Deictic Gestures with a Time-of-Flight Camera

Martin Haker, Martin Böhme, Thomas Martinetz, and Erhardt Barth

Institute for Neuro- and Bioinformatics, University of Lübeck,
Ratzeburger Allee 160, 23538 Lübeck, Germany
{haker, boehme, martinetz, barth}@inb.uni-luebeck.de
http://www.inb.uni-luebeck.de

Abstract. We present a robust detector for deictic gestures based on a time-of-flight (TOF) camera, a combined range and intensity image sensor. Pointing direction is used to determine whether the gesture is intended for the system at all and to assign different meanings to the same gesture depending on pointing direction. We use the gestures to control a slideshow presentation: Making a "thumbs-up" gesture while pointing to the left or right of the screen switches to the previous or next slide. Pointing at the screen causes a "virtual laser pointer" to appear. Since the pointing direction is estimated in 3D, the user can move freely within the field of view of the camera after the system was calibrated. The pointing direction is measured with an absolute accuracy of 0.6 degrees and a measurement noise of 0.9 degrees near the center of the screen.

1 Introduction

We use a novel type of sensor, the time-of-flight (TOF) camera, to implement simple and robust gesture recognition. The TOF camera [1] provides a range map that is perfectly registered with an intensity image at 20 frames per second or more, depending on the integration time. The camera works by emitting infrared light and measuring the time taken by the light to travel to a point in the scene and back to the camera; the time taken is proportional to the distance of the point from the camera, allowing a range measurement to be made at each pixel.

In this paper, we use gestures recognized using the TOF camera to control a slideshow presentation, similar to [2] where, however, a data glove was used to recognize the gestures. Another idea we adapt from [2] is to recognize only gestures made towards an "active area"; valid gestures made with the hand pointing elsewhere are ignored. This solves the problem (also known as the "immersion syndrome") that unintentional hand movements or gestures made towards other people may erroneously be interpreted as commands.

We expand this idea by allowing the same gesture to mean different things when made towards different active areas. Specifically, the slideshow is controlled in the following way: To go to the next slide, point to the right of the screen and make a thumbs-up gesture with the hand; to go to the previous slide, point to the left of the screen and make a thumbs-up gesture. Point at the screen and a

S. Kopp and I. Wachsmuth (Eds.): GW 2009, LNAI 5934, pp. 110–121, 2010.

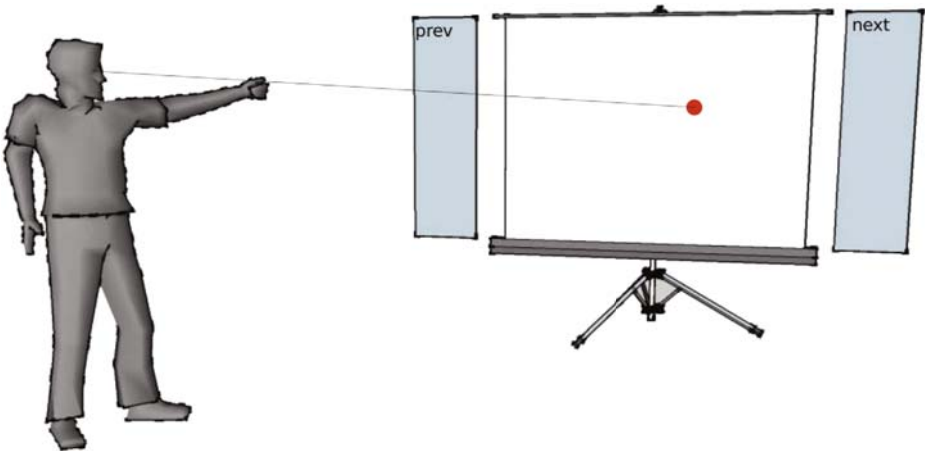

Fig. 1. The application scenario where a user controls a slideshow presentation using deictic gestures. The gestures include switching between the slides and pointing at the screen using a virtual laser pointer.

dot appears at the location you are pointing to, allowing you to highlight certain elements of the slide. This scenario is depicted in Fig. 1.

