Detection and Localisation of Pointing, Pairing and Grouping Gestures for Brainstorming Meeting Applications

Simon Liechti, Naina Dhingra(✉), and Andreas Kunz

Innovation Center Virtual Reality, ETH Zurich,
Leonhardstr. 21, 8092 Zurich, Switzerland
liechtsi@student.ethz.ch, {ndhingra,kunz}@iwf.mavt.ethz.ch
https://www.icvr.ethz.ch

Abstract. The detection of gestures and their interpretation is crucial for blind and visually impaired people (BVIP). In a card-based brainstorming meeting, sighted users use non-verbal communication when referring to cards on a common workspace using pointing, grouping, or pairing gestures. While sighted users could easily interpret such gestures, they remain inaccessible to BVIP. Thus, there is a need for capturing, interpreting and translating gestures for BVIP.

To address this problem, we developed a pointing gesture detection system using Unity with the SteamVR Plugin and HTC Vive. HTC's trackers are attached to a user's hands to measure the hand position in 3D space. With pointing gestures, a user controls a virtual ray that will intersect with a virtual whiteboard. This virtual whiteboard is invisible to the sighted users, but its position and size corresponds to a physical whiteboard. The intersection of the ray with the virtual whiteboard is calculated, resulting in a pointing trajectory on it. The shape of the trajectory is analyzed to determine, which artifacts are selected by the pointing gesture. A pointing gesture is detected when a user is pointing at a card on the screen and then ending the gesture by pointing outside of the screen. A pairing gesture is detected when pointing at one artifact and then on another one before leaving the screen. The grouping gesture is detected when performing an encircling gesture around multiple artifacts before leaving the screen.

Keywords: Blind and visually impaired · Pointing gestures · Virtual reality

1 Introduction

In team meetings, gestures are used together with speech to support arguments during a discussion. These gestures belong to non-verbal communication (NVC), which could make up to 55% of the overall information flow [8]. Since these gestures are inaccessible for the blind and visually impaired (BVIP), they need to be detected and interpreted [3], otherwise they would not be able to participate in such meetings to a full extent. Among these NVC elements, pointing gestures are the most important ones, which typically refer to artifacts in the 3D space of a meeting room, e.g. on a common whiteboard. Within this paper, we focus on the so-called "Metaplan" method, which is well established in industry and research as well for idea generation in team meetings. When looking at the NVC elements, most gestures refer to artifacts (cards) on a common whiteboard. These pointing gestures can typically be divided into i) pointing at an artifact, ii) pairing of two artifacts, and iii) grouping. Accordingly, an imaginary "pointing ray" on the whiteboard would be dot-shaped, line-shaped, or encircling.

This paper focuses on the reliable interpretation of pointing gestures. For this, the paper is structured as follows: Sect. 2 will describe the state-of-the-art in this field, followed by Sect. 3 that describes how we track and interpret the gestures. Section 4 will describe the technical setup. Section 6 will show preliminary results, before Sect. 7 will conclude this paper.

2 Related Work

There is only little work in interpreting pointing gestures on artifacts in a common workspace. In [5], an MS Pixelsense table was used together with an MS Kinect depth camera to track pointing gestures above the table. To overcome still existing interference problems between Kinect and Pixelsense, later work used LEAP Motion sensors [6,11]. To interpret the pointing gestures, an information infrastructure was developed in [10] to translate the pointing gestures of sighted users to a BVIP output interface. The application used in this study was a Mindmap tool, and users were supposed to stand around the table [1]. Thus, the users' gestures were close to the artifacts on the Pixelsense table, making their pointing gestures more accurate. However, when users refer to a common whiteboard like in our Metaplan method, their distance to the artifacts is larger and the pointing gestures tend to be less accurate as described in [2]. In their setup, an MS Kinect was placed on the vertical whiteboard, while users performed pointing gestures from a 1.5 m distance. In their study, the pointing accuracy was found to be around 80%, but no other gesture interpretation was involved. If the pointing hand can be captured from a shorter distance like in [7], an accuracy of approximately 90% at a distance of 1m in front of an ideal background for a background segmentation can be achieved. However, no other gesture interpretation was researched. In [9], Hidden Markov Models and Particle Filters were used to improve the detection accuracy of pointing gestures to about 90%. More recently, [12] used convolutional neural networks to detect hand gestures for pointing and also achieved an accuracy of 90%. However, we found no research work that brings hand gestures in relation to artifacts on a common whiteboard for pointing, pairing, and grouping.

While the accuracy of such deep learning approaches is already sufficiently high, these approaches still suffer from the fact that they require a lot of time to train the underlying network properly, and also the detection times are usually in the range of second. Due to this lack of real-time capability, these approaches can yet not be used in lively discussions such as brainstorming session in a team.

