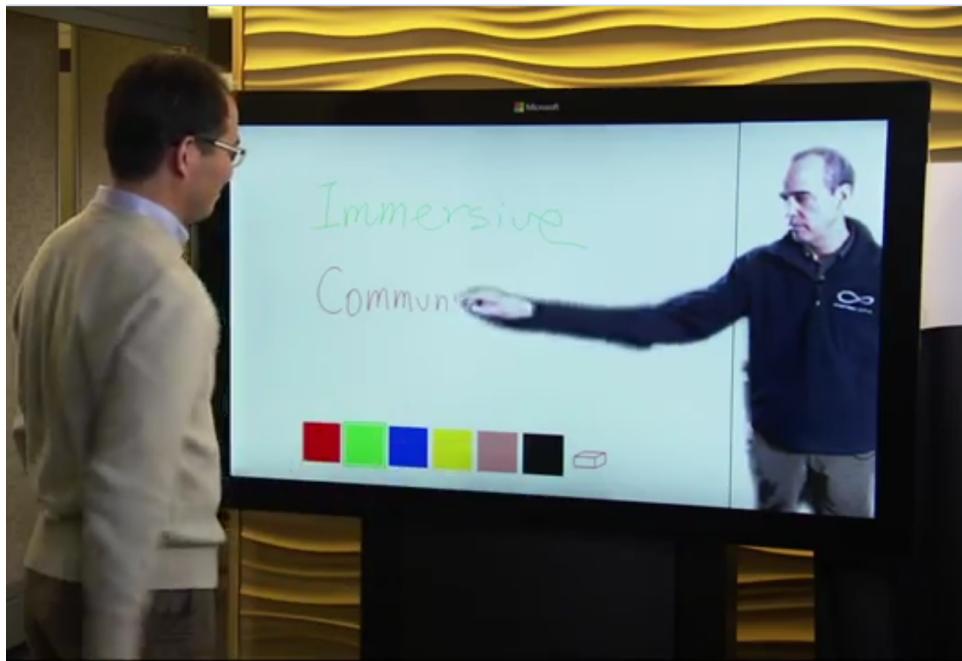


Figure 2.4: The *ViiBoard* transmits a simple RGB-image of the remote user [2]. On editing the content, the writing hand of the remote user gets scaled and follows the writing. This technique facilitates following the context and avoids interference of the collaborators.

content by the remote user is avoided while information about the current writing activities of the remote user is preserved.

ImmerseBoard [6] is the successor of the *ViiBoard*. It is a system for remote collaborate work based on a simple HW setup. All that is required is a Kinect v2 camera and a big touch display. The camera for the first

participant is positioned on the right-hand-side of the screen; the camera of the second participant is positioned on the left-hand-side of the screen. This setup is required for a more realistic visualization of the remote participant. The system allows the user to choose between four different modes. The *video mode* shows the remote participant in the side of the screen where the camera is positioned. So the participants almost have eye contact when they look into the camera. On the video the background is removed the remote participant is cropped and positioned. The *hybrid mode* is an enhancement of the video mode. The arm which is used for the interaction with the digital whiteboard is stretched if it is within a certain distance to the board. When the board is touched the hand points exactly to the touch point. This makes it easier to follow the remote collaborator when he is interacting with the whiteboard. The *tilt board mode* shows the full sized remote collaborator. This requires more space on the display compared to the “hybrid” mode and that’s why the virtual board needs to be tilted. Apart from that the concept is the same as for the “hybrid” mode. The computation of the arm stretch has to consider the tilt of the board. The remote collaborator is more present but the content of the virtual board has a perspective distortion caused by the tilt. The *ImmerseBoard* allows better expressions through pointing, gaze direction and other gestures, and provides additional information through body posture, proxemics and eye contact. Furthermore, its setup integrates easily in existing offices.

The *mirror mode* simulates the real-world mirror. It seems as the remote collaborator stand next to the local collaborator and he can be seen through the collaborative board which behaves mirror. The remote collaborator is transparent and blended over the content of the board. Because the participants look straight at the board, parts of the face cannot be recorded by the camera. The bodies themselves are reconstructed using the 3d point cloud from the Kinect camera. The quality could only be improved by using additional camera.

Valadares et. al. [18] analyze the IBM collaboration tool *Sametime 3D*. The tool allows a user to control a 3D avatar within a virtual world. Collaboration entirely takes place in the virtual environment, all users having avatars representing themselves. Documents, videos or flip charts can be shared in virtual meeting places (see Figure 2.5). Additionally, the system supports spatial voice and highlights the currently talking user’ avatar. Similar to real world scenarios, users can express their interest by spatial proximity to the speaker. The only relation to the real person is the avatars’ voice. Natural gestures and movement such as walking, hand gestures and head movement can be achieved via mouse and keyboard inputs.

A vision-based system introduced by Izadi et al. [11] provides a combination of bi-manual and tangible interaction. It uses a tablet computer as a shared collaboration interface and a separate chat screen to display the remote user (see Figure 6). A simple webcam captures the user while a



Figure 2.5: Users represented by avatars move within the *Sametime 3D* virtual environment [18]. Additionally, the system supports shared content, spatial voice and avatar gestures.

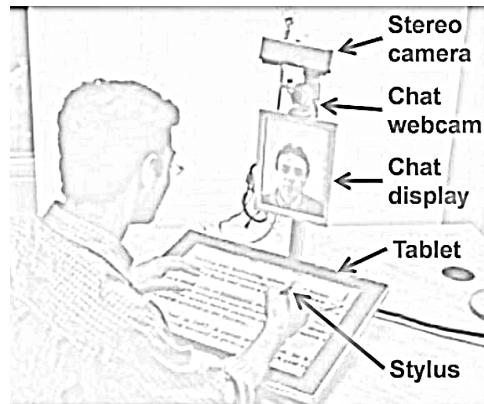


Figure 2.6: The *C-Slate* system setup [11]. The top-down stereo camera captures the tablet surface while the webcam captures the face. The tablet display shows shared and remote content, such as the hands of the remote user. Gaze awareness is provided through the chat display.

top-down stereo camera captures the tablet surface. Virtual hands provide information about the remote users' actions and intentions. The system uses machine learning algorithms to recognize fingertips and touch interaction with the tablet surface. Additionally, physical objects such as pocket

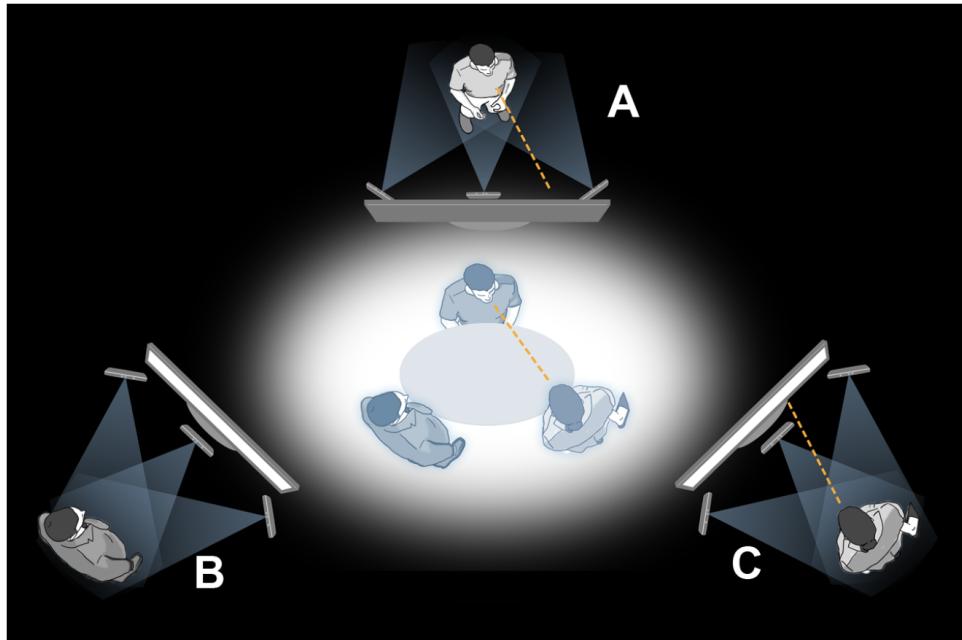


Figure 2.7: The figure shows *Viewport* by Microsoft Research [19]. Each participant A, B and C has a computer monitor as well as three cameras in front. Through eye-tracking, the system recognizes eye-contact between user A and C. Consequently, user B sees only a side view of user A and C.

cameras are recognized and prompt a predefined action.

Zhang et al. [19] developed a teleconference system which imitates a face-to-face conversation. The system supports life-size 3D video, spatial audio and provides a virtual environment that covers furniture, lighting and wall colors. In this way, the illusion of sitting on the same table is created. The virtual conference room is rendered from each users' view-port separately for correct motion parallax. Therefore, an eye-tracking algorithm is implemented (see Figure 2.7). Spatial audio maps the voice of the currently talking user to its location within the virtual space.

2.2 Analysis of existing work

Recent work on collaborative systems differ in

- the remote collaboration ability,
- camera position (behind, on top, in front of, or next to the display),
- the environment (background removed or artificially created),
- proxemics (distance to the remote collaborator),
- content-user-relation (where is the user located in relation to the con-

- tent),
- gaze awareness or eye-contact,
 - gestures,
 - number of users supported (one-to-one or multiuser),
 - the dimension of the environment (2D or 3D), or
 - setup and hardware requirements.

As previously stated, gestures and mimics can influence communication strongly. However, not all presented systems offer access to this meta-information. *Dynamo* has a simple and intuitive setup and presents inventive approaches about information sharing. It does not provide meta-information of the collaborators and does not support remote collaboration.

Likewise, *IllumiRoom* and *RoomAlive* also provide no functionality for remote collaboration. They offer unique techniques that could be used for remote collaboration, such as body tracking, object recognition and touch detection.

