Precise Pointing Target Recognition for Human-Robot Interaction

Thomas Breuer, Paul G. Ploeger, Gerhard K. Kraetzschmar

Bonn-Rhein-Sieg University of Applied Sciences,
Grantham Allee 20, 53757 Sankt Augustin, Germany
{thomas.breuer@gmx.net}

Abstract. This work presents a person independent pointing gesture recognition application. It uses simple but effective features for the robust tracking of the head and the hand of the user in an undefined environment. The application is able to detect if the tracking is lost and can be reinitialized automatically. The pointing gesture recognition accuracy is improved by the proposed fingertip detection algorithm and by the detection of the width of the face.
The experimental evaluation with eight different subjects shows that the overall average pointing gesture recognition rate of the system for distances up to 250 cm (head to pointing target) is 86.63% (with a distance between objects of 23 cm). Considering just frontal pointing gestures for distances up to 250 cm the gesture recognition rate is 90.97% and for distances up to 194 cm even 95.31%. The average error angle is 7.28°.

Keywords: Pointing Gesture Recognition, Pointing Gesture Detection, HRI, Human-Robot Interaction, Gesture Recognition, Gesture-based HRI

1 Introduction

Robots which may help in the household will have to be controlled from laymen and not from experts, which necessitates a simple and easy to use HRI. Especially pointing gestures are a promising and natural way for the interaction with a robot. Jojic et al. [1] even state that from natural human gestures, the pointing gestures are the easiest to interpret. For the interaction with a robot, pointing gestures could indicate objects and locations. It is easier and more accurate to point at an object than to verbally describe the object itself or its location [2]. Pointing gestures are especially useful for technical inexperienced users, since pointing is an every day life operation and the user does not need to have a priori knowledge about the system [3]. However, pointing gestures are difficult to recognize [4]. The difficulty is to detect the precise 3D positions of the face and the fast moving hands of an unknown user in front of a dynamic background under unknown and changing lighting conditions. This paper describes all steps for the implementation of a person independent pointing gesture recognition application using a stereo vision camera. The main achievement in this paper is the improvement of the pointing target recognition accuracy by the proposed fingertip detection algorithm and the detection of the width of the face.

Proceedings of SIMPAR 2010 Workshops
Intl. Conf. on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS
Darmstadt (Germany) November 15-16, 2010
ISBN 978-3-00-032863-3
pp. 229-240

2 Related Work

Most modern approaches for gesture detection are computer vision-based. The advantage from computer vision-based approaches is, that they are cost effective and minimal obtrusive to the user [5]. This work uses a low-level feature-based approach [5]. A pointing gesture recognition does not require the complete modeling of the body, such that the complexity of the gesture can be reduced to low-level features [5]. The advantages of low-level features like color are that they are robust to (partial) occlusion, rotation of the object, and also are scale invariant and therefore independent of the object's distance from the camera. This advantages are especially relevant for non-rigid objects.

Independent of the color space, there are some general problems when it comes to the detection of body parts. One problem is that there is not always color constancy in the scene. The appearance of color may change when light changes from ambient to bright light or when shadows occur [6]. Also the skin tone varies within and across individuals [6]. Nevertheless, several studies showed that skin color has a compact distribution in chromaticity space and is just differing in brightness and intensity [7]. One main problem is the difficulty to differ between hands and objects which have the same color characteristics as hands. Such objects could be of wood, leather, paperboard, or just have the same color as skin.

Any kind of *Bayesian methods* are used for tracking. For example Bennewitz et al. [8] first locate hands and faces in the image and then update a probabilistic belief of each object (hands and faces) to track them. Also depth information could be (additionally) used to track some objects / body parts. Seemann et al. [9] state that the usage of depth information makes it easier to track a face in an image sequence. One easy way to perform color tracking (e.g. skin color) is the usage of the CAMSHIFT (Continuously Adaptive Mean Shift) algorithm, as used in [10].