To determine where the user is pointing on the screen, we need to know its position relative to the camera. This is determined in a calibration procedure where the user points at the four corners of the screen from two different locations; this information is sufficient to compute the position of the screen. After calibration the user is allowed to move freely within the field of view of the camera, as the system estimates both the screen and the pointing direction in 3D with respect to the camera coordinate system.

The "thumbs-up" gesture is recognized using a simple heuristic on the silhouette of the hand. This simple technique is sufficient because hand gestures are only recognized when the user is pointing at one of the two active regions; when pointing elsewhere, the user need not be concerned that hand movements might be misinterpreted as gestures.

One important advantage of the TOF camera in this setting is that it directly measures the three-dimensional position of objects in space, so that the pointing direction can easily and robustly be obtained as a vector in space. This is much more difficult for approaches that attempt to infer pointing direction using a single conventional camera. One solution is to restrict oneself to pointing directions within the camera plane (see e.g. [3,4]), but this places restrictions on the camera position and type of gestures that can be recognized. A physical arm model with kinematic constraints (see e.g. [5]) allows arm pose to be estimated from a single camera image, but the depth estimation can be unreliable for some poses of the arm. In contrast, the approach we will present here is simple but at the same time accurate and robust.

In the remainder of the paper we will first discuss the detection of the pointing and the "thumbs-up" gesture. We will then describe the calibration of the system. Finally, the accuracy of the virtual laser pointer will be evaluated in an experimental setup where users had to point at given targets. This evaluation is conducted in two scenarios: One, where the user does not receive visual feedback, and another, where the estimated pointing position is indicated by the virtual laser pointer.

2 Method

Our method can be divided into three individual components: (i) the detection of the pointing gesture, (ii) the detection of the thumbs-up gesture used for navigating between slides, and (iii) the calibration of the system. This section will cover each component in the order mentioned above. For simplicity, we assume that the user always points towards the left as seen from the camera throughout this section although this is not a restriction of the system.

2.1 Pointing Gesture

The algorithm for detecting pointing gestures can be subdivided into four main stages. The first stage segments the person in front of the camera from the background. The second stage uses the segmented image to identify both the head and the extended hand that is used for pointing. During the third stage, the 3D coordinates of head and hand in space are estimated, which are then used to determine the location on the screen the user is pointing to during the fourth stage. In the following, we will discuss each step of this procedure individually in more detail.

Stage 1: The segmentation of the person in front of the camera uses combined information from both the range and intensity data of the TOF camera. Previous work [6,7] has shown that the combined use of both range and intensity data can significantly improve results in a number of different computer vision tasks. We determine adaptive thresholds for range and intensity based on histograms. In case of the intensity data the threshold discards dark pixels. This has two effects: Firstly, the amount of light that is reflected back into the camera decays proportionally to the squared distance of the object from the camera, thus the background generally appears significantly darker than the foreground. Secondly, this procedure discards unreliable pixels from the range measurement, because the intensity can be considered a confidence measure for the depth estimation as it is related to the signal-to-noise-ratio. In case of the range data, peaks in the histogram can be assumed to correspond to objects at different distances in front of the camera. The threshold is determined as the one that separates the peak of the closest object from the remaining range values. The final segmented image is composed of those pixels that were classified as foreground pixels with respect to both types of data. To ensure that only a single object is considered, only the largest connected component of foreground pixels is retained, all other

Fig. 2. Sample image taken with a MESA SR4000 TOF camera. The leftmost image shows the intensity data. The range image is given in the center, and the resulting segmentation is shown on the right.

objects are considered background. A sample TOF image showing both range and intensity with the resulting segmented foreground is given in Fig. 2.