3 Method

During a Metaplan session, participants use gestures to support a facilitator in rearranging cards on the whiteboard. The artifacts (cards) are part of the visible whiteboard and it is assumed that all types of gestures are only valid when they are detected inside the whiteboard's area. When gestures are detected to be valid, they mainly differ in their shape. During a brainstorming session, mainly three kinds of gestures will occur: pointing, pairing, grouping (see Fig. 1).

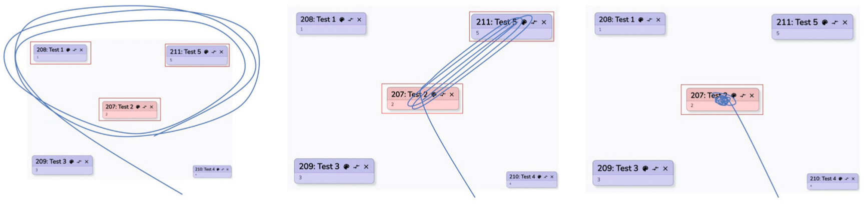

Fig. 1. Trajectories of groupin, pairing, and pointing gestures.

Users typically perform such pointing gestures imprecisely, so that an imaginary pointing ray on the whiteboard would have an elliptic shape even though the user actually wanted i) not to move (pointing), ii) draw a straight line (pairing), or iii) draw a circle (grouping). To differentiate between these gestures, the area of a gesture is approximated as an ellipse and their properties are compared. As a differentiation measure, we use the ratio between the semi-minor and the semi-major axes, as well as the curvature.

In a first step, our algorithm will compute a "G-value" as a basic distinction between circle and ellipse. For this, it connects the sample points (see Sect. 4) by straight lines and calculates the angles between subsequent lines. Summing up all angles and averaging them by the amount of angles will give α_{avg}. Next, each angle α is compared to this average by subtracting it from the average value. If the pointing trajectory has a circular shape, the difference is ideally equal to Zero (or close to it for small "deformations" of the circle. Thus, summing up all differences will result in a G-value close or even equal to zero, or in a high G-value for ellipses, where some angles are significantly different (see Eq. 1).

$$\sum_{i=0}^{n} |\alpha_{\text{avg}} - \alpha_{\text{i}}| = G \tag{1}$$

This G-value is the threshold for distinguishing between circle and ellipse, i.e. between a grouping and a pairing gestures. Based on an empirical study, this G-value was set to be $G = 100$. Once the G-value is determined, a bounding box (blue) is placed around sample points of the ellipse (see Fig. 2).

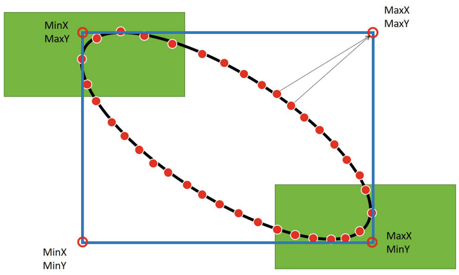

Fig. 2. Determining the orientation of the ellipse and the corner sample points.

Pairing Gesture. If the G-value determines a pairing gesture, the application needs to find the orientation of the ellipse to determine, which two corners of the bounding box will be used for note recognition. For this, the distance of each sample point to each corner of the bounding box is calculated to find the two smallest distances of sample points to the two corners of the bounding box. The two sample points with the smallest distances to the two corners of the bounding box will be further processed (see Fig. 2). For the correct recognition of a note, the distance of the note's center point and the "corner" sample points has to be smaller than a predefined value d. Based on empirical studies, this value was set to be $d = 0.3$ normalized to the size of the whiteboard. To avoid false positives, the pairing gesture's results will only be forwarded to the BVIP if two different cards are detected by the aforementioned method. If the corner sample points refer to the same note, the system will recognize it as a pointing gesture.

Grouping Gesture. If the G-value detects a grouping gesture, the coordinates of all the notes' centerpoints are compared to the bounding box. All notes inside are then checked for their uniqueness. If only one note is inside. The system recognises it as a pointing gesture. If two notes are inside, the system will again recognize it as a pairing gesture of two notes. If more than two notes are inside the bounding box, the system will output a grouping gesture.

4 Implementation

4.1 Modelling

The detection of gestures is done by overlaying an invisible virtual whiteboard on the physical one, having the same form factor and orientation (see Fig. 3). For modeling, the Unity game engine was used. In particular, the following elements or functions were relevant:

– *Screen.* The invisible virtual screen represents the digital whiteboard with the notes attached to it. It is a game object with colliders. The screen's size can be changed by changing its pixel count for the x- and y-axis.