ConnectBoard and *ClearBoard* offer sound solutions for remote collaboration. Gestures and gaze awareness are provided. The systems require special hardware for setup and only support one-to-one collaboration. As RGB images are used, the quality of the user segmentation heavily depends on lighting conditions.

ViiBoard, its successor *ImmerseBoard* and *Viewport* segment the users using depth information respectively. They provide both gestures and mimics. Gestures in the *ViiBoard* system are co-linear to the produced content as the hand follows the writing in real-time. Touch screen and RGBD camera are the only hardware requirements. The communication is based on one-to-one collaboration. Proxemics are invalidated through the close distance to the screen, and therefore the remote user. *Viewport* is not concerned about shared content but supports multiuser interaction. The collaborators and the environment are represented in a virtual 3D world. This enables viewport dependent rendering of the environment. The system uses commonly available hardware components.

Sametime 3D imitates real-life conferences in a completely virtual environment. It supports 3D collaboration and gestures. However, the social presence in the environment is very abstract and communication does not feel natural to the users.

C-Slate needs multiple commonly available components for setup and offers only one-to-one communication. Mimics are supported whereas gestures are not available as meta-information. However, actions and intentions of the remote user can easily be retraced on the tablet surface. Additional intelligent computer vision algorithms provide various possibilities for collaboration.

Although the presented approaches offer interesting solutions to collaborative work, none of them provides a system that meets all criteria mentioned

above. Most importantly, eye-contact, gestures and multiuser support for remote collaborative work.

Chapter 3

Methodology

A prototype to enhance remote collaboration was developed. This chapter describes the concept, the setup and the components of the prototype in detail.

3.1 Concept

Many video conferencing tools that support remote collaboration lack a linkage between the gestures of the participants and the content that is shared. For this reason, a prototype has been developed that allows the user a visually enhanced video conference experience by directly connecting the interacting hand to the produced content. In applications with multiple users, this approach allows the participants to quickly associate writings with the correct person. The perspective parallax is compensated for. In this way, the participants do not need to watch for moving mouse cursors and are able to quickly associate the writing with its owner.

A remote collaboration tool with enhanced user visualization was created. The required components are

- a large collaboration screen (wall surface, touch display screen, or similar),
- an input device (such as bluetooth pen) or direct touch input,
- a video conferencing tool (such as Skype[®]), and
- an RGBD sensor to capture the collaborator.

The system needs to be available for all participating collaborators. Figure 3.1 shows an example for the setup of the system.

3.2 Setup

It is assumed that the user is participating in a video conference. All conference participants are connected to an application that allows them to

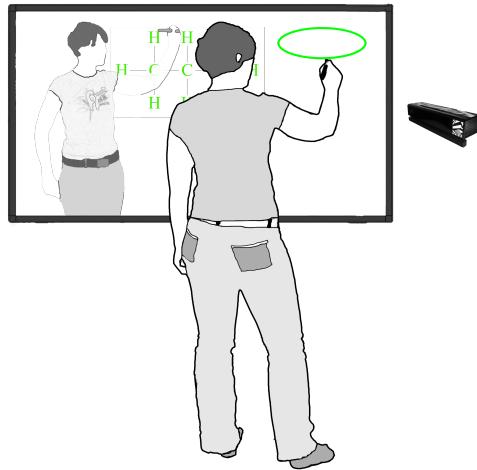


Figure 3.1: The remote collaboration scenario with the sensor placed next to the wall surface. The user and the remote conference participants work together on a collaborative surface.

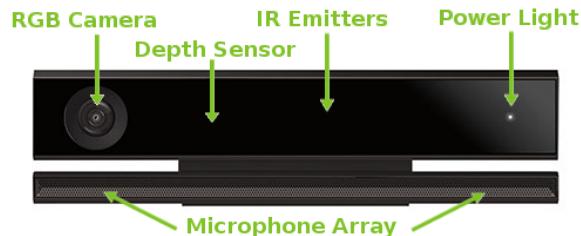


Figure 3.2: The Microsoft Kinect for Windows v2 sensor (source: [27]).

write and draw together on one screen. Each user is captured by a sensor that is placed next to the display wall, facing the user from the right front, the side of the writing hand. The wall recognizes inputs via touch or blue-tooth pen (cf. Fig. 3.6) and passes the touch input coordinates on to the application.

3.2.1 User caption with the MS Kinect v2 sensor

The Microsoft Kinect v2 sensor (cf. Fig. 3.2) offers several image capturing hardware components as well as a SDK for the implementation of custom applications. The RGBD camera has a field of view of 70° horizontally and 60° vertically (cf. Fig. 3.3) [26]. All specifications and the functionality described in this section origin from the Microsoft Developer Network [23], [24]. The system requirements for using the Kinect v2 sensor are identified in Appendix A.

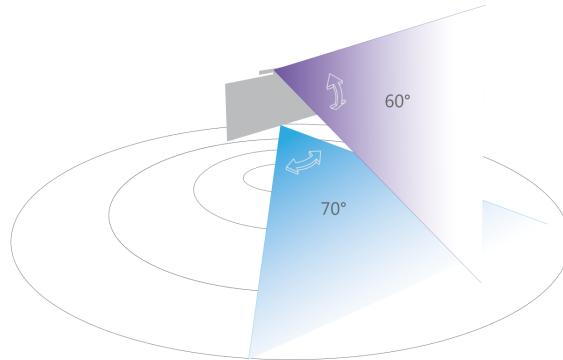


Figure 3.3: The Kinect v2 sensor has a field of view of 70° horizontally and 60° vertically (source: [26]).

The sensor provides image streams (cf. Figure 3.4) for the

- color source,
- infrared source,
- depth source,
- body index source,
- body source, and the
- audio source.

Color source

The color source (cf. Fig. 3.4a) delivers a high-definition video stream with a resolution of 1920×1080 pixel. Based on the lighting condition, 30 or 15 frames per second (fps) are provided.

The pixel values are able to store different color formats such as RGB, RGBA or YUV that are all saved as 8 bit unsigned integer per color channel.

Infrared source

The infrared (IR) source (cf. Fig. 3.4b) provides a grayscale image with a resolution of 512×424 pixel at 30 fps. The pixel values store the IR intensity values as 16 bit unsigned integer. To illuminate the image consistently, the ambient light has been removed. As the common light frequencies are removed, only the high frequencies pass. This guarantees the same base illumination for all lighting conditions. The procedure emphasizes the texture of the image and is therefore suited for facial recognition algorithms, reflective markers tracking and others.

Depth frames and IR frames are perfectly aligned as the image data is derived from the same sensor and therefore has the same resolution. The

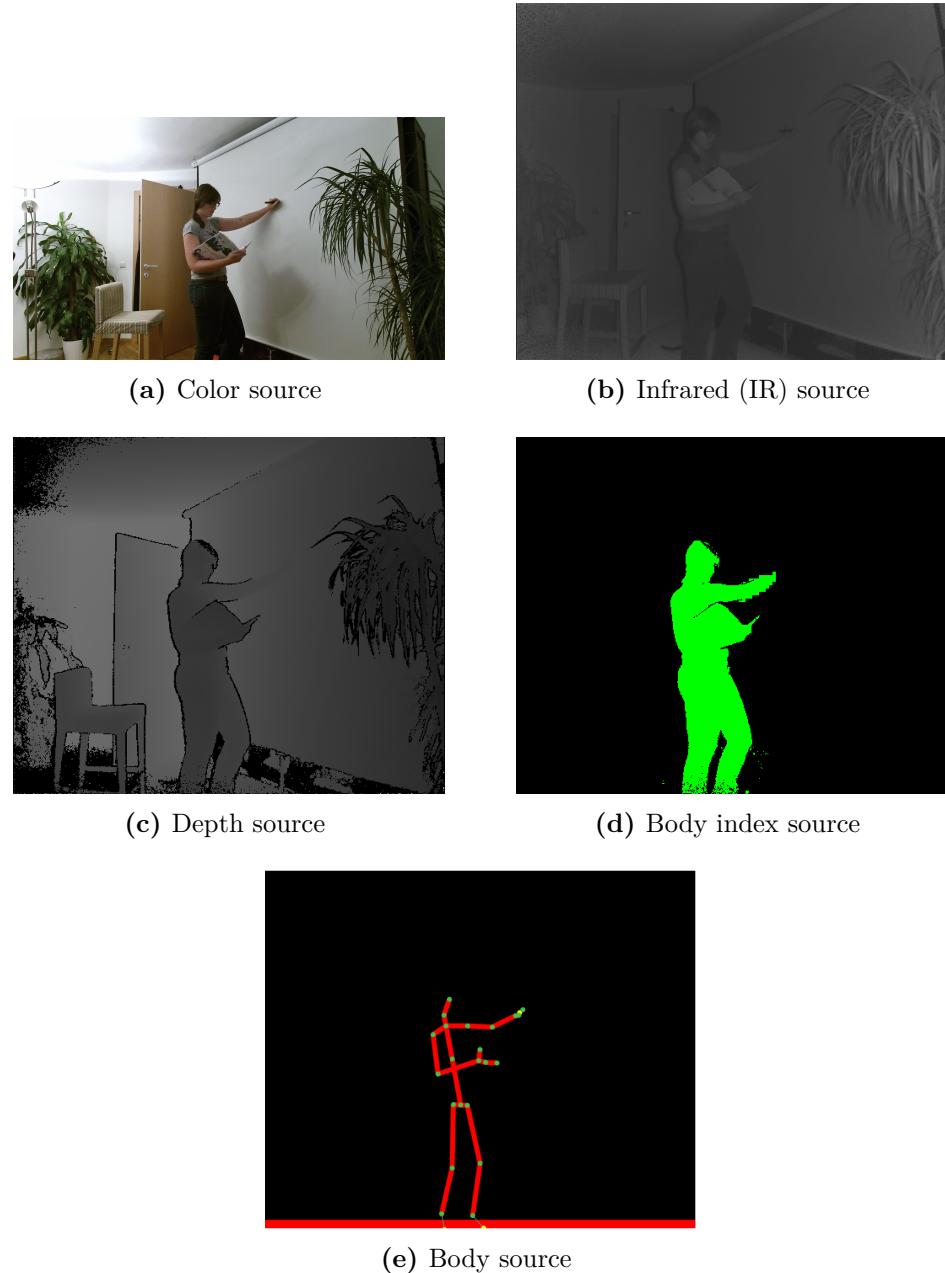


Figure 3.4: The different sources provided by the Microsoft Kinect v2 sensor.

sensor ranges from 0.5 meters to 8 meters, outside that scope no data is delivered.