The detection where a person points at is solved by a human intuitively, but is not as easy solvable by a computer. Detecting the line of sight starting from the eyes through the fingertip will provide the better recognition accuracy compared to the detection of the extension line of the forearm [11] [12] [13]. The task is now to detect the 3D coordinates of the face / eyes and of the hand / fingertip. One of the most promising works in the area of pointing gesture recognition is the work from Nickel and Stiefelhagen [13]. For the detection of the users head and hand they use a combination of color and disparity information. They build two color histograms (one for skin and one for not skin). To track the hands and head, the color and disparity information are integrated into a multi-hypothesis tracking framework. The pointing gesture itself is then detected by a Hidden Markov Model-based pointing gesture recognizer. They further state that their system has an pointing gesture recognition accuracy of about 25°-30° at a distance of 2 meters from the pointing target.

This paper describes in section "Optimization" how this algorithms can be improved to receive a higher pointing gesture recognition accuracy.

Proceedings of SIMPAR 2010 Workshops
Intl. Conf. on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS
Darmstadt (Germany) November 15-16, 2010
ISBN 978-3-00-032863-3
pp. 229-240

3 Requirements

There are some restrictions especially for the skin detection in an image . For example wood, paperboards, and leather have equal color characteristics as skin. Jojic et al. [1] state that skin color detection is sensitive to lighting changes and depends on the background color and the subject's clothing color. Also the skin tone varies within and across individuals [6].

The requirements for a good pointing gesture recognition application for a household robot are (some mentioned in [14]):

- Robustly detect the face in the image
- Robustly detect the hands in the image (includes: being not bothered by skin colored objects in the area of feasible arm positions; algorithm should work wearing a t-shirt; failure detection of tracking algorithm (detect when tracked hand is lost)
- Receive precise 3D position of face and hands to estimate pointing direction
- Work person independent
- Cope with different skin colors without manual retraining
- Cope with variable and complex backgrounds
- Cope with skin colored regions in the background (like wood, paperboard, leather)
- Cope with various lighting conditions
- Cope with changing lighting conditions (during runtime)
- Real-time performance
- Work on a normal PC
- Easy to integrate

4 Hardware and Software Dependencies

As hardware for the implementation a BumbleBee stereo vision camera and a MacBook with 2 GHz Intel Core 2 Duo and 2 GB RAM is used. The whole application is implemented from scratch and can easily be reimplemented. Its strength is the usage of simple features and therefore the low needed processing power. With the above mentioned hardware the application runs with around 7.1 Hz.

Since the application is implmemented from scratch there are very few software dependencies (namely on OpenCV for all image processing, the Bumblebee libraries, IVT for the capturing of the BumbleBee stereo vision camera, Boost to make the application multithreading ready and ICE for the distributed communication, if parts of the application are running on a distributed machine). The clear interface to the depth image device makes the replacement of the stereo vision camera by another camera (like a time-of-flight camera) easily possible.

Proceedings of SIMPAR 2010 Workshops
Intl. Conf. on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS
Darmstadt (Germany) November 15-16, 2010
ISBN 978-3-00-032863-3
pp. 229-240

5 Pointing Target Detection

The target of a pointing gesture lies on the line-of-sight between the center of the eyes and the fingertip of the pointing person [13]. This line-of-sight can be represented as a vector in the 3D space (V_{hand} in figure 1). Ideally this vector should directly point on the 3D position of the pointed object. Due to measurement errors and the simplification of the pointing gesture, the line-of-sight vector will point to another direction with an error angle. This error angle α will be calculated between every object and every visible hand. The hand and the object which together have the smallest α are supposed to be the pointing hand and pointing target. The angle α between V_{hand} and V_{object} can be calculated by:

$$\alpha = arccos \frac{V_{hand} * V_{object}}{|V_{hand}||V_{object}|}$$

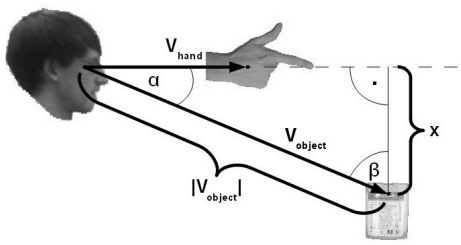


Fig. 1. Pointing gesture recognition

Figure 2 shows the performed actions in the gesture recognition software. Before being able to track face and hand and detect a dynamic gesture, the initialization of face- and hand-tracker is necessary.

5.1 Face Detection and Tracking

For the detection of the face two different methods are implemented. One method is based on Haar-like features. The other method is based on the commercial face recognition software from L-1 Identity Solutions. The face recognition software from L-1 Identity Solutions has a less false positive detection rate, but needs a high resolution image to robustly detect a face. Therefore it is more accurate, but will not detect a face with the stereo vision camera which is far away (e.g. 3m). Depending on the situation the choice of the algorithm can be set in a configuration file without recompilation.