Stage 2: In the second stage, the segmented image is used to determine an initial guess for the location of the head and hand in the image. We employ a simple heuristic based on the number of foreground pixels in each column of the segmented image. The initial guess for the hand is the topmost pixel in the leftmost pixel column of the silhouette; the head is the topmost pixel in the tallest pixel column. This procedure is extremely simple to implement, yet reliable. We use a single parameter θ to determine whether we have a valid initial estimate, i.e. whether the hand is actually extended and the person is performing a pointing gesture:

$$|i_{\text{head}} - i_{\text{hand}}| \geq \theta \tag{1}$$

Here, i_{head} and i_{hand} denote the indices of the pixel columns corresponding to the initial guess for the head and hand, respectively, where indices of pixel columns increase from left to right.

Stage 3: During the third stage of the method, the initial guesses are refined to more accurate pixel positions in the image. Once these positions are determined, the corresponding range values are estimated, and finally the coordinates of both the head and hand can be computed in 3D by inverting the camera projection using the known intrinsic camera parameters.

In order to refine the pixel positions of the head and hand in the image, we define rectangular regions of interest (ROIs) around the initial guesses and compute the centroids of the foreground pixels in the ROIs to find the centers of the head and hand blobs; these refined positions are marked by crosses in Figure 3.

To invert the camera projection we require the actual distance of the head and hand from the camera. Again, we define ROIs around the estimated pixel coordinates and take the average range value of the foreground pixels within the ROI to obtain estimates for the two range values. Finally, from the pixel coordinates (x, y), the distance from the camera r, and the intrinsic parameters

Fig. 3. Segmented image of the user with the detected locations of the head and hand marked by crosses. The time-of-flight camera measures the three-dimensional positions of these points, which are then used to compute the pointing direction.

of the camera one can infer the 3D coordinates of the corresponding point $\boldsymbol{x}$ in camera coordinates using the following formula:

$$\boldsymbol{x} = r\frac{((c_x - x)\cdot s_x, (c_y - y)\cdot s_y, f)^T}{\|((c_x - x)\cdot s_x, (c_y - y)\cdot s_y, f)^T\|_2} \tag{2}$$

Here, (c_x, c_y) denotes the principal point, i.e. the pixel coordinates of the point where the optical axis intersects the image sensor. The width and height of a pixel are defined by s_x and s_y, and the focal length is given by f. To obtain a more stable estimate, a Kalman filter [8] tracks the 3D coordinates of the head and hand from frame to frame.

Stage 4: Because the TOF camera allows us to determine the position of the head and hand in space, we directly obtain an estimate for the pointing direction from the ray that emanates from the head and passes through the hand. (As Nickel and Stiefelhagen [9] show, the line connecting the head and hand is a good estimate for pointing direction.) This ray can be represented in camera coordinates by the following line equation:

$$\boldsymbol{r} = \boldsymbol{o} + \lambda\boldsymbol{d} \tag{3}$$

Here, $\boldsymbol{o}$ denotes the origin of the ray and corresponds to the 3D position of the head. The direction of the ray is given by $\boldsymbol{d} = \boldsymbol{p} - \boldsymbol{o}$, where $\boldsymbol{p}$ denotes the position of the hand. The parameter $\lambda \geq 0$ defines a point $\boldsymbol{r}$ in front of the person along the pointing direction.

We now intersect this ray with the screen used for projecting the slides. To this end we represent the screen by its center $\boldsymbol{c}$ and the normal $\boldsymbol{n}$ of the screen plane. Assuming that both $\boldsymbol{c}$ and $\boldsymbol{n}$ are also given in camera coordinates, the intersection $\boldsymbol{x}$ of the ray and the screen is given by:

$$\boldsymbol{x} = \boldsymbol{o} + \frac{\langle \boldsymbol{c} - \boldsymbol{o},\ \boldsymbol{n}\rangle}{\langle \boldsymbol{d},\ \boldsymbol{n}\rangle}\boldsymbol{d} \tag{4}$$

The intersection is only valid if the scalar product $\langle \boldsymbol{c}-\boldsymbol{o},\ \boldsymbol{n}\rangle$ is positive, otherwise the user is pointing away from the screen plane.