Depth source

The depth source (cf. Fig. 3.4c) offers the same resolution (512×424) and range (0.5–8 meters) as the IR source, and a framerate of 30 fps. The pixel values are stored as 16 bit unsigned integer and represent the depth data, which is the distance in millimeters from the sensor's focal plane. The different shades of gray within the depth image indicate different depth values. All regions with the same gray level have the same distance from the sensor.

Body index source

The body index source (cf. Fig. 3.4d) indicates a recognized body within the sensor's field of view and is computed based on the depth image. A recognized body's contour is filled with a unique index value to distinguish between body and background pixel. The maximum number of bodies to be recognized is six. The indices dedicated for the bodies range from zero to five, a value greater than five indicates a background pixel. The index is stored as 8 bit unsigned integer in the body index frame.

Body source

The body source (cf. Fig. 3.4e) reveals information about the skeleton of a body (cf. Fig. 3.5) at a framerate of 30 fps. One body consists of 25 known joints, each of them storing information about its position in the 3D space and its orientation. The scope ranges from 0.5 meters to 4.5 meters. The sensor can recognize up to six bodies simultaneously and the hand states (such as closed or open) of two bodies at the same time.

Audio source

The Kinect v2 sensor includes a microphone array with a steerable cone of focus for audio. However, the audio source has not been used for the prototype, as it focuses on enhanced gesture visualizations.

Frame access

The frame sources can be addressed individually via a dedicated frame reader or all sources combined via a multi source frame reader. Individually addressed frames need to be synchronized manually while the multi source frame reader synchronizes the frames automatically and allows the application access to all chosen frames on a single event. Each frame has a descriptor which stores metadata information such as the width and the height of the image stream.

The coordinate mapper is required for mapping the points of the different coordinate systems. These are

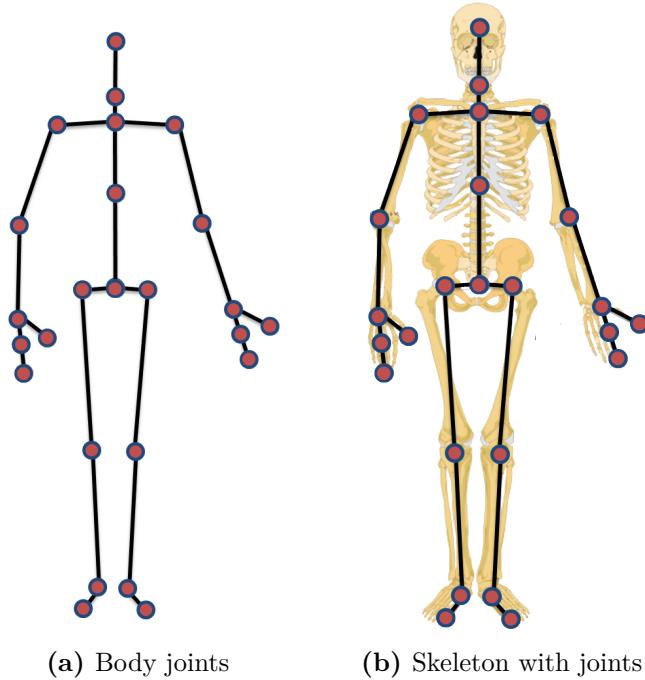


Figure 3.5: The joints provided by the body source (source: [24])

- color space points (originate from color source),
 - depth space points (originate from depth, IR or body index source), and
 - camera space points (originate from body source).

As the coordinate systems provide frames with different resolutions, the coordinate mapper contains a lookup table to find the equivalent pixel positions in the coordinate systems. The mapper stores information about the sensor's intrinsic and extrinsic parameters and therefore makes a manual calibration of the sources redundant.

3.3 Remote visualization

In an ideal environment, the camera would be placed behind the screen to capture the user from the front, but state of the art wall surfaces are not lucent. With the camera positioned at the right front of the collaborative wall (cf. Fig. 3.1), the captured images shows a perspective parallax: All lines meet at one point at infinite depth. As a result, a point on the display screen from the camera's perspective does not match the same point on the display screen. That means when the display receives a touch input (cf. Fig. 3.6), the

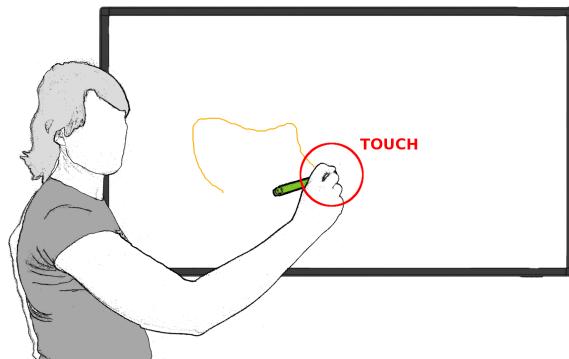


Figure 3.6: When the user touches the whiteboard with the input device, the coordinates of the touch position are used to determine the position of the manipulated arm.

input coordinates do not match the coordinates from the video stream. In order to match the touch input coordinates with the finger tip coordinates from the sensor, the user's arm in the video frame has to be translated to the exact position of the touch input.

The prototype offers three different modes (cf. Fig. 3.7) to the user to support remote collaborative video conferences, these are

- symbol mode,
- hand mode, and
- arm extension mode.

All three modes accompany the users writing and replace the conventional mouse cursor with a video or an image. The symbol mode (cf. Fig. 3.7a) replaces the cursor with an image of choice and does not require any additional camera setup. The hand mode (cf. Fig. 3.7b) and the arm extension mode (cf. Fig. 3.7c) need a sensor to track and capture the user. In this way, the captured image is manipulated by the application and shown to the remote user.

The manipulated frame is linked to the user and the user's gestures are visible at all time. In this way, it is easier for the collaborator to perceive at what point in time and on what region the remote user starts writing or drawing onto the collaboration surface.

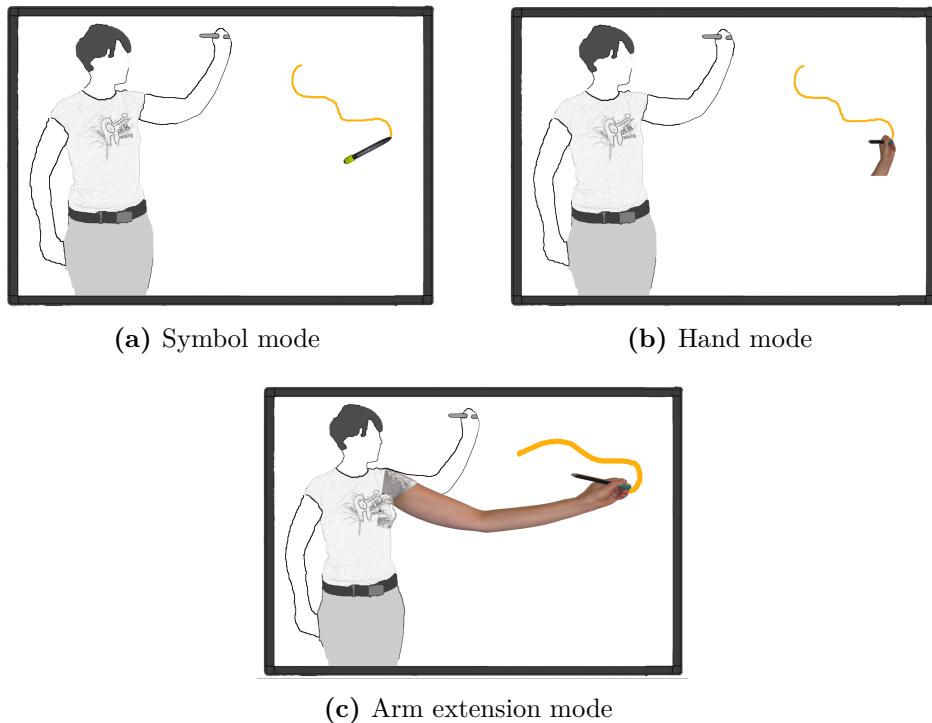


Figure 3.7: The different modes to emphasize the cursor of the remote user. The symbol mode (a) replaces the cursor with an image of choice. The hand mode (b) duplicates the right hand of the user and copies it to the touch input position. The arm extension mode (c) provides a manipulated arm that is scaled and rotated to the exact touch input position.

Chapter 4

Prototype implementation

A prototype has been created to investigate the adaptability of transformed motion images in video conferencing software. A Microsoft Kinect v2 sensor was used to capture the image data.

This chapter gives an overview of the computation pipeline (cf. Sec. 4.1) and explains the implemented scenarios and used algorithms (cf. Sec. 4.2) in detail. The user can choose between

- a symbol pointer that follows the input device (cf. Fig. 3.7a),
- a representation of the hand that is used as a cursor and follows the pointing device (cf. Fig. 3.7b), and
- a mode to scale and rotate the lower arm to the input point (cf. Fig. 3.7c).

4.1 Pipeline

The application aims to transform the video source in a way that the background is eliminated and the collaborator is extracted. The user's gestures accompany the writing on the collaborative surface. Therefore, the data passes different stages of computation (cf. Figure 4.1):

1. Acquisition of data
2. Transformation of the frame regarding the selected scenario
3. Rendering the transformed frame on the collaborative surface

These stages are explained in detail below.