After the face is found in the image, it is converted into the HSV color space. The reason for the transformation in the HSV color space is that the saturation and value change (the hue keeps roughly constant) when the lighting conditions

Proceedings of SIMPAR 2010 Workshops
Intl. Conf. on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS
Darmstadt (Germany) November 15-16, 2010
ISBN 978-3-00-032863-3
pp. 229-240

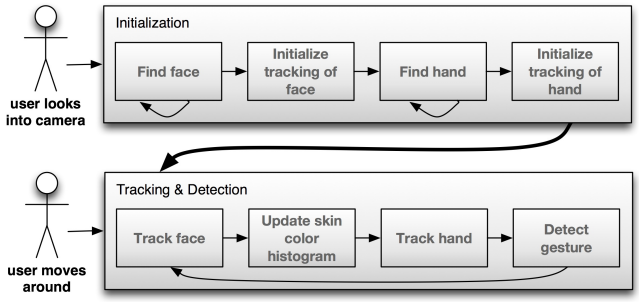


Fig. 2. Actions in dynamic gesture recognition software

change. The *hue* values in the image are extracted and a two-dimensional histogram of the region of the face is build. This histogram represents the skin colors as probability values. From this histogram the backprojection image (or skin color probability image) is calculated. The backprojection image is binarized with a threshold and then morphological operations (erosion followed by dilation) are performed to get rid of outliers in the backprojection image.

The procedure to track the face is as follows: A 2D track-box is defined around the last known face position and all pixels outside this 2D track-box are deleted. Then a 3D track-box is defined around the last known 3D face position. Every pixel, whose 3D position lies outside this 3D track-box is deleted in the 2D image. The 2D trackbox of the face is always set to be a square defined by the width of the face to prevent the face region to grow towards skin colored regions on the clothes of the user. After deleting all pixels outside the 2D and 3D track-boxes, the hand is tracked in the 2D image via the CAMSHIFT algorithm.

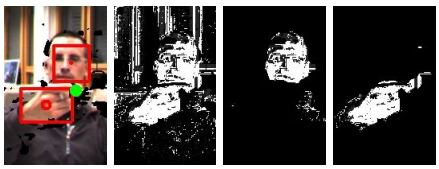
5.2 Hand Detection and Tracking

The hand position for the initialization of the hand tracking is strictly defined. It is defined that the hand which should be tracked has to be somewhere in front of the chest, but not between the face and the stereo camera (such that face keeps visible for initialization). The hand should not be very close to the chest, but a few cm towards the stereo camera (such that initial hand detection is not influenced by skin colored areas on the t-shirt. All pixels which are not in this defined region are deleted and also all pixels, whose 3D position is an impossible hand position (too far away from the head) is deleted. The remaining biggest skin colored blob is supposed to be the hand which should be tracked.

The tracking of the hand is similar to the tracking of the face. Again a 2D and a 3D trackbox of the last known hand position is defined. The 2D trackbox is always set to the size of the trackbox of the face. This avoids that the trackbox of the hand can grow over a bigger area (e.g. if other skin colored areas are next to the hand). After deleting all pixels outside the 2D and 3D track-boxes, the hand is tracked in the 2D image via the CAMSHIFT algorithm.

Proceedings of SIMPAR 2010 Workshops
Intl. Conf. on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS
Darmstadt (Germany) November 15-16, 2010
ISBN 978-3-00-032863-3
pp. 229-240

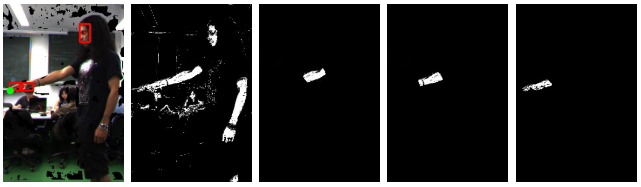
Figure 3 shows why it is important to use a 3D track-box additionally to the 2D track-box. Sometimes it happens that skin colored areas are *overlapping* in the 2D image, but do not belong to each other. In this cases they still have to be separable. Figure 3(a) shows the track-boxes of the hand and of the head, figure 3(b) shows the backprojection image where the skin colored areas overlap in the 2D image, figure 3(c) shows that the tracked face is not affected by this overlap and figure 3(d) show that the tracked hand is also not affected by this overlap.



