Recognition and Localisation of Pointing Gestures Using a RGB-D Camera

Naina Dhingra(✉) , Eugenio Valli, and Andreas Kunz

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

Abstract. Non-verbal communication is part of our regular conversation, and multiple gestures are used to exchange information. Among those gestures, pointing is the most important one. If such gestures cannot be perceived by other team members, e.g. by blind and visually impaired people (BVIP), they lack important information and can hardly participate in a lively workflow. Thus, this paper describes a system for detecting such pointing gestures to provide input for suitable output modalities to BVIP. Our system employs an RGB-D camera to recognize the pointing gestures performed by the users. The system also locates the target of pointing e.g. on a common workspace. We evaluated the system by conducting a user study with 26 users. The results show that the system has a success rate of 89.59 and 79.92 % for a 2×3 matrix using the left and right arm respectively, and 73.57 and 68.99% for 3×4 matrix using the left and right arm respectively.

Keywords: Pointing gesture · Robot Operating System · Kinect sensor · Openptrack · Localization · Recognition · Non verbal communication

1 Introduction

In a meeting environments, when sighted people and BVIP are working together, sighted people tend to do some habitual gestures, from which the most common ones are: facial expressions, hand gestures, pointing gestures, eye gaze, etc. There are in total 136 gestures [1] which are termed as part of non-verbal communication (NVC). They need to be understood together with the verbal communication to understand the complete meaning of the conversation. However, for BVIP, the information from visual gestures are missing [4]. To understand the meaning of pointing gestures, it is crucial to know where a person is pointing at. Pointing gestures are the most common ones in nonverbal communication, and they become important in meetings where the speakers point towards objects in the room, or at artefacts on a whiteboard, as a reference to their speech. However, these pointing gestures are not accessible for BVIP and thus they lack important information during a conversation within a team meeting. To address

this issue, we developed a system that automatically detects pointing gestures and determines the position where a person is pointing at. However, although NVC is easily understood by the humans, it is difficult for machines to recognize and interpret it reliably [3] and to avoid false alerts to the BVIP.

The main contributions of this paper are as follows: (1) We developed an autonomous system using OpenPTrack and ROS (Robot Operating System) to detect and localise the position of a pointing gesture. (2) We designed our system to work in real time and performed experiments using 2×3 and 3×4 grids. (3) We conducted a user study with 26 users to evaluate our system. We expect that our work will help the researchers to integrate BVIP in team meetings.

This paper is organized as follows: Sect. 2 describes the state of the art in OpenPtrack software, pointing and related gestures. Section 3 describes the methods and techniques used in our system, while Sect. 4 gives an overview of the user study conducted and the setup of the built system. Finally, Sect. 5 discusses the results obtained from the user study as well as the accuracies obtained by our system in detecting and localizing pointing gestures.

2   State of the Art

2.1   OpenPtrack

OpenPTrack is an open source software for tracking people and calibrating a multi- RGB-D camera setup [7]. It can track multiple people at a frame rate of the sensor. It can also employ heterogeneous 3D sensors group. OpenPTrack uses a calibration procedure which depends on ROS communication networking capabilities and communication. In the past, detection and tracking systems exploited the color and depth information of a user, since cheap RGB-D sensors are available. Further, previous software were limited to single camera usage tracking system. These systems did not use multiple cameras and could not be implemented in distributed settings. Four our system, we use OpenPtrack since it allows expanding our pointing gesture system to multiple camera setup.

2.2   Pointing Gesture Recognition

Pointing gestures can be measured in different ways. Glove based techniques were used initially to sense the gesture being performed by the hand [10]. Nowadays, computer vision based techniques [3,11] or Hidden Markov Models (HMMs) [12] are used for detection. In particular, for pointing gesture detection, cascaded HMMs along with a particle filter was used for pointing gesture detection in [9]. The HMM in their first stage takes estimation of the hand position. It maps the estimated position to a precise position by modeling the kinematic features of the pointing finger. The output 3D coordinates are fed into their HMM in a second stage that differentiates the pointing gestures from other types of gesture. This technique requires a long processing time and a large training dataset.