4.1.1 Data acquisition

A video stream that is acquired directly from the sensors shows the user and its entire environment within the camera's viewport (cf. Fig. 3.4a). To manipulate the image, all sources described in Chapter 3 are used for the computation, except for the audio source that is not required for the prototype implementation, and the IR source as the depth source that is

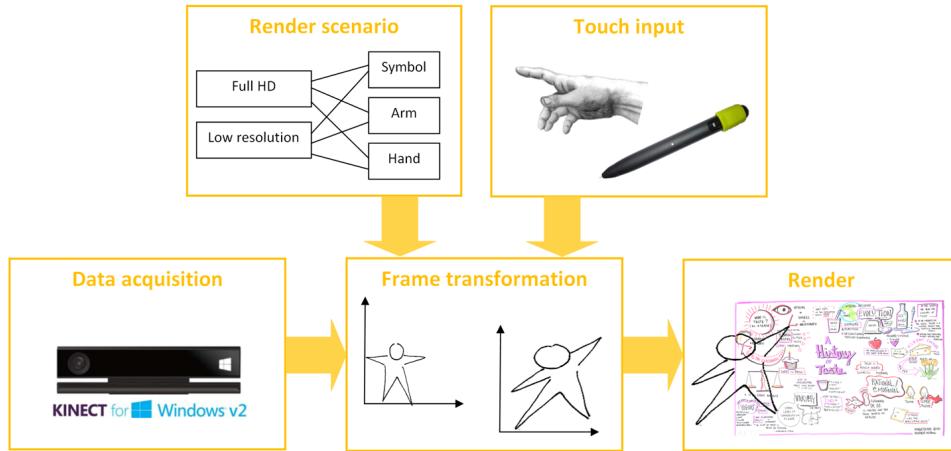


Figure 4.1: The image manipulation pipeline.

derived from IR delivers all required information. The following sources are needed for further processing.

- Color source
- Depth source
- Body index source
- Body source

All three video manipulation scenarios defined above require a colored output, therefore the color source is needed. Information about the silhouette of the user is stored in the body index source, which is useful for extracting the background from the user within the color image. The color source and the body index source have different resolutions and origins – a coordinate mapper is required for mapping the points of one space into another. The mapper requires the depth source for operating.

For transforming particular parts of the body, the body source is needed as well. The sensor offers access to so called *body joints* that indicate 25 different body parts (cf. Sec. 3.2.1).

Composition of components

Figure 4.2 shows the sensor positioned in various angles to the display in order to compare the different camera views of the user. The camera positions are compared as follows:

- **Camera B** and **camera C** are positioned above the display in front of the user. Camera C, that is placed on the upper edge of the display, has a good view of the user and captures gestures and mimics respectively. However, the near plane of the camera has to be considered when the user interacts with the upper region of the display. In this case, all body

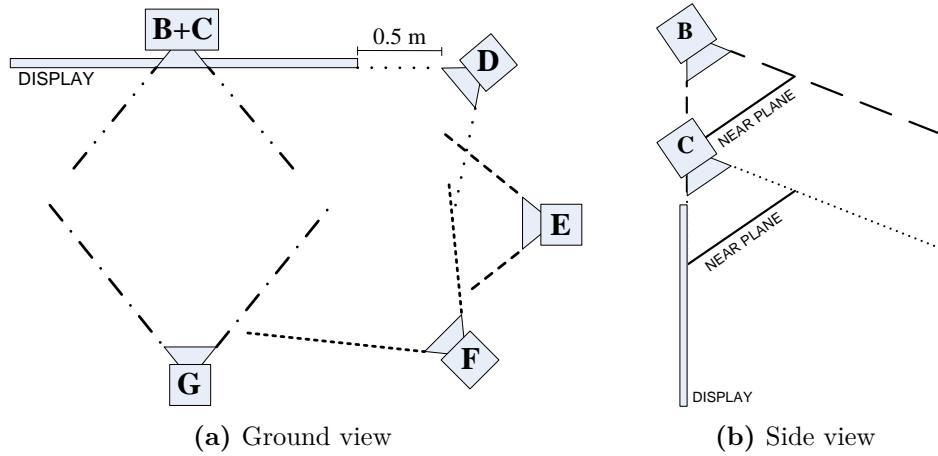


Figure 4.2: Different setups of the Kinect camera. The field of view is 70° horizontally and 60° vertically.

parts that are closer to the camera than the near plane are discarded. In contrast, camera B does not interfere with the user. Due to the increased distance, the user's face and mimics are hardly visible and the view of the user is perspectively distorted.

- **Camera D** shows the user from front right. Most of the gestures and mimics are visible, and the skeleton is correctly detected. In addition, this position shows much of the environment. According to the shift to the display, perspective distortions occur compared to a view from camera A (cf. Fig. 1.1) or G.
- **Camera E** and **camera F** show a side and rear side view of the user. The sensor is not able to identify all limbs of the body correctly as they are occluded.
- **Camera G** shows the user from the rear, opposite to the display. It correctly detects the skeleton, but it loses track of the hands and arms when the user is writing on the display. Opposed to camera C and E, no perspective distortions occur. However, this setup does not allow the camera to capture the user's mimics.

Considering the advantages and disadvantages of the camera positions described above, the scenario of camera D is preferred as it shows the most natural view of the user. In order to adjust the perspective distortion, algorithmic solutions are examined in chapter 4.

4.1.2 Frame transformation

This stage prepares the incoming data from the sensor for the remote transmission. The user can set the preferred parameters in the GUI.

Input parameters

The user can choose between two video qualities and three different representations of the virtual conference partner. Concerning the quality, the user can choose between

- full high-definition resolution (1920×1080), and
- low resolution (512×424).

One of the following three scenarios can be chosen to represent the remote partner:

1. **Symbol mode.** The point where the user touches the collaborative surface is marked with a symbol that follows the movements of the input.
2. **Hand translation mode.** Similar to the symbol mode, but instead of a symbol, the input movement is followed by a duplication of the users' own hand.
3. **Arm extension mode.** Similar to the approach of Higuchi et al. [6], the lower arm of the user is extended to where the user draws on the collaborative surface.

Data stream manipulation

The sensor's SDK offers an interface to read the video stream frame by frame. Each valid frame is manipulated according to the mode chosen (cf. Figure 4.5).

Independent of the chosen scenario, a background elimination is performed to extract the user from the environment. Subsequently, the sensor detects the body, as the body index source does not contain color information, the body index frame and the color frame have to be mapped to each other. The coordinate mapper contains information about what pixel of the body index belongs to what pixel in the color source (cf. Chapter 3.2.1). By matching the pixel of the body index with the color source pixel, all color pixel that do not match a body index are discarded. Consequently, the target image shows the user in color without background (cf. Figure 4.3).

Figure 4.4 shows the lookup procedure that has to be performed for each pixel.

Subsequently, the colored body is transformed according to the chosen transformation mode (cf. Fig. 4.5). The wrist body joint is required to determine the position of the wrist, as well as the elbow joint that is required to determine the vector of the lower arm. The arm extension mode



Figure 4.3: The resulting image after background elimination.

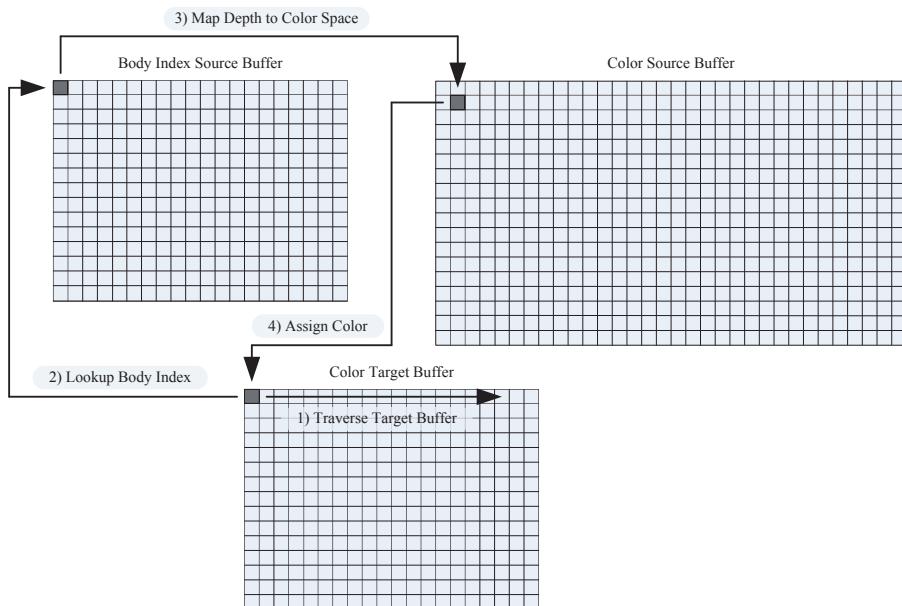


Figure 4.4: Three buffers are required for the data manipulation process: The color target buffer T , the body index source buffer B and the color source buffer C . The user can determine the size of T via the GUI. For each pixel of T it is verified if the matching pixel in B contains a valid body value. If this is the case, the corresponding color pixel of C is obtained and its value is written into T .

(cf. Fig. 4.5c) scales the lower arm to the input point. Therefore, the body is rendered without manipulation up to the right elbow.