(a) (b) (c) (d)

Fig. 3. Depth clustering of the 2D backprojection image

If the user is wearing a t-shirt instead of a long shirt, there will be also the skin colored pixels from the arm in the track-boxes. The 2D CAMSHIFT algorithm is not able to detect the hand anymore and will detect some area on the arm. In this detected area it is searched for the from the head farthest away skin colored pixel. This pixel should be nearer to the hand (if hand is not already detected). All pixels farther away than 15 cm of the from this pixel are deleted (in a copied image). If not the hand but some area of the arm is detected, this deletion will result in keeping skin colored pixels nearer to the hand. In the resulting image it is searched again for hand region pixels. The new found hand region is now nearer to the hand than before (if hand was not already detected). The last to above mentioned steps are repeated for two more times to finally detect the correct hand region. Figure 4 shows the process of the algorithm until the hand is found in figure 4(e).



(a) (b) (c) (d) (e)

Fig. 4. Tracking of the hand in t-shirt

Proceedings of SIMPAR 2010 Workshops
Intl. Conf. on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS
Darmstadt (Germany) November 15-16, 2010
ISBN 978-3-00-032863-3
pp. 229-240

In the proposed implementation it is detected whether the hand of the user is still successfully tracked. The detection is done by continuously validating that the tracked hand is still at an anatomically possible position. If it is detected that the detected hand is at an impossible position (e.g. hand is in 3D region of the face or too far away from the face), a counter is increased. The counter is reset if a valid hand position is detected again. If the counter is continiously increased for about 3 seconds, the hand is considered to be lost and the tracking is stopped and could automatically be restarted.

To cope with changing lighting conditions during runtime, the skin color histogram has to be updated during runtime. To do so the histogram of the skin colored area of the tracked face is extracted every frame. This extracted histogram is saved and used as new histogram. Not the complete area of the tracked face is used to update the histogram, but only half of the area in the center of the tracked face. The reason is to be sure not to include the color of the hair and other non skin colored areas in the face region. Since this algorithm is just needed in an environment with frequently changing lighting conditions, the algorithm can be turned on and off via a variable in the configuration file without recompiling.

6 Optimization

6.1 Real-time Performance

One very important defined requirement is that the tracking of the face and hands has to perform in real-time. Before doing the following described optimization steps, the application has an overall processing time of around 381 ms (ca. 2.6 Hz), which is way to slow to track a moving hand. By *decreasing the disparity range* of the stereo vision camera from 240 to 100 and *turning off all validation functions besides the surface validation function*, the processing time for just the capturing and stereo processing of the stereo vision image can be decreased from around 240 ms to around 125 ms. With a disparity range of 100 objects with distances farther than 70 cm can be detected. Nearer objects can not be detected correctly anymore, such that the disparity range is not further decreased. By decreasing the disparity range, the processing time decreases to 266 ms. For every depth calculation (often performed for every pixel of an area or even of the whole image) an API call to the stereo vision library is made. This API calls are very slow, such that it is decided to *store the depth data of the whole image in an own data structure* (matrix of 3D vectors). By doing so the API call to the sereo vision library has to be done just a single time for every pixel, such that the processing time of the application can be decreased to around 204 ms. By *making the application multithreaded ready* (i.e. the caputing + stereo processing of the image in parallel with all other image processing), both cores of the Macbook can be used efficiently, such that the processing time of the application can be decreased to around 141 ms, which is enough to robustly track a moving face and hand. So all in all the above mentioned optimization

Proceedings of SIMPAR 2010 Workshops
Intl. Conf. on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS
Darmstadt (Germany) November 15-16, 2010
ISBN 978-3-00-032863-3
pp. 229-240

steps could decrease the processing time of the application from around 385 ms to around 141 ms.