Deep learning [6] has been successfully used in various applications of computer vision, which has inspired its use for gesture and body pose estimations

as well [8]. Deep learning approaches also solved pointing gesture recognition in [5]. However, it requires large training dataset and only works with the specific data type on which it was trained on.

Our problem statement is to solve the pointing gesture recognition for BVIP more robustly. Using deep learning approaches would have required a large training dataset to make them applicable on different setups, such as a variety of meeting room layouts with a different number of people interacting at the same time. Thus, we chose a traditional way by using mathematical geometry and feature localisation. At first, a Kinect sensor along with OpenPtrack is used to locate the body joints. Next a mathematical geometry transformation is applied to achieve the spatial position of the pointing gesture's target. This position is classified into 6 fields (for 2×3 matrix), and into 12 fields (for 3×4 matrix).

3   Methodology

The implementation of our pointing gesture recognizer and localizing system is based on OpenPTrack [2]. Using Kinect v2 as sensor, this software allows person tracking in real time over large areas. However, since this framework is not capable of directly detecting pointing gestures or other behavioral features of a user, we also forward the data to ROS[1]. By doing so, we can obtain the joints' coordinates in space for human gestures such as pointing. The main idea is that different packages of ROS could be implemented that contain so-called nodes, which are units that perform logic and computation for different parts of a robot, i.e. control of actuators, transform or change resolution of images provided by a sensor, etc. The different nodes of ROS can communicate with each other in order to share useful information or the functioning of the whole system, which is done by the *topics*. Every node implementation of a pointing gesture recognizer for blind users can subscribe to such a topic to receive information or publish on a topic to share its content. OpenPtrack thus uses ROS to allow the information provided by Kinect to be further processed. The joints x, y and z coordinates are published under a repository which also contains different IDs for the different joints. The coordinate transformation from the sensor's reference frames to the world reference frame are performed using a ROS package called/TF, which rotates and translates the reference frames to the desired positions.

A deictic (or pointing) gesture consists of the movement of an arm to point at a target in space and to highlight it by this gestures for other people without necessarily having to verbally describe its position exactly. The joint's coordinates that have to be obtained are thus from elbow and hand, since these represent the human forearm and hence the major components for pointing. In order to define a pointing action, the link connecting the two aforementioned joints was measured, and named pointing vector as shown in the Fig. 1.
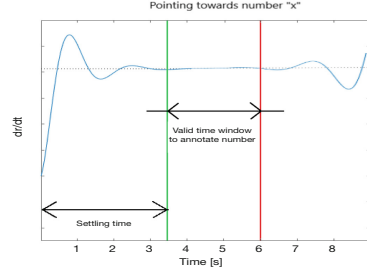

Fig. 1. Left: Pointing gesture with pointing vector. Right: Stabilization time of a pointing gesture where dr/dt is the change in the circle's diameter.

Equation 1 is used to locate the position of the target on a vertical plane (e.g. a whiteboard) the user is pointing at.

$$P_p = H + \frac{(H - P_f) \cdot \boldsymbol{N}_f}{\boldsymbol{EH} \cdot \boldsymbol{N}_f} \cdot \boldsymbol{EH},$$ (1)

where P_f is a point predefined on the ground plane, $\boldsymbol{N}_f$ is the normal vector to the plane and H, E are the positions of hand and elbow joints, respectively.

OpenPtrack defines all measurements in a world reference frame. To understand the definition of the world reference frame in OpenPtrack, the TF package of ROS is used, which is a predefined package for coordinate transformation using rotation matrices and quaternions. The next step is to define a whiteboard/matrix plane coordinate frame in order to obtain the measured target point on it. This plane coordinate frame is achieved by applying the rotation matrix in relation world coordinate frame. The output values from OpenPtrack are converted in whiteboard/matrix plane coordinate frames. These converted values analysed by putting hard limits for each box in the matrix along both x and y direction. All of these values are evaluated on run time.

Before getting the information from the system, on which target a user is pointing at, it is required to wait about 3–3.5 s for the pointing gestures to get stabilized. This waiting time is for a user to reach the stable pointing gesture without moving or vibrating his/her arm. The stability output from the system is achieved after the setting time as shown in Fig. 1. It also has to be noted that the pointing gesture will become unstable again after certain time period.