- *Notes.* The notes are a virtual representation of the notes that are displayed on the digital screen during the brainstorming. They are created by a script at the start of the application. This script makes a request to the server to get the information which notes are currently on the selected digital whiteboard. Using this information, virtual twins are created as game objects on the virtual screen as children of it. For any game object being a child of another game object, the child has normalized local coordinates reaching from -0.5 to 0.5 for each axis. The origin of the local coordinate system is in the center of the parent game object. Each note has a script attached to it that holds further information of the note, such as *positionx, positiony, rotation, scale, title*, and *body.* This information is updated each frame by a server request. The note also has a unique tag called "note". This tag allows for differentiation of colliders when using the raycasting method.
- *Scripting.* The scripting in Unity is done with C#. To use a script inside Unity, the script needs to be attached to a game object. Variables of that game object like position, scale, and rotation can also be used in the script. In the Unity script, there are two important functions. void Start() and void Update(). The "Start" function is executed when the application is started. The "Update" function is executed for each frame the application is running.
- *Raycasting.* For detecting pointing gestures, Unity's "Raycasting" method is used. It projects a one-dimensional beam from a point of origin in a certain direction. The direction is controlled by the tracker of the HTC Vive. This method returns a "true" value if it hits a collider. Information about the game object with the collider can be accessed with the RaycastHit structure.
- *Rest API.* The visual content on the real whiteboard, i.e. the Metaplan cards, is generated by the sever-based *Rest API* [4]. The script for the communication with the server is attached to a game object. Other scripts can access the function to communicate with the server by referencing that game object. The game objects in Unity will thus communicate their activation to *Rest API*, which in turn will output the card's information (the written text on the card and the cluster it belongs to) to the BVIP interface.
- *Generating Sphere Colliders.* Recognition spheres are primitive sphere game objects with colliders. They are created at the intersection point of the pointing ray with the virtual screen. The spheres are generated with a predefined spawn rate 10 Hz. Each sphere has a specific "Sphere" tag attached to it with a unique ID. This is used to differentiate between sphere collider, screen collider, and note collider. The spheres have also an activation time and a lifetime to avoid any overflow with spheres that might slow down the system's responsiveness. The lifetime determines the duration after which they are deleted. The activation time is the predefined duration of 2 s for a Boolean attached to each sphere to switch from "false" to "true".

As soon as the pointing ray hits a previously generated sphere, the gesture is set as "completed" and the script will perform a gesture recognition as described above by evaluating the position of every sphere (the sampling point).

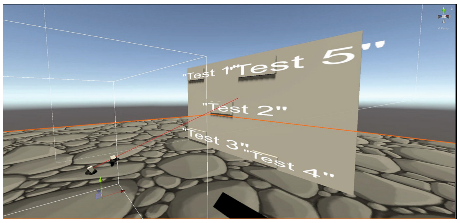

Fig. 3. Virtual screen overlay.

4.2 Technical Setup

Our application uses the HTC Vive system with two lighthouses and two trackers. With this, we generated the typical interaction space of 5×5 m. The HMD and the controllers are not used. We use the Steam VR plugin for Unity and run it together with our virtual environment on an AMD Ryzen 5 2600 six cores processor at 3.4 GHz and 8 GB RAM. The Rest API runs on an external server and communicated over the Internet.

5 User Study

Each of the three gestures will be tested ten times, together with ten false positive tests by moving randomly in the empty space on the whiteboard. For the study, the user will only see the physical whiteboard together with the artifacts.

– *Pointing gesture test.* The user starts pointing outside of the whiteboard and then moves to an artifact on the whiteboard. He points at the artifact for four seconds. Afterwards he moves outside the whiteboard again.
– *Pairing gesture test.* The user starts pointing outside the whiteboard area. Then he moves to one artifact and instantly continues moving to the second artifact. After pointing at this second artifact the user returns to the first one. This is repeated times before the gesture leaves the whiteboard.
– *Grouping gesture test.* The user starts pointing outside the whiteboard, moves inside, and encircles three artifacts. The circular movement is repeated three times. Finally, the gesture moves outside the whiteboard area.
– *False positive test.* The user starts the pointing outside the whiteboard area. He then moves his pointing direction inside an empty space on the whiteboard. The user moves for four seconds inside this empty space and ends the test by moving outside of the whiteboard area.

6 Results

The results of the user study are summarized in Table 1. The results show that the pointing gesture is detected 100% correctly, which was expected as it can be derived either from the circular or from the elliptic gesture. The recognition of parting was 90% successful, while pairing performed only 80%. The lower performance in detecting the pairing gesture might be due to the fact that the bounding box is only a rough approximation of a circular shape, which might group more cards than actually intended by the users.