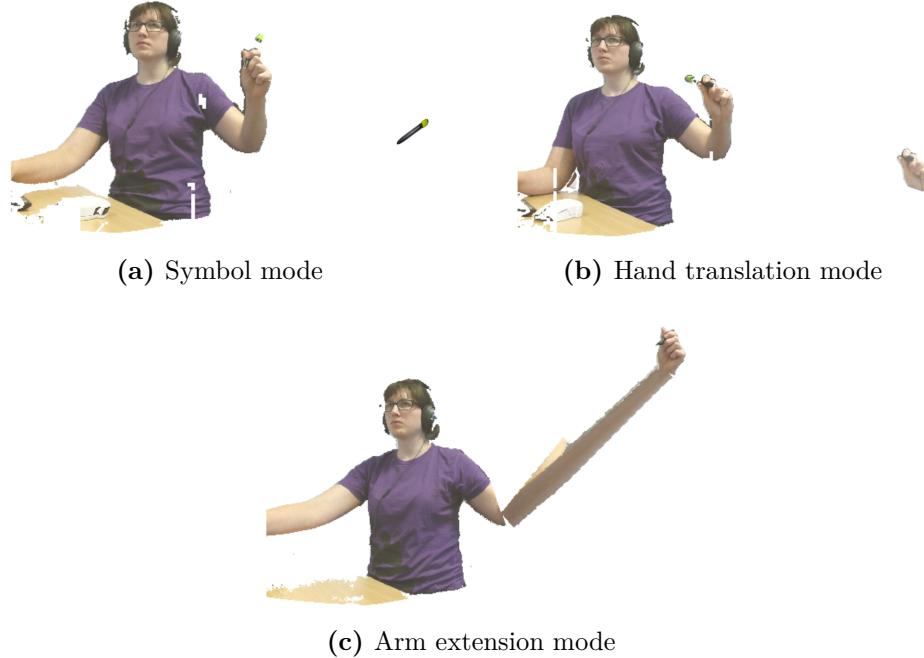


Figure 4.5: One of the following modes replaces the mouse cursor: (a) An arbitrary symbol can be chosen, eg. a pencil. (b) the writing hand is duplicated and translated, or (c) the writing arm is scaled to the input device's position.

4.1.3 Rendering

Finally, the transformed image of the user is projected onto the collaborative surface. To guarantee a minimum occlusion of the collaborative content, each client is able to adapt the opacity of the remote user with a slider in the GUI. Displaying the remote user on the collaborative surface creates the effect of facing each other.

4.2 Implementation details

For the prototype implementation, it is assumed that the user writes with the right hand. Different algorithms have been implemented to examine the performance on high-resolution images as well as on low resolution images. An image I is defined by

$$I = \{p = (x, y) \in \mathbb{N} \mid 1 \leq x \leq m, 1 \leq y \leq n\}, \quad (4.1)$$

$$\{m, n \in \{T_{LR}, T_{HD}\} \mid m_{T_{LR}} = 512, n_{T_{LR}} = 424, m_{T_{HD}} = 1920, n_{T_{HD}} = 1080\}. \quad (4.2)$$

4.2.1 Body detection

The user's body is detected in order to

- eliminate unnecessary information such as the background, and
- to ease the selection of user body parts for manipulation.

In order to extract the body from the background, the information in the body index frame (cf. Fig. 3.4d) is used. The color of each pixel p that belongs to a body b is written directly into the back buffer. A coordinate mapper (cf. Sec. 3.2.1) is used to check if p has a corresponding body index that indicates a body. As the information about the body is stored in the body index space as depth space points, the pixel position in color space has to be mapped to the depth space. The resulting image shows a person in RGBA color while the background is discarded.

In the prototype, the algorithm used to traverse the body index buffer is traversed can be chosen in the GUI. The different algorithms are compared and described in detail below. The performance of the presented algorithm depends on the number of pixel visited and on the number of checks if the pixel belongs to a body.

In short, the following steps have to be performed:

1. Select a pixel and check for body values.
2. For all pixel with a body value, get the color value using the mapper.
3. Write the color value into the back buffer.

Sequential approach

This iterative algorithm traverses the body index buffer pixel by pixel. It checks if the currently visited pixel p_i has a body assigned by

$$0 \leq p_i < 6, \quad i = 0, 1, \dots, m \times n \quad (4.3)$$

as the sensor assigns an index to each recognized body (cf. Sec. 3.2.1). Pixel that do not fulfill the above equation are discarded and the next pixel is processed. For all other pixel, the color is looked up in the color buffer and written into the target image.

Floodfill approach

In contrast to the sequential approach described above, flood fill algorithms do not have to visit each pixel of the body index buffer. Instead, they start at a given seed point s that lies inside the body, visit all neighboring pixel of s and terminate if a background pixel is hit. A seed point takes the value of a valid body joint. In case no body is detected, the sequential approach is executed and shows the unaltered color stream.

Several flood fill algorithms have been examined that differ in the order the neighboring pixel are visited. The search patterns are

- recursive (cf. Alg. 4.1),
- breadth-first search (BFS, cf. Alg. 4.2), and
- depth-first search (DFS, cf. Alg. 4.3).

The recursive algorithm traverses the region by using the calculated parameters in a new call of the function itself. A drawback of this algorithm is that the number of elements pushed onto the stack memory grows with each recursion, which could cause a stack overflow.

Algorithm 4.1: The recursive flood fill algorithm [1].

```

 $I = \text{bodyIndexBuffer}$  and  $\text{backgroundPixel} = 0\text{xff}$ 
 $\text{FLOODFILL}(x, y)$ 
  if coordinate  $(x, y)$  is within image boundaries and  $I(x, y) < \text{maxNumberofBodies}$  then
     $\text{GetColorAndDrawIntoBackbuffer}(x, y)$ 
     $\text{FloodFill}(x + 1, y)$ 
     $\text{FloodFill}(x - 1, y)$ 
     $\text{FloodFill}(x, y + 1)$ 
     $\text{FloodFill}(x, y - 1)$ 
  else
    return
  end if
end

```

The iterative BFS algorithm uses a queue to store the pixel to visit. Compared to the recursive approach, the risk of a memory overflow is exiguous as the object is stored in the heap memory. The pixel enqueued first are visited first (first-in first-out principle). The performance is slower as many elements are queued until a termination criteria comes true. The iterative DFS algorithm uses a stack to store the pixel to visit. Like the BFS algorithm, the object is stored in the heap memory. The pixel on top of the stack are visited first (last-in first-out principle). Therefore, all pixel are successively processed in one direction and therefore reaches the termination criteria at an earlier point in time than the BFS. Figure 4.6 shows how the different algorithms traverse an object.

Benchmark

The three algorithms presented above have been benchmarked to find the best algorithm in terms of framerate.

The measurement criteria were the mean frames f per second

$$\overline{fps} = \frac{1}{n} \sum_{i=1}^n f_i, \quad 1 \leq n \leq 30 \quad (4.4)$$

Algorithm 4.2: The BFS flood fill algorithm using a queue [1].

```

FLOODFILL(x, y)
    Create an empty queue  $Q$ 
    Insert the seed point into  $Q$ :  $ENQUEUE(Q, (x))$ ,
 $ENQUEUE(Q, (y))$ 
    while  $Q$  is not empty do
        Get next point  $p_{next}$  by dequeuing the first coordinate of  $Q$ :  $x \leftarrow DEQUEUE(Q)$ ,  $y \leftarrow DEQUEUE(Q)$ 
        if  $p_{next}$  is within image boundaries and  $I(x, y) \neq \text{backgroundPixel}$ 
        then
             $GetColorAndDrawIntoBackbuffer(x, y)$ 
             $SetPixelVisited(p_{next})$ 
             $ENQUEUE(Q, x + 1)$ 
             $ENQUEUE(Q, y)$ 
             $ENQUEUE(Q, x - 1)$ 
             $ENQUEUE(Q, y)$ 
             $ENQUEUE(Q, x)$ 
             $ENQUEUE(Q, y + 1)$ 
             $ENQUEUE(Q, x)$ 
             $ENQUEUE(Q, y - 1)$ 
        end if
    end while
end

```

n stating the maximum number of frames in 1 second, and the standard deviation from the mean

$$s_n = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (f_i - \bar{f})^2} \quad (4.5)$$

Note that the sensor cannot process more than 30 fps with good lighting conditions (cf. Sec. 3.1).

The following figures have been measured when rendering a using high-definition target buffer were the following:

Algorithm	sequential	recursive	DFS	BFS
Frame duration (s)	0.0974	0.0495	0.0775	96.7405
\overline{fps}	11.1046	20.8889	13.0563	0.0107
s_N	5.6996	3.5752	1.2959	0.0025

The measurements using a target buffer with a resolution of 512×424 were the following:

Algorithm 4.3: The DFS flood fill algorithm using a stack [1].

```

FLOODFILL(x, y)
    Create an empty stack  $S$ 
    Push the seed point onto  $S$ :  $PUSH(S, x), PUSH(S, y)$ 
    while  $S$  is not empty do
        Get next point  $p_{next}$  by pulling the first coordinate of  $S$ :  $x \leftarrow POP(S), y \leftarrow POP(S)$ 
        if  $p_{next}$  is within image boundaries and  $I(x, y) \neq \text{backgroundPixel}$ 
        then
             $GetColorAndDrawIntoBackbuffer(x, y)$ 
             $SetPixelVisited(p_{next})$ 
             $PUSH(S, x + 1)$ 
             $PUSH(S, y)$ 
             $PUSH(S, x - 1)$ 
             $PUSH(S, y)$ 
             $PUSH(S, x)$ 
             $PUSH(S, y + 1)$ 
             $PUSH(S, x)$ 
             $PUSH(S, y - 1)$ 
        end if
    end while
end

```

Algorithm	recursive	DFS	BFS
Frame duration (s)	0.0335	0.0334	5.1768
\overline{fps}	30.0970	30.1286	0.2035
s_N	2.8035	2.3637	0.0444

A benchmark of the three algorithms stated above showed that using a high-definition target buffer T_{HD} , the recursive algorithm performs best if sufficient stack space is available. The second best performance was delivered by the DFS which comes at a low cost concerning stack size, followed by the sequential approach and the BFS, which has a very low framerate what results in a stagnant output stream.

Using a low-definition target buffer T_{LR} , the DFS and the recursive approach have an equal performance. The BFS algorithm again has the lowest framerate, but shows a fluid output stream..