4   Experimental Setup

The setup resembles an environment in which sighted users have to perform pointing gestures, which are automatically recognized by our system. The pointing gesture's target will then be determined to be provided to a suitable output device for BVIP. The experiments consist of four parts: two studies using the

left arm and two using the right arm for pointing. The pointing gestures have to be performed on two different grid sizes in order to evaluate the accuracy of our system, i.e., at each 2×3 and 3×4 grid printed on the board.

The setup is shown in Fig. 2. The board has the dimensions $1290 \, \text{mm} \times 1910 \, \text{mm}$ and was $1000 \, \text{mm}$ above the ground. The Kinect sensor was placed at a height of $1300 \, \text{mm}$ above the top edge of the board and centered. Each box in the grid was numbered. The setup is shown in 2. The user had to stand at a constant distance of $1.5 \, \text{m}$ and centered in front of the board. Then he was asked to point towards the numbers following a given sequence told by the experimenter, and point for a few seconds to achieve a stable gesture before moving to the next number in the sequence. The stability time procedure is illustrated in Fig. 1. After the user was prompted to point at a certain box and the wait-time from Fig. 1 was exceeded, the measured target number was recorded.

Fig. 2. Left: Measurement setup; Right: Experimental setup of the system. The Kinect is placed above the board having the matrix of numbers for the user to point at.

5 User Study and Results

The system was evaluated in a user study with 26 participants. Different parameters such as handedness, user's height, and arm length were measured. Since a user's pointing is significantly influenced by the pointing stability, this also impacts the accuracy of our system, resulting in noticeable differences for the 2×3 and the 3×4 grids. The error increases with decreasing box sizes, i.e. it is larger for the 3×4 grid. The confusion matrix in Fig. 3 left gives an overview on the percentage of the correct pointing at a target number in the 2×3 matrix using the left arm. Similarly, Fig. 3 (right) describes the quantitative values for right arm corresponding to 2×3 matrix, Fig. 4 for the left arm pointing at 3×4 matrix and Fig. 5 for the right arm pointing at 3×4 matrix.

2x3 L	N.A.	1	2	3	4	5	6
1	16 12.30%	110 84.60%	3 2.30%	0 0.00%	1 0.80%	0 0.00%	0 0.00%
2	2 1.50%	4 3.20%	113 86.90%	2 1.50%	2 1.50%	6 4.60%	1 0.80%
3	4 3.80%	0 0.00%	4 3.80%	83 79.80%	0 0.00%	6 5.80%	7 6.80%
4	5 3.00%	0 0.00%	0 0.00%	1 0.60%	150 96.40%	0 0.00%	0 0.00%
5	1 0.80%	0 0.00%	0 0.00%	0 0.00%	2 1.50%	125 96.20%	2 1.50%
6	7 5.40%	0 0.00%	0 0.00%	2 1.50%	0 0.00%	0 0.00%	121 93.10%

2x3 R	N.A.	1	2	3	4	5	6
1	10 7.70%	118 90.70%	1 0.80%	0 0.00%	1 0.80%	0 0.00%	0 0.00%
2	8 6.20%	12 9.20%	103 79.20%	1 0.80%	3 2.30%	3 2.30%	0 0.00%
3	7 5.40%	0 0.00%	7 5.40%	114 87.70%	0 0.00%	0 0.00%	2 1.50%
4	22 16.90%	8 6.20%	0 0.00%	0 0.00%	99 76.10%	0 0.00%	1 0.80%
5	4 3.10%	0 0.00%	2 1.50%	0 0.00%	30 23.10%	94 72.30%	0 0.00%
6	1 0.80%	0 0.00%	3 2.30%	16 12.30%	0 0.00%	0 0.00%	110 84.60%

Fig. 3. Pointing accuracy for the left/right arm using a 2×3 grid.