What remains to determine is if the intersection lies within the limits of the screen. In that case, the intersection can be converted to pixel coordinates on the screen in order to display the virtual laser pointer.

The location and size of the screen are determined by the calibration procedure introduced in Sect. 2.3. Since the procedure determines the 3D position of the four corners of the screen independently, the screen is generally not represented by a perfect rectangle. Thus, we determine the intersection by considering two triangles that are obtained by dividing the screen diagonally along the line from the bottom left corner to the top right corner. Assume that the triangles are defined by their three corners $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{c}$ in counter-clockwise order such that either the top left or the bottom right corner are represented by $\boldsymbol{a}$. For both triangles one can solve the following equation under the constraint that $d_1 = 1$:

$$\boldsymbol{x} = \begin{pmatrix} a_1 & b_1 - a_1 & c_1 - a_1 \\ a_2 & b_2 - a_2 & c_2 - a_2 \\ a_3 & b_3 - a_3 & c_3 - a_3 \end{pmatrix} \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} \tag{5}$$

Intuitively, we check if the intersection $\boldsymbol{x}$, represented as a linear combination of the two sides of the triangle given by $\boldsymbol{b} - \boldsymbol{a}$ and $\boldsymbol{c} - \boldsymbol{a}$, lies within the bounds of the triangle. Thus, if $d_2 + d_3 \leq 1$ holds for the upper triangle, the intersection $\boldsymbol{x}$ lies above the diagonal through the screen. Correspondingly, $\boldsymbol{x}$ lies below the diagonal if $d_2 + d_3 \leq 1$ holds for the lower triangle.

We now convert the coefficients d_2 and d_3 to coordinates x and y on the screen in such a way that the top left corner corresponds to $(x, y) = (0, 0)$ and the bottom right corner corresponds to $(x, y) = (1, 1)$. This is achieved by setting $(x, y) = (d_2, d_3)$ if $\boldsymbol{x}$ was above the diagonal through the screen and setting $(x, y) = (1 - d_2, 1 - d_3)$ otherwise. As a result one obtains for example the four different interpretations of the pointing gesture listed in Tab. 1. These interpretations correspond to the scenario depicted in Fig. 1.

In the "on screen" case, the virtual laser pointer is displayed on the screen at the location (x, y) the user is pointing to. If the user is pointing to one of the two active areas "left of screen" or "right of screen", a small triangle is displayed at the corresponding edge of the screen to indicate that the system is now expecting input in form of the "thumbs-up" gesture to navigate between the slides of the presentation. In all other cases, any detected pointing gesture is ignored, which avoids the so-called immersion syndrome [2].

Table 1. Interpretation of pointing gesture

on screen	$0.0 \leq x \leq 1.0$	$\wedge$	$0.0 \leq y \leq 1.0$
left of screen	$-0.05 \leq x < 0.0$	$\wedge$	$0.0 \leq y \leq 1.0$
right of screen	$1.0 < x \leq 1.05$	$\wedge$	$0.0 \leq y \leq 1.0$
off screen	otherwise		

Despite the fact that the estimation of the head and hand is quite robust and we apply a Kalman filter to the approximated 3D coordinates for temporal smoothing, the estimated intersection of the pointing direction and the screen in the "on screen" case is not entirely free of noise. This is dealt with by applying a smoothing filter with an exponential impulse response. The strength of the smoothing is adaptive and depends on the amount by which the pointing position changed: The greater the change, the less smoothing is applied. In this way, we suppress "jitter" in the virtual laser pointer when the user's hand is stationary but allow the pointer to follow large hand movements without the lag that would be caused by a non-adaptive smoothing filter.

2.2 Thumbs-Up Gesture

The detection of the thumbs-up gesture is only triggered when a pointing gesture made towards one of the two active areas was detected for the current frame according to the procedure described above.

The thumbs-up detector uses the segmented image and the pixel coordinates that were estimated for the hand. The main idea of the algorithm is that the silhouette of an extended thumb that points upwards is significantly narrower along the horizontal axis than a fist.