Performance enhancement

As high definition images are used within the prototype, the algorithm performs background checks and coordinate mappings for 1920×1080 in the worst case. Efficient algorithms as stated in Sec. 4.2.1 might lead to an overhead on less powerful computers. In order to increase the performance of

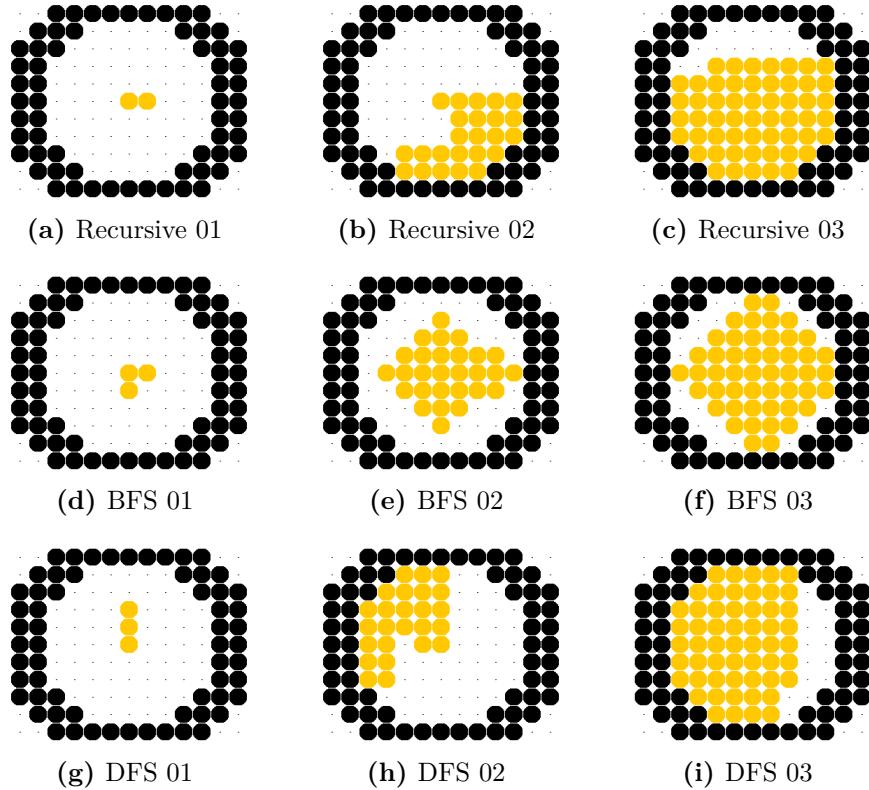


Figure 4.6: The recursive and the DFS algorithm proceed into the different directions successively while the BFS algorithm grows circular from the seed point. In all cases, the seed point was placed in the middle of the object.

the application, several considerations have to be taken, such as

- recursive flood fill operations are started in their own thread to avoid a stack overflow,
- unmanaged code is preferred over managed code,
- color values are written directly into the back buffer,
- operations within a loop are to be kept to a minimum,
- primitive types are preferred over structures, and
- operations such as modulo and division are to be kept to a minimum.

4.2.2 Arm transformation

Transformation of the arm has been implemented with per pixel approaches (cf. scale and rotation) and a vector based approach (cf. combined scale and rotation). The strengths and drawbacks of each approach are described in detail below.

Per pixel transformation: Scale

I devised an algorithm for scaling a subregion of the color stream with a low resolution target buffer. The example describes the scaling of the right lower arm A . As a precondition it is assumed that all pixel on the right side of the elbow joint

$$\forall p \notin A \mid p_{x_i} < p_{x_{elbow}} \quad (4.6)$$

are scaled.

When scaling the arm it has to be considered that the prolonged arm has to have the same length at all time. Using the same scale factor for all angles of the arm would result in an unproportional scale in one of the directions.

First, a scaling factor has to be determined for the x and y direction respectively. Therefore, the slope is taken into account. To get the slope parameters, the angle α between a horizontal line \mathbf{l} starting from p_{elbow} and the arm vector \mathbf{a} is calculated using the dot product [7]

$$\mathbf{a} \cdot \mathbf{l} = |\mathbf{a}| |\mathbf{l}| \cos(\alpha) \Rightarrow \alpha = \arccos\left(\frac{\mathbf{a} \cdot \mathbf{l}}{|\mathbf{a}| |\mathbf{l}|}\right) \quad (4.7)$$

The angle is used for the calculation of the scale factors

$$\text{factor}_x = \cos \alpha \text{ and} \quad (4.8)$$

$$\text{factor}_y = \sin \alpha. \quad (4.9)$$

The lookup position and color for scaled regions is calculated as follows.

The distance of the current pixel to the elbow calculated by

$$\Delta_x = x_i - x_{elbow} \text{ and} \quad (4.10)$$

$$\Delta_y = y_i - y_{elbow}. \quad (4.11)$$

This offset is used to determine the lookup position (cf. Fig. 4.7)

$$x'_i = x_{elbow} + \frac{\Delta_x}{\text{factor}_x} \text{ and} \quad (4.12)$$

$$y'_i = y_{elbow} + \frac{\Delta_y}{\text{factor}_y}. \quad (4.13)$$

Subsequently, the color for the target frame T is determined. The color lookup from the color frame C for scaled and non-scaled regions is defined by

$$T(p_i) = \begin{cases} C(p_i) & \text{if } p_x \leq p_{x_{elbow}} \\ C(p'_i) & \text{if } p_x > p_{x_{elbow}} \end{cases} \quad (4.14)$$

Consequently, the arm is scaled into the direction of the touch input. However, the resultant target frame shows that sine and cosine are insufficient parameters for an exact mapping onto the touch point vector.

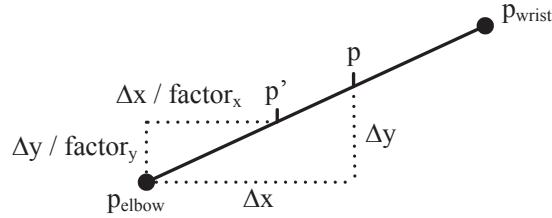


Figure 4.7: The scale factor is calculated using the sine for step size in the y-direction (factor_y) and the cosine for the step size in the x-direction (factor_x).



Figure 4.8: (a) The body before transformation and (b) after the scale operation has been applied to the lower arm.

Per pixel transformation: Rotation

Hearn and Baker [5] define a 2D-rotation matrix for the angle θ by

$$R = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \quad (4.15)$$

In order to rotate the right arm A that belongs to the body of the remote collaborator, each pixel

$$p_i \in A, \quad i = 0, 1, \dots, n, \quad A \subset I \quad (4.16)$$

has to be rotated. The rotation angle θ is determined by the angle between the arm vector

$$\mathbf{a} = \begin{pmatrix} x_{wrist} - x_{elbow} \\ y_{wrist} - y_{elbow} \end{pmatrix}, \quad a_0, a_1, \dots, a_n \in A \quad (4.17)$$

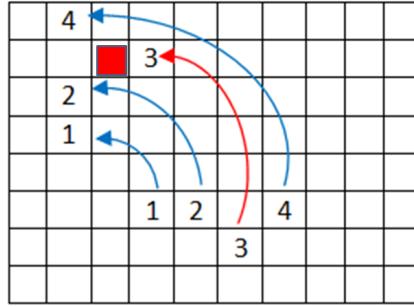


Figure 4.9: The 2D-rotation uses floating points while for the determination of the new position, fixed point values are required. Due to the translation into fixed point values, some pixel within the image (red colored pixel) are never reached. Consequently, white noise occurs in the final image.

and the rotated target arm vector

$$\mathbf{a}_{rotated} = \begin{pmatrix} x_{touch} - x_{elbow} \\ y_{touch} - y_{elbow} \end{pmatrix}, \quad a_{rotated_0}, \dots, a_{rotated_n} \in I \quad (4.18)$$

that is identified by the touch point of the input device (cf. Fig. 3.6). Based on these vectors, the angle from \mathbf{a} to $\mathbf{a}_{rotated}$ is calculated by the dot product [7]

$$\mathbf{a} \cdot \mathbf{a}_{rotated} = |\mathbf{a}| |\mathbf{a}_{rotated}| \cos(\theta) \Rightarrow \theta = \arccos\left(\frac{\mathbf{a} \cdot \mathbf{a}_{rotated}}{|\mathbf{a}| |\mathbf{a}_{rotated}|}\right), \quad (4.19)$$

with $|\mathbf{a}|$ being the magnitude of \mathbf{a} . Subsequently, the rotation R is applied to \mathbf{a} :

$$\mathbf{a}R(\theta) = \mathbf{a}_{rotated}. \quad (4.20)$$

The arm is correctly transformed within the resulting image but a regular pattern of white noise occurs (cf. Fig. 4.10b). This noise is caused by the inevitable translation of floating point positions resulting from the rotation R to fixed point positions that are required to draw the final pixel to the image (cf. Fig. 4.9).

Vector based transformation: Combined scale and rotation

To eliminate floating point errors and to match the touch input point more precisely than the scaling algorithm, a vector based approach was formulated. Given are the two vectors of the arm \mathbf{a} (cf. Eq. 4.17) and the virtual rotated and scaled arm $\mathbf{a}_{virtual}$ (cf. Eq. 4.18). In contrast to the per pixel operation, the rotation is not calculated explicitly, but implicitly by simply stepping along \mathbf{a} and subsequently applying its color values onto $\mathbf{a}_{virtual}$ (cf. Fig. 4.11).

The adequate step size or sampling size is determined by the magnitude of \mathbf{a} and $\mathbf{a}_{virtual}$. According to the Nyquist sampling interval described by

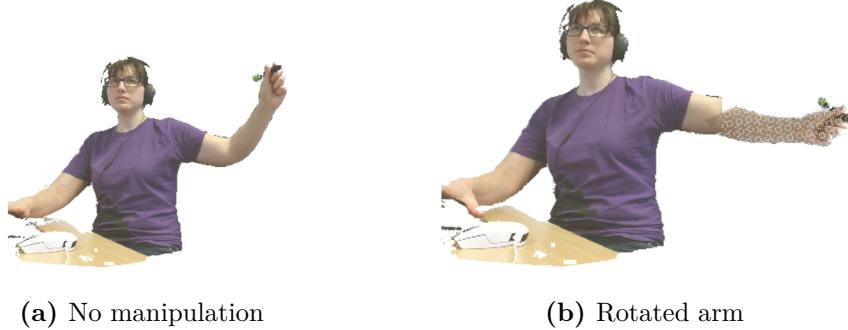


Figure 4.10: The image before 4.10a and after 4.10b the applied rotation of the right lower arm.