Optimization	Time saving	Total processing time
Without optimization		381 ms
Parameter tuning of stereo camera	115 ms	266 ms
Store depth data in own data structure	62 ms	204 ms
Using multiple threads	63 ms	141 ms

Table 1. Timing optimization of the implementation

6.2 Improvement of Pointing Target Detection

Most approaches in the related works detect the pointing target by using the line-of-sight between the face and the detected hand, instead of using the face and the fingertip. Figure 5 shows the problem of the detection of the pointing direction when detecting the center of the hand instead of detecting the fingertip for the detection. For the two different hands, two complete different hand positions will be recognized. The outer arrows show what the recognition system will detect if the user points with the left hand or the right hand. In contrast to that, if the fingertip instead of the center of the hand is recognized, there is not so much difference anymore, whether the user uses his right or left hand.

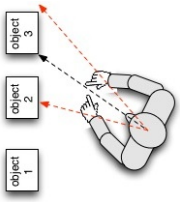


Fig. 5. Pointing with the fingertip

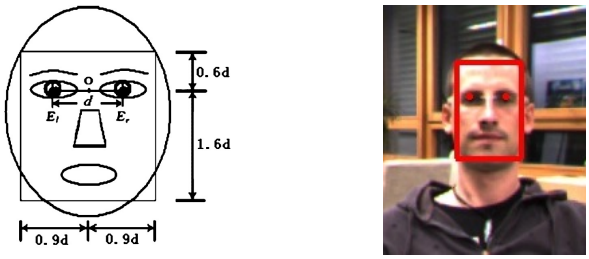
The algorithm to detect the fingertip uses the tracked hand and model knowledge. In the found skin colored blob, the fingertip is the from the head farthest pixel (in the 3D space). This should hold true at least for an outstretched arm during a pointing gesture. To find the fingertip, all pixels which are farther than 15 cm from the detected center of the hand are deleted (anatomically no possible fingertip positions). Afterwards for every remaining skin colored pixel in a bounding box around the center of the hand the distance to the detected center of the head is calculated. The pixel with the biggest distance to the center of the head is supposed to be the fingertip. The fingertip is often pretty thin in

Proceedings of SIMPAR 2010 Workshops
Intl. Conf. on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS
Darmstadt (Germany) November 15-16, 2010
ISBN 978-3-00-032863-3
pp. 229-240

the backprojection image and the skin colored blob of the fingertip is not always connected to the rest of the hand region. Therefore it is important that the fingertip is searched in the backprojection image without any morphological operations applied, since otherwise the fingertip may be deleted through this operations.The evaluation showed that the fingertip detection works correctly for all subjects in most of the cases (in around 92.5% of all pointing gestures).

As already described, the pointing target lies on the line of sight between the fingertip and the center of the head. The stereo camera will not be able to detect the center of the head, but will detect the cheek or some other face area instead. Therefore the pointing target recognition can be improved by adding half of the width of the face to the measured face depth. Deng et al. [15] present a facial model which shows the correlation between the eye distance and the width of the face. This model is shown in figure 6(a). From this model two important formulas result:

- width of face = eye distance * 1.8
- height of face = eye distance * 2.2



(a) Facial model (b) Eyes detection

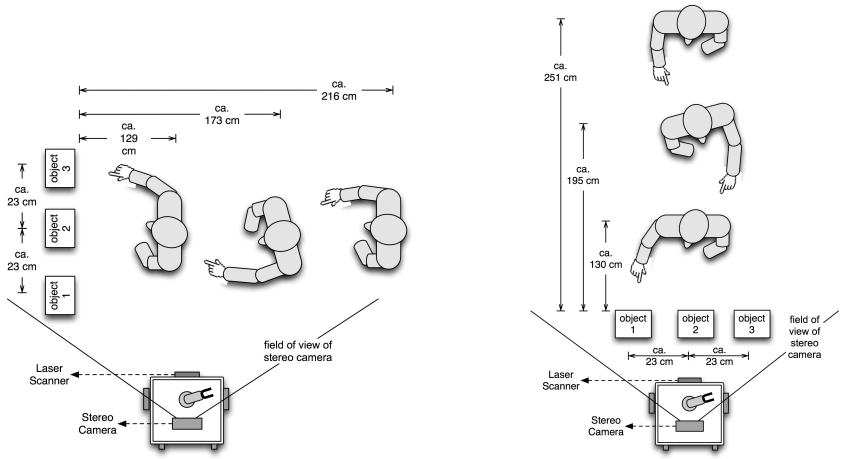
Fig. 6. Correlation eye distance and width of face [15]

Knowing the correlation between the eye distance, it is possible to calculate the width of the face. The commercial face recognition software from L-1 Identity Solutions already provides the eye positions. With the 2D eye positions, the 3D eye positions are measured and the distance between the eyes are calculated. With this distance the width of the face can be calculated with the above showed formula. The face detection which is based on Haar-like features does only provide a bounding box of a face. Nevertheless, with the bounding box of the face and the above formulas, the eye positions can be estimated. The result is shown in figure 6(b). For this approximated eye positions the 3D coordinates are calculated and equivalently as described above the width of the face can be measured. The evaluation shows that adding half of the width of the face to the depth measurement of the face improves the gesture recognition rate compared to adding an approximated value of 5 cm (slightly improvement) or nothing (good improvement).