Thus, we define an ROI around the position of the hand and count the number of foreground pixels in each row. Next, we estimate w_{fist}, which denotes the width of the fist, by taking the maximum number of foreground pixels counted per row. The parameter w_{fist} is then used to determine the presence of both the fist and the thumb. We count the number c_{fist} of rows containing at least $0.8 \cdot w_{\text{fist}}$ foreground pixels and the number c_{thumb} of rows containing at least one and at most $0.3 \cdot w_{\text{fist}}$ foreground pixels. A thumb is detected in the current frame if both c_{fist} and c_{thumb} exceed a threshold of two. Due to the fact that the thresholds for detecting the fist and thumb depend on the estimated width of the fist w_{fist} in the current image, the procedure is relatively independent of the distance of the user from the camera, i.e. the algorithm is scale-invariant.

To avoid misdetections due to noise, we keep track of the detections of the thumb per frame over a certain time window, i.e. the command for switching to the next slide is only issued if the thumbs-up gesture was detected in four out of six consecutive frames. At the same time, we want to avoid multiple activations of the command for switching to the next slide if the above criterion is fulfilled in a number of consecutive frames. Otherwise, the user would not be able to go from one slide to the next in a controlled fashion without unintentionally skipping slides. Thus, we ignore any detections of the gesture for a total of 50 frames once a switch-to-next-slide command was issued. Since our system operates at roughly 25 Hz, a new command can only be issued every two seconds. This gives the user sufficient time to end the thumbs-up gesture once the gesture takes effect in order to prevent the system from switching directly to the next slide.

2.3 System Calibration

The third component of our method deals with the calibration of the system. To determine where the user is pointing on the screen, we need to know its position relative to the camera. This is determined in a calibration procedure where the user points at the four corners of the screen from two different locations; this information is sufficient to compute the position of the screen, as we will demonstrate in more detail in the following.

During calibration the user is asked to point continuously at one of the four corners of the screen for a total of 50 frames. This allows us to obtain a robust estimate for the position of the head and hand for the given pointing direction. Again, the pointing direction can be represented by a ray $\boldsymbol{r} = \boldsymbol{o} + \lambda \boldsymbol{d}$ that emanates from the head and passes through the hand. We can assume that this ray passes through the corner the user was pointing at. However, we do not know the exact location of the corner along the ray.

To obtain this information, the user is asked to move to a different position in the field of view of the camera and to point again at the same corner for a total of 50 frames. By this procedure we estimate a second ray that should also pass through the corner of the screen. Ideally, the two rays intersect at the position of the corner; however, this assumption does generally not hold due to measurement noise. Nevertheless, a good estimate for the position of the corner can be obtained from the point that is closest to both rays in 3D space.

Assuming that the two estimated pointing directions are represented by rays $\boldsymbol{r}_i = \boldsymbol{o}_i + \lambda_i \boldsymbol{d}_i$ where $i \in \{1, 2\}$, one can obtain this point by minimizing the squared distance $(\boldsymbol{o}_1 + \lambda_1 \boldsymbol{d}_1 - \boldsymbol{o}_2 - \lambda_2 \boldsymbol{d}_2)^2$ between the two rays with respect to λ_1 and λ_2. This leads to the following linear system of equations where we assume $\|d_i\| = 1$ without loss of generality:

$$\begin{pmatrix} 1 & \langle \boldsymbol{d}_1, \boldsymbol{d}_2 \rangle \\ -\langle \boldsymbol{d}_1, \boldsymbol{d}_2 \rangle & -1 \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix} = \begin{pmatrix} -\langle \boldsymbol{o}_1 - \boldsymbol{o}_2, \boldsymbol{d}_1 \rangle \\ -\langle \boldsymbol{o}_1 - \boldsymbol{o}_2, \boldsymbol{d}_2 \rangle \end{pmatrix} \tag{6}$$