Table 1. Detection accuracy for pointing, pairing, grouping and false positives.

Test no.	Pointing	Pairing	Grouping	Random
1	Yes	Yes	No	Yes
2	Yes	Yes	No	Yes
3	Yes	No	Yes	Yes
4	Yes	Yes	Yes	No
5	Yes	Yes	Yes	No
6	Yes	Yes	No	Yes
7	Yes	Yes	Yes	Yes
8	Yes	Yes	Yes	Yes
9	Yes	Yes	Yes	Yes
10	Yes	Yes	Yes	No

7 Summary and Outlook

We presented a system for the recognition of pointing, pairing and grouping gestures. Based on previous work, a new approach using the HTC Vice technology was chosen. The core idea was to create an invisible virtual overlay on the physical digital whiteboard to detect and interpret pointing gestures.

The user study examined three gestures. A comparison of the pointing trajectories on the virtual whiteboard to an ellipse was chosen for differentiating the three gestures. The curvature changes along the ellipse were used to differentiate between the pairing and grouping gesture, while pointing could be detected in both cases. The study with a reduced amount of users due to the COVID disease showed that the recognition performed well with an accuracy of at least 80%, which is comparable to deep learning approaches introduced in Sect. 2.

Future work will focus on increasing the overall performance by taking into account additional information sources. The knowledge of already existing connections between artifacts on the white help increasing the pairing accuracy even though an individual card might not be detected by out algorithm.

References

1. Alavi, A., Kunz, A.: Tracking deictic gestures over large interactive surfaces. Comput. Support. Coop. Work (CSCW) **24**(2), 109–119 (2015). https://doi.org/10.1007/s10606-015-9219-4

2. Dhingra, N., Valli, E., Kunz, A.: Recognition and localisation of pointing gestures using a RGB-D camera. In: Stephanidis, C., Antona, M. (eds.) HCII 2020. CCIS, vol. 1224, pp. 205–212. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-50726-8_27
3. Kane, S.K., Wobbrock, J.O., Ladner, R.E.: Usable gestures for blind people: understanding preference and performance. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 413–422. ACM, New York (2011). https://doi.org/10.1145/1978942.1979001
4. Koutny, R., Günther, S., Dhingra, N., Kunz, A., Miesenberger, K., Mühlhäuser, M.: Accessible multimodal tool support for brainstorming meetings. In: Miesenberger, K., Manduchi, R., Covarrubias Rodriguez, M., Peňáz, P. (eds.) ICCHP 2020. LNCS, vol. 12377, pp. 11–20. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58805-2_2
5. Kunz, A., Alavi, A., Sinn, P.: Integrating pointing gesture detection for enhancing brainstorming meetings using kinect and pixelsense. In: 8th International Conference on Digital Enterprise Technology, pp. 205–212 (2014). https://doi.org/10.1016/j.procir.2014.10.031, http://www.det-2014.de/en/home.html
6. Kunz, A., Schnelle-Walka, D., Alavi, A., Poelzer, S., Muehlhaeuser, M., Miesenberger, K.: Making tabletop interaction accessible for blind users. In: Interactive Tabletops and Surfaces, pp. 327–332. ACM, New York (2014). https://doi.org/10.1145/2669485.2669541, http://its2014.org/
7. Le, H.-A., Mac, K.-N.C., Pham, T.-A., Tran, M.-T.: Realtime pointing gesture recognition and applications in multi-user interaction. In: Selamat, A., Nguyen, N.T., Haron, H. (eds.) ACIIDS 2013. LNCS (LNAI), vol. 7802, pp. 355–364. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36546-1_37
8. Mehrabian, A., Ferris, S.R.: Inference of attitudes from nonverbal communication in two channels. J. Consult. Psychol. **31**(3), 248–252 (1967). https://doi.org/10.1037/h0024648
9. Park, C., Roh, M., Lee, S.: Real-time 3D pointing gesture recognition in mobile space. In: 2008 8th IEEE International Conference on Automatic Face Gesture Recognition, pp. 1–6. IEEE, New York (2008). https://doi.org/10.1109/AFGR.2008.4813448
10. Pölzer, S., Schnelle-Walka, D., Pöll, D., Heumader, P., Miesenberger, K.: Making brainstorming meetings accessible for blind users. In: AAATE Conference (2013)
11. Miesenberger, K., Fels, D., Archambault, D., Peňáz, P., Zagler, W. (eds.): ICCHP 2014. LNCS, vol. 8547. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08596-8
12. Wang, J., Liu, T., Wang, X.: Human hand gesture recognition with convolutional neural networks for K-12 double-teachers instruction mode classroom. Infrared Phys. Technol. **111**, 103464 (2020). https://doi.org/10.1016/j.infrared.2020.103464