Proceedings of SIMPAR 2010 Workshops
Intl. Conf. on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS
Darmstadt (Germany) November 15-16, 2010
ISBN 978-3-00-032863-3
pp. 229-240

7 Results

To test the robustness of the implemented pointing gesture recognition application, eight different subjects repeated a pointing gesture on different objects from different positions and distances. The two main setups (pointing frontal towards the camera and pointing sideways in the field of view of the camera) are visualized in figure 7. For each setup the pointing person repeats pointing on all three objects in a row for four times (two times with the right hand and two times with the left hand). Afterwards the pointing person repeats this step for a different distance. The pointing person does not receive any feedback whether the pointing gesture was correctly detected.



(a) Pointing sideways (b) Pointing frontal

Fig. 7. Experimental evaluation setup

For each pointing position the pointing gesture recognition rate (how often the gesture is correctly detected), the average distance to the correct object in degree, and the average distance to the correct object in cm is evaluated. The values correspond to α and $\boldsymbol{x}$ in figure 1. Figure 8 shows a diagram with the comparison of the results for the usage of different methods (differing between sideways and frontal pointing). "NoFingertipNoHead" means, that the fingertip algorithm is not applied (i.e. center of the hand is detected) and nothing is added to the detected depth of the face. "Fingertip5cmHead" means that the fingertip detection algorithm is used and that 5 cm are added to the detected depth of the face. Other methods are named similar.

The diagram shows that for sideways pointing (blue bars) the method without using the fingertip but adding either 5 cm (*NoFingertip5cmHead*) or half of the width of the face to the depth of the face (*NoFingertipHalfWidthFace*) provides

Proceedings of SIMPAR 2010 Workshops
Intl. Conf. on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS
Darmstadt (Germany) November 15-16, 2010
ISBN 978-3-00-032863-3
pp. 229-240

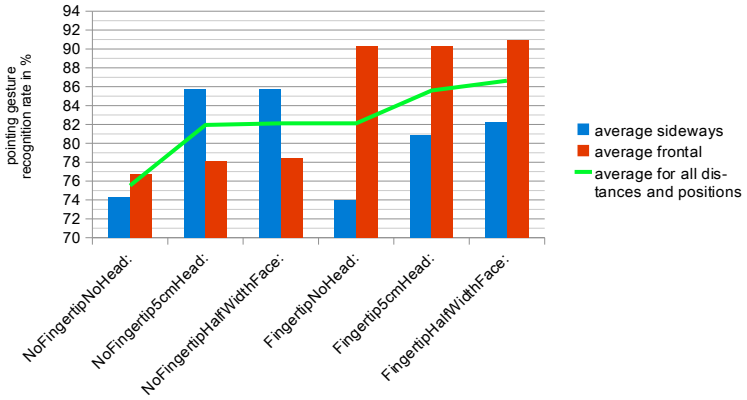


Fig. 8. Comparison of methods

the best result (in average 85.76%). Nevertheless, for frontal pointing (red bars) and the average over all pointing positions (green line) the method which uses the fingertip and adds the half of the width of the face to the depth value of the face (*FingertipHalfWidthFace*) has the best results (for frontal pointing 90.97% and in average over all pointing gestures 86.63%). The recognition accuracy decreases with an increasing distance between the subject and the pointing target. The effect is stronger for sideways pointing gestures. Frontal pointing provides more robust results.

8 Conclusions

The presented person independent pointing gesture recognition application is able to cope with different skin colors (people from China, Mexico, Germany and India participated in the evaluation), variable and complex backgrounds (background of the evaluation was a crowded lab with moving people), skin colored regions in the background, and with various lighting conditions (evaluation was done under artificial lighting conditions, but system is also successfully tested outside under sunlight). The proposed implementation is able to robustly track a hand in a t-shirt under dynamically changing lighting conditions, is able to detect when the tracking is lost and can be reinitialized automatically. Therefore all defined requirements are met.