3x4 L	N.A.	1	2	3	4	5	6	7	8	9	10	11	12
1	10 19.30%	42 80.70%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%
2	3 5.70%	3 5.70%	37 71.40%	3 5.70%	0 0.00%	3 5.70%	0 0.00%	2 3.90%	0 0.00%	0 0.00%	1 1.90%	0 0.00%	0 0.00%
3	4 7.70%	0 0.00%	4 7.70%	36 69.20%	3 5.70%	0 0.00%	4 7.70%	1 1.90%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%
4	8 15.40%	2 3.80%	2 3.80%	2 3.80%	37 71.40%	0 0.00%	0 0.00%	1 1.90%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%
5	9 17.30%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	43 82.70%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%
6	2 3.90%	0 0.00%	2 3.90%	0 0.00%	0 0.00%	4 7.70%	30 57.40%	0 0.00%	0 0.00%	2 3.90%	10 19.30%	2 3.90%	0 0.00%
7	0 0.00%	0 0.00%	0 0.00%	2 3.90%	0 0.00%	1 1.90%	2 3.90%	39 75.00%	0 0.00%	0 0.00%	3 5.70%	3 5.70%	2 3.90%
8	0 0.00%	0 0.00%	0 0.00%	4 7.70%	5 9.60%	0 0.00%	0 0.00%	1 1.90%	33 63.50%	0 0.00%	0 0.00%	6 11.60%	3 5.70%
9	21 40.40%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	31 59.60%	0 0.00%	0 0.00%	0 0.00%
10	1 1.90%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	1 1.90%	48 92.30%	2 3.90%	0 0.00%
11	1 1.90%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	3 5.70%	1 1.90%	0 0.00%	0 0.00%	9 17.30%	38 73.20%	0 0.00%
12	3 5.70%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	2 3.90%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	1 1.90%	46 88.50%

Fig. 4. Pointing accuracy for the left arm using a 3×4 grid.

Table 1 shows the accuracy values achieved in the four experiments, i.e., (1) Left arm using 2×3 matrix, (2) Right arm using 3×4 matrix, (3) Left arm for 3×4 matrix and (4) Right arm for 3×4 matrix. This accuracy is calculated by converting the output from the system to binary output, i.e., 1, if the output from the system was correct, otherwise it is 0. Then the total number of correct results is divided by the total of trials in the experiment multiplied by 100 to get the percentage of the accuracy.

Each of the four tests resulted in a higher accuracy when using the left arm for pointing. This could be caused by the inherent asymmetry within Kinect v2. The IR emitter is centered in the Kinect box, while the IR receiver is off-centered. This leads to a camera's perspective that sees the left arm slightly better than the right one, i.e. the left arm is measured slightly longer than the right one.

6   Conclusion

We worked on automatic pointing gesture detection and pointing target localization in a meeting environment. A prototype of the automatic system was built

3x4 R	N.A.	1	2	3	4	5	6	7	8	9	10	11	12
1	6 11.60%	42 80.80%	3 5.70%	0 0.00%	0 0.00%	1 1.90%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%
2	16 30.70%	2 3.90%	32 61.50%	0 0.00%	0 0.00%	0 0.00%	2 3.90%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%
3	12 23.10%	0 0.00%	4 7.70%	31 59.60%	3 5.70%	0 0.00%	0 0.00%	0 0.00%	2 3.90%	0 0.00%	0 0.00%	0 0.00%	0 0.00%
4	9 17.30%	0 0.00%	0 0.00%	0 0.00%	43 82.70%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%
5	7 13.60%	1 1.90%	0 0.00%	0 0.00%	0 0.00%	40 76.90%	0 0.00%	0 0.00%	0 0.00%	3 5.70%	1 1.90%	0 0.00%	0 0.00%
6	0 0.00%	4 7.70%	5 9.60%	2 3.90%	0 0.00%	8 15.40%	28 53.80%	0 0.00%	0 0.00%	4 7.70%	0 0.00%	0 0.00%	1 1.90%
7	0 0.00%	1 1.90%	6 11.60%	0 0.00%	3 5.70%	2 3.90%	1 1.90%	33 63.50%	0 0.00%	0 0.00%	1 1.90%	3 5.70%	2 3.90%
8	2 3.90%	0 0.00%	0 0.00%	2 3.90%	21 40.40%	0 0.00%	0 0.00%	1 1.90%	23 44.20%	0 0.00%	0 0.00%	0 0.00%	3 5.70%
9	9 17.30%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	4 7.70%	0 0.00%	0 0.00%	0 0.00%	36 69.20%	2 3.90%	0 0.00%	1 1.90%
10	2 3.90%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	14 26.90%	36 69.20%	0 0.00%	0 0.00%
11	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	2 3.90%	0 0.00%	0 0.00%	0 0.00%	8 15.40%	42 80.70%	0 0.00%
12	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	4 7.70%	0 0.00%	0 0.00%	0 0.00%	48 92.30%