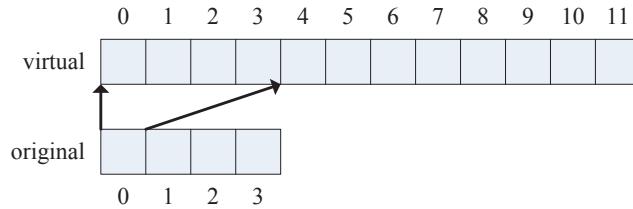


Figure 4.11: The values of the original vector \mathbf{a} are subsequently applied to the virtual vector $\mathbf{a}_{virtual}$

Hearn and Baker [5], the magnitude of the longer vector multiplied by a factor of 2 is chosen for a step size that is used to traverse \mathbf{a} and $\mathbf{a}_{virtual}$. This oversampling of the step size guarantees a noise free result, regardless of the resolution of the final image.

Given

$$\mathbf{a} = \begin{pmatrix} x \\ y \end{pmatrix}, \quad (4.21)$$

the rest of the arm is calculated by stepping along its two normals, the left normal

$$\hat{\mathbf{a}}_{left} = \begin{pmatrix} y \\ -x \end{pmatrix}, \quad (4.22)$$

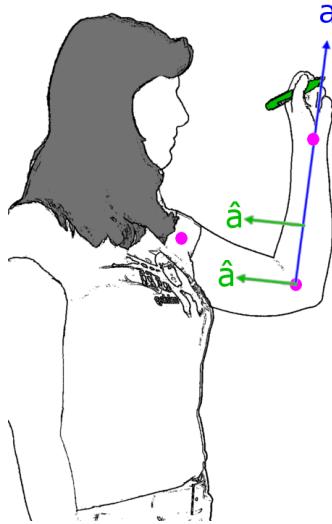


Figure 4.12: In the region of the elbow, the normal vector points into the body. Without an appropriate termination criteria, parts of the upper arm are transformed along with the lower arm.

and the right normal

$$\hat{\mathbf{a}}_{right} = \begin{pmatrix} -y \\ x \end{pmatrix}, \quad (4.23)$$

considering that the origin of the coordinate system is the upper left corner of the image.

The normals of \mathbf{a} are proceeded and its color values drawn onto $\mathbf{a}_{virtual}$. Oversampling is not required, therefore the step sizes are x and y . In case the normal $\hat{\mathbf{a}}_{left}$ points into the body, a suitable boundary has to be set as explained in detail below (cf. Fig. 4.12).

4.2.3 Determination of accurate boundary vectors

In case the normal $\hat{\mathbf{a}}_{left}$ points into the body (cf. Fig. 4.12), a suitable boundary has to be set. Furthermore, the region of the hand has to be separated from region of the lower arm in order to prevent the hand from the transformations that are applied to the lower arm. Two boundary vectors have to be evaluated:

1. The lower boundary located at the elbow (cf. Fig. 4.13a), and
2. the upper boundary located at the wrist (cf. Fig. 4.13b).

The lower boundary \mathbf{b}_{lower} is most likely to interfere with the body as $\hat{\mathbf{a}}_{left}$ points into the upper arm in the region of the elbow point. The standard algorithm steps along $\hat{\mathbf{a}}_{left}$ until it reaches a background pixel. In this case, without a limiting boundary for \mathbf{b}_{lower} , the algorithm continues

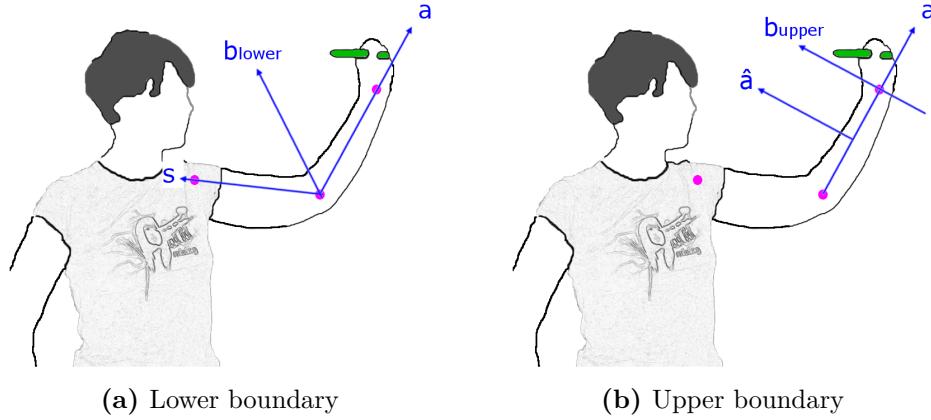


Figure 4.13: (a) The lower boundary is defined by the half vector \mathbf{b}_{lower} of the shoulder vector \mathbf{s} and the arm vector \mathbf{a} . (b) The upper boundary is defined by the normals of the arm vector $\hat{\mathbf{a}}_{left}$ and $\hat{\mathbf{a}}_{right}$.

reading pixel of the upper arm. Those pixels would also enter the next step of transformation and would get scaled and rotated. In the rendered image artefacts would occur. An improved termination criteria is the half vector between the shoulder vector

$$\mathbf{s} = \begin{pmatrix} x_{shoulder} - x_{elbow} \\ y_{shoulder} - y_{elbow} \end{pmatrix} \quad (4.24)$$

and the lower arm vector \mathbf{a} . It is calculated as follows:

$$\mathbf{b}_{lower} = \frac{\mathbf{s} + \mathbf{a}}{|\mathbf{s} + \mathbf{a}|}. \quad (4.25)$$

The upper boundary \mathbf{b}_{upper} is set to the vector normal

$$\mathbf{b}_{upper} = \hat{\mathbf{a}}_{left}. \quad (4.26)$$

As \mathbf{b}_{upper} does not interfere with other body parts, the boundary is naturally limited by the wrist joint and does not have to be checked explicitly. However, the boundary criteria would not hold if the wrist touches other parts of the body or if it is placed in front of the body. A possible solution to this problem is presented in Section 4.2.4

4.2.4 Overlapping body parts

In practice, the writing hand often occurs to be in front of the body (cf. Fig. 4.14a). This occurs to be a problem within the hand extraction mode as well as the arm extension mode respectively, as both use a flood fill algorithm (cf. Sec. 4.2.1) to determine the hand section. Furthermore, the termination

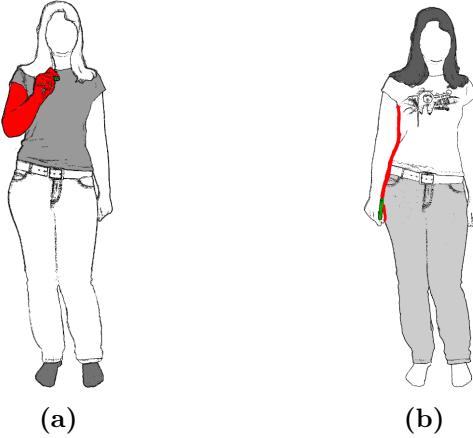


Figure 4.14: (a) The algorithm uses the depth value to identify hand and body region. (b) The algorithm cannot distinguish between body and arm as the depth is equal and no background pixel occur in between them.

criteria within the arm extension mode are no longer valid if the user places the arm parallel to the body, closing the gap between arm and body. Consequently, the algorithm would not terminate as no background pixel is visited. The solution is to extend the abort criterion by the depth value. The sensor delivers the depth value in millimeters from its focal plane (cf. Sec. 3.2.1) and that value is used in combination with a predefined threshold t in order to determine if the point belongs to the writing hand a or the body b . The difference of the depth values

$$\Delta d = I(p_i) - I(p_{i+1}) \quad (4.27)$$

is compared with the threshold and if the absolute value

$$||\Delta d|| > t, \quad (4.28)$$

the algorithm terminates. In this way an overflow of the hand fill is prevented, and the arm is extracted, scaled, and positioned correctly.

However, if the hand has a contact to the body, the depth values of the arm cannot be distinguished from the rest of the body (cf. Fig. 4.14b).

4.3 Limitations of the prototype implementation

For the prototype it is assumed that the user is right handed. To generalize the writing hand, the depth frame may be used to determine which hand is closer to the collaborative surface.

The output of the manipulated images strongly depends on the sensor. There are two cases to consider:

1. the sensor looses track of all body joints, and
2. the sensor looses track of the right arm joints.

If the sensor looses track of all body joints, there is no body index assigned. Accordingly, all pixel are background pixel and therefore not processed. The result is either a blank screen or jitter in the video stream, depending on the stability of the sensor signal.

In case the sensor looses track of the right arm joints, the frame is processed, but without a manipulation of the user's arm. The resultant frame shows the user without background. If the sensor signal changes steadily from tracked to non-tracked, the output stream is disturbed by alternating manipulated frames.

Furthermore, the performance depends on the used computing machine. The less memory and the less powerful the CPU is, the longer the rendering of the frames take. In this case, the output stream would have a noticeable low framerate.

Chapter 5

Conclusion

Multiple collaboration methods have been examined and analyzed according to several criteria. As stated above, communicative meta-information such as gestures and mimics are important to enhance conversations. However, they are not considered by all related works. The aspect of multiuser support is difficult as multiple objects have to be tracked while eye-contact has to be maintained. A simple setup and standard hardware requirements are desirable to make the system affordable. RGBD cameras are commonly available and offer a broad palette of functionality.