Solving Eq. (6) yields the parameters λ_1 and λ_2, which specify the closest point on one ray with respect to the other, respectively. Taking the arithmetic mean of both solutions as specified by Eq. (7) yields the approximation of the intersection of both rays and, thus, an estimate for the position of the corner in camera coordinates:

$$\boldsymbol{x} = 0.5 \cdot (\boldsymbol{o}_1 + \lambda_1 \boldsymbol{d}_1 + \boldsymbol{o}_2 + \lambda_2 \boldsymbol{d}_2) \tag{7}$$

This procedure can be repeated for the remaining three corners of the screen. The approach does not guarantee, however, that all four corners lie in one plane. Thus, we fit a plane through the four corners by least squares and project the corners onto this plane to obtain their final estimates. The normal to this plane and the four corners are used to determine where the user is pointing on the screen, as described in Sect. 2.1. Obviously, this calibration procedure does not generally yield a screen that resembles a perfect rectangle in 3D space. How this problem can be treated by dividing the screen into two triangles along its diagonal was also discussed in Sect. 2.1.

We consider the procedure of not enforcing the screen to be rectangular an advantage, because it provides an implicit way of correcting systematic errors. Such errors may for example be caused by measurement errors or a simplified approximation of the camera parameters. The former can e.g. be due to multiple reflections of the scene [10]. The latter can occur if the optical system is not modelled accurately in the process of inverting the camera projection, e.g. if radial distortions of the lens or other effects are not taken into account.

3 Results

The method was implemented in C++ under the Windows operating system. On a 2 GHz Intel Core 2 Duo, it requires 40 ms per frame, achieving a frame rate of 25 frames per second.

To assess the accuracy with which the pointing direction is measured, we performed a test with 10 users. Each user was first given a few minutes to practice using the system. We then presented a sequence of nine targets at predefined positions on the screen; users were instructed to point at a target as soon as it appeared. Once a pointing gesture towards the screen was detected, each target was presented for a total of 50 frames, which corresponds to a time interval of roughly two seconds, before it disappeared. Users were asked to return to a normal standing position after the target had disappeared. Before presenting the next target, the system waited for four seconds to allow the user to rest the arm. The order in which the targets appeared was chosen in such a way that the average distance between successive targets on the screen was maximized.

For each user, we performed this test under two different conditions: Under the first condition, the virtual laser pointer was switched off, i.e. the users did not receive any feedback about the measured pointing direction. This gives an impression of the overall accuracy of the system. For the second test condition, the virtual laser pointer was switched on, allowing users to compensate for systematic calibration and measurement errors. This test condition therefore gives an impression of the residual measurement noise after the temporal smoothing described in Sect. 2.1.

Fig. 4a shows the results of the first test condition (without visual feedback) to assess the overall accuracy of the system. Here, measured error in pointing direction can have two sources: (i) Systematic errors due to measurement noise and inaccurate calibration and (ii) errors induced by the assumption that the ray emanating from the eyes across the hand corresponds to the natural human pointing direction [11]. The horizontal axis plots the frame number after the pointing gesture was detected, and the vertical axis plots the distance between the target and the measured pointing position in pixels. In the test setup the screen had a size of 1.71 m $\times$ 1.29 m and the user was standing at a distance of roughly 3.2 m from the screen. As a result, an offset of 20 pixels corresponds to approximately one degree. The solid line gives the mean distance, averaged over all users and pointing targets, and the shaded area indicates an interval of two standard deviations above and below the mean, i.e. 95% of the errors fell within this range.