Fig. 5. Pointing accuracy for the right arm using a 3×4 grid.

Table 1. Accuracy for the experiments performed in the user study by using 2×3 and 3×4 matrix and by using left and the right arm.

	Left arm	Right arm
2×3	89.59%	79.92%
3×4	73.57%	68.99%

and tested by conducting a user study. The output of this system will be converted to suitable modality which will help BVIP to get the extra information. Although for our application it is required to have its good performance for 2×3 but it proves to have high precision for small areas for localizer function and performs good for both 2×3 and 3×4 grids in all the four the experiments. We also found out that the stable time for getting the value of localizer is achieved after around 3 s and the hand of the user starts to vibrate after an interval again. Our user study also showed that the height of the user did not effect much on the performance. The arm size which is either very small or very large has a small decrease in the accuracy.

In future, the output of our system will be converted by a suitable haptic interface helping BVIP to access these pointing gestures. Moreover, we will expand our system with multiple cameras, and we will have several users pointing simultaneously. Also, we will improve the system to a have more symmetrical output, i.e., the same performance for pointing using the left and the right arm.

References

1. Brannigan, C.R., Humphries, D.A.: Human non-verbal behavior, a means of communication. Etholog. Stud. Child Behav., 37–64 (1972)
2. Carraro, M., Munaro, M., Burke, J., Menegatti, E.: Real-time marker-less multi-person 3D pose estimation in rgb-depth camera networks. In: Strand, M., Dillmann, R., Menegatti, E., Ghidoni, S. (eds.) IAS 2018. AISC, vol. 867, pp. 534–545. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-01370-7_42
3. Dhingra, N., Kunz, A.: Res3atn-deep 3D residual attention network for hand gesture recognition in videos. In: International Conference on 3D Vision (3DV 2019), pp. 491–501. IEEE (2019)
4. Günther, S., et al.: Mapvi: meeting accessibility for persons with visual impairments. In: Proceedings of the 12th ACM International Conference on PErvasive Technologies Related to Assistive Environments, pp. 343–352. ACM (2019)
5. Huang, Y., Liu, X., Zhang, X., Jin, L.: A pointing gesture based egocentric interaction system: dataset, approach and application. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 16–23 (2016)
6. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436 (2015)
7. Munaro, M., Basso, F., Menegatti, E.: Openptrack: open source multi-camera calibration and people tracking for RGB-D camera networks. Robot. Auton. Syst. **75**, 525–538 (2016)
8. Neverova, N., Wolf, C., Taylor, G.W., Nebout, F.: Multi-scale deep learning for gesture detection and localization. In: Agapito, L., Bronstein, M.M., Rother, C. (eds.) ECCV 2014. LNCS, vol. 8925, pp. 474–490. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-16178-5_33
9. Park, C.B., Lee, S.W.: Real-time 3D pointing gesture recognition for mobile robots with cascade HMM and particle filter. Image Vis. Comput. **29**(1), 51–63 (2011)
10. Quam, D.L.: Gesture recognition with a dataglove. In: IEEE Conference on Aerospace and Electronics, pp. 755–760. IEEE (1990)
11. Rautaray, S.S., Agrawal, A.: Vision based hand gesture recognition for human computer interaction: a survey. Artif. Intell. Rev. **43**(1), 1–54 (2012). https://doi.org/10.1007/s10462-012-9356-9
12. Wilson, A.D., Bobick, A.F.: Parametric hidden markov models for gesture recognition. IEEE Trans. Pattern Anal. Mach. Intell. **21**(9), 884–900 (1999)