A prototypical setup has been implemented to enhance interaction gestures of the user. The main aim was to imitate a face-to-face meeting to preserve the personal touch even in a remote collaboration system. As a transparent whiteboard with the camera capturing the user from behind is difficult to obtain and set up, an alternative system has been chosen which uses a videoconferencing system, an interactive whiteboard and a RGBD camera. A setup where the camera is located at the front right of the user, next to the whiteboard, shows perspective disturbances when it is overlaid with the collaborative content. Therefore, the user's extremities have to be manipulated to balance them out. This enhanced gesture visualizations are achieved by showing the whole body of the user and by additionally replacing the user's cursor by either a selectable symbol, a duplicate of the user's own hand or scaling the hand to the correct position.

5.1 Future work

Further improvements on the prototype include

- a hardware acceleration by using the GPU for the computation and the rendering stage,
- a user study to fully explore the potentials and drawbacks of the prototype, and
- an automatic recognition of writing hand to support both left-handed

and right-handed users.

Appendix A

System requirements

Microsoft [23, 24] recommends appropriate system components when working with a Kinect v2 sensor as following.

The personal computer should possess

- Windows 8.1 or later as an operating system,
- a built-in USB 3.0 host controller (PCIe 2.0 with IntelRenesas Chipsets),
- a 64 bit processor (x64),
- at least 4 GB memory,
- at least an i7 processor with 3.1 GHz (physical dual-core with 2 logical cores per physical core).

Furthermore, the sensor requires a DirectX 11.0 capable graphics adapter, such as

- Intel HD 4400 integrated display adapter, or
- ATI Radeon HD 5400 series, or
- ATI Radeon HD 6570, or
- ATI Radeon HD 7800 (256-bit GDDR5 2GB1000Mhz), or
- NVidia Quadro 600, or
- NVidia GeForce GT 640, or
- NVidia GeForce GTX 660, or
- NVidia Quadro K1000M.

Concerning the software and libraries, the following components are recommended:

- Visual Studio 2012 Windows Desktop,
- C# .NET WPF Project, and
- Kinect v2 SDK.

References

Literature

- [1] Wilhelm Burger and Mark J. Burge. *Digitale Bildverarbeitung*. 1st ed. Springer Publishing Company, Incorporated, 2005 (cit. on pp. 28–30).
- [2] Yinpeng Chen et al. *ViiBoard: Vision-enhanced Immersive Interaction with Touch Board*. http://research.microsoft.com/en-us/projects/mic_viiboard/ (cit. on pp. 6, 7).
- [3] Teresa Farroni et al. “Eye contact detection in humans from birth”. In: *Proc. National Academy of Sciences* 99.14 (2002), pp. 9602–9605. eprint: <http://www.pnas.org/content/99/14/9602.full.pdf+html>. URL: <http://www.pnas.org/content/99/14/9602.abstract> (cit. on p. 1).
- [4] Edward T. Hall. *The Hidden Dimension*. Doubleday, 1966 (cit. on p. 1).
- [5] Donald Hearn and M. Pauline Baker. *Computer Graphics (2nd Ed.): C Version*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1997 (cit. on pp. 33, 35).
- [6] Keita Higuchi et al. “ImmerseBoard: Immersive Telepresence Experience Using a Digital Whiteboard”. In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. CHI ’15. Seoul, Republic of Korea: ACM, 2015, pp. 2383–2392. URL: <http://doi.acm.org/10.1145/2702123.2702160> (cit. on pp. 7, 24).
- [7] Francis S. Hill Jr. and Stephen M Kelley. *Computer Graphics Using OpenGL (3rd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006 (cit. on pp. 32, 34).
- [8] Ellen A. Isaacs and John C. Tang. “What Video Can and Can’t Do for Collaboration: A Case Study”. In: *Proc. Multimedia*. Anaheim, California, USA: ACM Press, 1993, pp. 199–206. URL: <http://doi.acm.org/10.1145/166266.166289> (cit. on p. 1).

- [9] Hiroshi Ishii and Minoru Kobayashi. “ClearBoard: A Seamless Medium for Shared Drawing and Conversation with Eye Contact”. In: *Proc. SIGCHI*. Monterey, California, USA: ACM Press, 1992, pp. 525–532. URL: <http://doi.acm.org/10.1145/142750.142977> (cit. on p. 5).
- [10] Shahram Izadi et al. “Dynamo: A Public Interactive Surface Supporting the Cooperative Sharing and Exchange of Media”. In: *Proc. UIST*. Vancouver, Canada: ACM Press, 2003, pp. 159–168. URL: <http://doi.acm.org/10.1145/964696.964714> (cit. on p. 6).
- [11] S. Izadi et al. “C-Slate: A Multi-Touch and Object Recognition System for Remote Collaboration using Horizontal Surfaces”. In: *TABLETOP - International Workshop on Horizontal Interactive Human-Computer Systems*. IEEE Computer Society, 2007, pp. 3–10 (cit. on pp. 8, 9).
- [12] Brett R. Jones et al. “IllumiRoom: Peripheral Projected Illusions for Interactive Experiences”. In: *Proc. SIGCHI*. Paris, France: ACM Press, 2013, pp. 869–878. URL: <http://doi.acm.org/10.1145/2470654.2466112> (cit. on pp. 6, 7).
- [13] Brett Jones et al. “RoomAlive: Magical Experiences Enabled by Scalable, Adaptive Projector-camera Units”. In: *Proc. UIST*. Honolulu, Hawaii, USA: ACM Press, 2014, pp. 637–644. URL: <http://doi.acm.org/10.1145/2642918.2647383> (cit. on p. 6).
- [14] Shunichi Kasahara et al. “Second surface: Multi-user Spatial Collaboration System Based on Augmented Reality”. In: *SIGGRAPH Asia*. ACM Press, 2012, 20:1–20:4. URL: <http://doi.acm.org/10.1145/2407707.2407727> (cit. on p. 6).
- [15] Kar-Han Tan et al. “ConnectBoard: Enabling Genuine Eye Contact and Accurate Gaze in Remote Collaboration”. In: *Multimedia, IEEE Transactions on* 13.3 (2011), pp. 466–473 (cit. on p. 5).
- [16] Cameron Teoh, Holger Regenbrecht, and David O’Hare. “How the Other Sees Us: Perceptions and Control in Videoconferencing”. In: *Proc. OzCHI*. Melbourne, Australia: ACM Press, 2012, pp. 572–578. URL: <http://doi.acm.org/10.1145/2414536.2414624> (cit. on p. 1).
- [17] Cameron Teoh, Holger Regenbrecht, and David O’Hare. “Investigating Factors Influencing Trust in Video-mediated Communication”. In: *Proc. OzCHI*. Brisbane, Australia: ACM Press, 2010, pp. 312–319. URL: <http://doi.acm.org/10.1145/1952222.1952289> (cit. on p. 1).
- [18] A. Valadares and Cristina V Lopes. “Virtually Centralized, Globally Dispersed: A Sametime 3D Analysis”. In: *Workshop on Location Awareness for Mixed and Dual Reality (LAMDa’11), in conjunction with IUI 2011*. 2011 (cit. on pp. 8, 9).

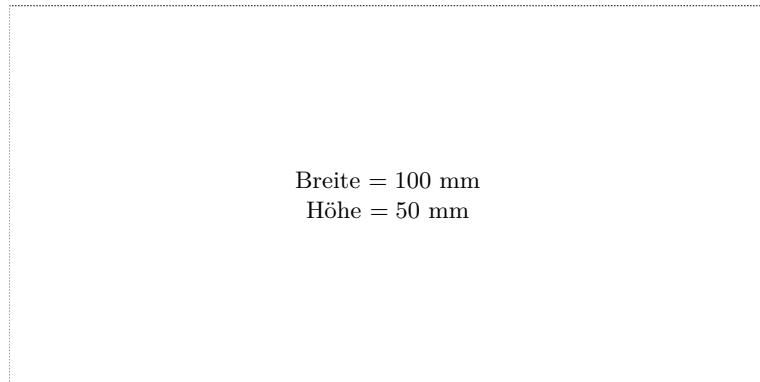
- [19] Cha Zhang et al. “Viewport: A Distributed, Immersive Teleconferencing System with Infrared Dot Pattern”. In: *MultiMedia, IEEE* 20.1 (2013), pp. 17–27 (cit. on p. 10).
- [20] Jakob Zillner. “3D-Board: A Remote Collaborative Workspace Featuring Virtual 3D Embodiments”. MA thesis. Hagenberg, Austria: FH Oberösterreich – Fakultät für Informatik, Kommunikation und Medien, 2014 (cit. on p. 1).

Online sources

- [21] Inc. CASIO America. 2015 (accessed August 3, 2015). URL: http://www.casio-intl.com/asia-meia/en/projector/whiteboard/ya_w72m/ (cit. on p. 3).
- [22] w’inspire GmbH. 2015 (accessed August 3, 2015). URL: <https://we-inspire.com/technology> (cit. on p. 3).
- [23] Microsoft. 2014 (accessed July 3, 2015). URL: <https://msdn.microsoft.com/en-us/library/dn799271.aspx> (cit. on pp. 14, 42).
- [24] Microsoft. 2014 (accessed July 3, 2015). URL: <https://www.microsoftvirtualacademy.com/en-us/training-courses/programming-kinect-for-windows-v2-jump-start-9088> (cit. on pp. 14, 18, 42).
- [25] Microsoft. 2015 (accessed August 3, 2015). URL: <https://www.microsoft.com/microsoft-surface-hub/> (cit. on p. 3).
- [26] Microsoft. 2015 (accessed August 4, 2015). URL: <http://download.microsoft.com/download/6/7/6/676611B4-1982-47A4-A42E-4CF84E1095A8/KinectHIG.2.0.pdf> (cit. on pp. 14, 15).
- [27] Derek Strickland. 2015 (accessed July 3, 2015). URL: <http://www.vrworld.com/2015/04/05/microsoft-will-stop-producing-kinect-for-windows/> (cit. on p. 14).

Messbox zur Druckkontrolle

— Druckgröße kontrollieren! —



Breite = 100 mm
Höhe = 50 mm

— Diese Seite nach dem Druck entfernen! —