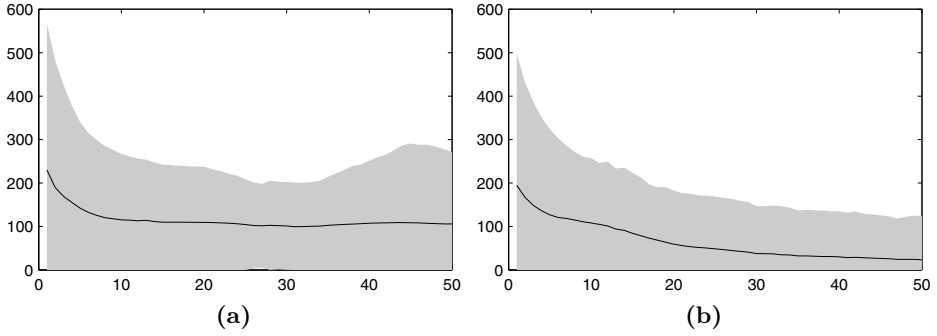

Fig. 4. Measurement error in pointing direction (a) without visual feedback (the virtual laser pointer was switched off) and (b) with visual feedback (virtual laser pointer switched on). The horizontal axis plots the time in seconds after the target appeared, and the vertical axis plots the distance between the target and the measured pointing position.

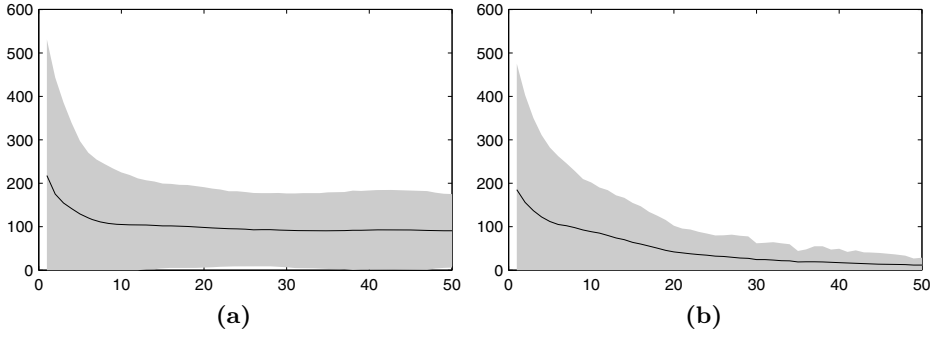

Fig. 5. Measurement error in pointing direction (a) without visual feedback (the virtual laser pointer was switched off) and (b) with visual feedback (virtual laser pointer switched on). Only targets near the center of the screen where considered.

From the plot, we can see that users took around 10 frames or 400 ms to point at a target; after this time, the average error stabilizes at around 106 pixels (or 5.3 degrees), with 95% of error falling between 0 and 271 pixels.

Fig. 4b shows the results of the second test condition (with visual feedback). As expected, the error stabilizes at a lower value of 23 pixels (or 1.2 degrees) on average but also takes a longer time to do so – around 1600 ms. This is because, after pointing at the target as in the first test, users need to correct their hand position to compensate for the systematic measurement error and bring the virtual laser pointer onto the target.

A closer look at the data reveals that the largest measurement errors occur for targets that are close to the corners of the screen. This is mainly due to shortcomings of the rather simple calibration procedure. However, the system

is well calibrated for the center of the screen where most of the content of a presentation is likely to be placed. Thus, the impact of calibration errors near the corners on the usability is rather low. This becomes clear by looking at Fig. 5a and Fig. 5b. Again, the plots show the distance between the target and the measured pointing position without and with visual feedback, respectively. This time however, only targets near the center were considered. In case of the first test condition without visual feedback the average error amounts to 91 pixels (4.6 degrees). For the second test condition with visual feedback the average error was halfed to 12 pixels, which corresponds to 0.6 degrees. Note also that the standard deviation decreased significantly to 17 pixels (0.9 degrees) in the test with visual feedback, which indicates that the system is very robust and hence intuitive to use near the center of the screen.

4 Discussion

We have presented a framework that implements simple and robust gesture recognition in the context of a slideshow presentation. The system is based on a TOF camera that allows us to detect and interpret pointing gestures in an intuitive and effective way, because the provided range data facilitates the localization of the user in front of the camera and allows the estimation of the pointing direction in 3D space once the head and hand have been identified.

We believe pointing is a powerful way to determine whether a gesture is intended for the system at all and to assign different meanings to the same gesture depending on where the user is pointing. A simple gesture with a simple recognition procedure is sufficient for our application because the meaning of the gesture is strongly tied to the direction it is made in.