The pointing gesture recognition rate is improved by the proposed fingertip detection algorithm and by the detection of the width of the face (and adding it to the measured depth value of the face).

The experimental evaluation with eight different subjects shows that the overall average pointing gesture recognition rate of the system for distances up to 250 cm (head to pointing target) is 86.63%. Considering just frontal pointing gestures for distances up to 250 cm (head to pointing target) the gesture recognition rate is 90.97% and for distances up to 194 cm (head to pointing target)

Proceedings of SIMPAR 2010 Workshops
Intl. Conf. on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS
Darmstadt (Germany) November 15-16, 2010
ISBN 978-3-00-032863-3
pp. 229-240

even 95.31%. The average error angle (measured angle between the line from the head to the pointing target and the line-of-sight from the face through the hand towards the pointing target) is 7.28°.

The application can mainly be improved by automating the initialization process of the tracking, such that the user does not have to be interrupted in his actions.

References

1. Nebojsa Jojic, Thomas S. Huang, Barry Brumitt, Brian Meyers, and Steve Harris. Detection and estimation of pointing gestures in dense disparity maps. In *FG*, pages 468–475, 2000.
2. Roger E. Kahn, Michael J. Swain, Peter N. Prokopowicz, and R. James Firby. Gesture recognition using the perseus architecture. In *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 734–741, 1996.
3. C. Colombo, A. Del Bimbo, and A. Valli. Visual capture and understanding of hand pointing actions in a 3-d environment. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 33(4):677–686, Aug. 2003.
4. Christian Eckes, Konstantin Biatov, Frank Hülsken, Joachim Köhler, Pia Breuer, Pedro Branco, and L. Miguel Encarnação. Towards sociable virtual humans: Multimodal recognition of human input and behavior. *IJVR*, 6(4):21–30, 2007.
5. Konstantinos Derpanis. A review of vision-based hand gestures, 2004.
6. B.D. Zarit, B.J. Super, and F.K.H. Quek. Comparison of five color models in skin pixel classification. *Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems, 1999. Proceedings. International Workshop on*, pages 58–63, 1999.
7. Konstantinos G. Derpanis. Vision based gesture recognition within a linguistics framework. Master Thesis, May 2003.
8. M. Bennewitz, T. Axenbeck, S. Behnke, and W. Burgard. Robust recognition of complex gestures for natural human-robot interaction. In *Proc. of the Workshop on Interactive Robot Learning at Robotics: Science and Systems Conference (RSS)*, 2008.
9. E. Seemann, K. Nickel, and R. Stiefelhagen. Head pose estimation using stereo vision for human-robot interaction. pages 626–631, May 2004.
10. J.M.S. Dias, P. Nande, N. Barata, and A. Correia. Ogre - open gestures recognition engine. *Computer Graphics and Image Processing, 2004. Proceedings. 17th Brazilian Symposium on*, pages 33–40, Oct. 2004.
11. Yu Yamamoto, Ikushi Yoda, and Katsuhiko Sakaue. Arm-pointing gesture interface using surrounded stereo cameras system. In *ICPR '04: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 4*, pages 965–970, Washington, DC, USA, 2004. IEEE Computer Society.
12. R. Kehl and L.J. Van Gool. Real-time pointing gesture recognition for an immersive environment. pages 577–582, 2004.
13. Kai Nickel and Rainer Stiefelhagen. Visual recognition of pointing gestures for human-robot interaction. *Image Vision Comput.*, 25(12):1875–1884, 2007.
14. Jochen Triesch and Christoph Von Der Malsburg. A gesture interface for human-robot interaction. In *Proc. of 3th IEEE Intl. Conf. on Automatic Face and Gesture Recognition*, pages 546–551, 1998.
15. Hong Deng, Lian Jin, Li Zhen, and Jian Huang. A new facial expression recognition method based on local gabor filter bank and pca plus lda, 2006.

Proceedings of SIMPAR 2010 Workshops
Intl. Conf. on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS
Darmstadt (Germany) November 15-16, 2010
ISBN 978-3-00-032863-3
pp. 229-240