Thus, we have developed an intuitive system that allows the user to control a slideshow by switching between slides through a simple thumbs-up gesture that is made towards one of the sides of the screen. Alternatively, we could have implemented an additional thumbs-down gesture using a single active area, but our intention here was to demonstrate the use of multiple active areas, i.e. the number of active areas multiply the number of actions that can be triggered with a given set of gestures. Finally, the user may highlight certain details on the slides simply by pointing at them; a virtual laser pointer is displayed at the location the user is pointing to. This virtual laser pointer has two advantages: First, the size and appearance can be chosen depending on the context. Second, the build-in smoothing of the pointer can veil a tremor of the users hand that may originate from an impairment or excitement.

Acknowledgment

This work was developed within the ARTTS project (www.artts.eu), which is funded by the European Commission (contract no. IST-34107) within the Information Society Technologies (IST) priority of the 6th Framework Programme. This publication reflects the views only of the authors, and the Commission

cannot be held responsible for any use which may be made of the information contained therein.

References

1. Oggier, T., Büttgen, B., Lustenberger, F., Becker, G., Rüegg, B., Hodac, A.: SwissRanger^{TM} SR3000 and first experiences based on miniaturized 3D-TOF cameras. In: Ingensand, K. (ed.) Proc. 1st Range Imaging Research Day, Zurich, pp. 97–108 (2005)
2. Baudel, T., Beaudouin-Lafon, M.: CHARADE: Remote control of objects using free-hand gestures. Communications of the ACM 36, 28–35 (1993)
3. Hofemann, N., Fritsch, J., Sagerer, G.: Recognition of deictic gestures with context. In: Rasmussen, C.E., Bülthoff, H.H., Schölkopf, B., Giese, M.A. (eds.) DAGM 2004. LNCS, vol. 3175, pp. 334–341. Springer, Heidelberg (2004)
4. Moeslund, T.B., Nøregaard, L.: Recognition of deictic gestures for wearable computing. In: Gibet, S., Courty, N., Kamp, J.-F. (eds.) GW 2005. LNCS (LNAI), vol. 3881, pp. 112–123. Springer, Heidelberg (2006)
5. Moeslund, T.B., Granum, E.: Modelling and estimating the pose of a human arm. Machine Vision and Applications 14, 237–247 (2003)
6. Haker, M., Böhme, M., Martinetz, T., Barth, E.: Geometric invariants for facial feature tracking with 3D TOF cameras. In: Proceedings of the IEEE International Symposium on Signals, Circuits & Systems (ISSCS), Iasi, Romania, vol. 1, pp. 109–112 (2007)
7. Haker, M., Martinetz, T., Barth, E.: Multimodal sparse features for object detection. In: Alippi, C., Polycarpou, M., Panayiotou, C., Ellinas, G. (eds.) ICANN 2009. LNCS, vol. 5769, pp. 923–932. Springer, Heidelberg (2009)
8. Kalman, R.E.: A new approach to linear filtering and prediction problems. Transactions of the ASME, Series D, Journal of Basic Engineering 82, 35–45 (1960)
9. Nickel, K., Stiefelhagen, R.: Pointing gesture recognition based on 3D-tracking of face, hands and head orientation. In: International Conference on Multimodal Interfaces, pp. 140–146 (2003)
10. Gudmundsson, S.A., Aanæs, H., Larsen, R.: Effects on measurement uncertainties of time-of-flight cameras. In: Proceedings of the IEEE International Symposium on Signals, Circuits & Systems (ISSCS), vol. 1, pp. 1–4 (2007)
11. Kranstedt, A., Lücking, A., Pfeiffer, T., Rieser, H., Wachsmuth, I.: Deixis: How to Determine Demonstrated Objects Using a Pointing Cone. In: Gibet, S., Courty, N., Kamp, J.-F. (eds.) GW 2005. LNCS (LNAI), vol. 3881, pp. 300–311. Springer, Heidelberg (2006)