A Kinect-based system to enable interaction by pointing in smart spaces

Ana Fernández, Luca Bergesio, Ana M. Bernardos, Juan A. Besada, José R. Casar

Abstract—**Pointing is a universal gesture that naturally expresses interest or attraction towards the pointed items. If some 'magic' is added, the gesture may also make these items perform actions. In this paper, we describe a system that enables to interact by pointing with digital or physical controllable resources distributed in a smart space. The system facilitates building an interactive room using COTS devices, in particular a pair of Kinect sensors. The pointing direction is inferred from the user's elbow-wrist vector, which together with a secondary elbow-object vector serves to filter the controllable objects in the area of pointing. Experiments with 8 users in a real setting demonstrate the feasibility of the concept and show that the accuracy of the system is very dependent on the relative position user-resource and on the user behaviour itself.**

Keywords—computer vision; natural interaction; pointing interaction; Kinect; smart spaces.

I. INTRODUCTION

The pointing gesture is universally used as a way to denote or attract interest towards an item [1]. When combined with the adequate technology, a pointing gesture may also enable to remotely control the pointed object. This is the idea under the interaction concept that is explored in this work: the use of pointing to govern the objects in a smart environment. Thus the paper describes a system for pointing interaction, or deictic interaction, which delivers a 'natural user interface' (NUI) to the environment. The proposed NUI is built on the use of a traditional human-to-human interaction model (pointing, in this case), it is not instrumented through artificial control devices and the interaction mechanism aims at becoming 'invisible' after a learning process.

A key point of the system is that it relies on commercial-of-the-shelf (COTS) sensors, in particular Kinect devices. The Kinect device includes an RGB camera, a low-cost depth sensor and a multiarray microphone to facilitate full-body 3D motion capture, facial and voice recognition. The device makes also possible to recognize the user's body or hand poses or in-air gestures. On the image streaming from two synchronized sensors, it is possible to transform a room in an 'interactive-through-pointing' space. The system is composed by several processing modules, which facilitate person detection and tracking, gesture identification and gesture-object correlation for effective control.

The context and the system is described as follows. Section II summarizes the state-of-the-art on pointing interaction-enabling systems. Section III describes the deployment scenario and the pre-operation processes for the pointing system. The system modules are detailed in Section IV, while Section V gathers result on a real deployment that has been built for user validation. Section VI concludes the work.

II. RELATED WORK

Controlling objects by pointing at them is not a new concept; actually, it has been explored since the 80's. For example, the 'Put-that-there' system (1980) [2] relies on a specific wearable device to calculate the orientation of a user seated in an armchair in a Media Room; the room is also equipped with a voice recognition system, thus specific sentences combined with natural pointing enables the user to command simple shapes in a graphics display surface.

In [3], authors apply Hidden Markov Models trained with 3D trajectories of the head and the hand when performing sample pointing gestures (described in three phases: begin, hold and end). The system works on a fixed-baseline stereo camera and combines stereoscopic range information and skin-color classification to achieve a gesture detection rate of 88% and a pointing detection rate of 90% on the correctly identified gestures (8 pointing targets, ten persons, 206 pointing gestures). Two approaches are used to estimate the pointing direction: the line of sight between the head and the hand and the forearm orientation. The 3D hand pointing gesture is estimated in [4] from two cameras, orthogonally located on top of the user and at his left side. The algorithm needs to be trained. First, the hand region is detected, then the hand is tracked in the limited search regions, using active appearance models to detect and track 14 feature points along the hand contour from a top and side view; finally, the 3D pointing direction is estimated by combining the different views. The correct pointing rate is 91% in 7600 frames. In [5], a system to track gestures in front of a computer screen was implemented. The system works on two cameras placed several feet above the desktop and several feet apart, such that they can both see the user's hand. The processing first extracts 2D information from the two images and then combines the information from each image to obtain 3D information about the thumb and pointing finger. The pose for each finger consists on three positional coordinates and two angles.

Authors of [6] propose to detect point gestures by using binocular stereovision and estimating the pointing direction on the line of sight connecting the point of pseudo-eyes below a certain distance of the head top vertically and the fingertip of the pointing arm. For this system to work, it is needed to place two mounted overhead cameras looking down at an oblique angle to capture head and pointing arm. Authors underline that their system do not need to capture entire bodies and faces and does not constrain the flat surface pointed by the user to be visible by the cameras. Validation is done on a total of 864 gestures performed by 36 users, 2 hands, pointing at 12 panels located in a range of [3.5, 12] m. away from the cameras in normal lighting. The recognition rate is 93% in the most far away panel and increases up to 97% in the closest one.

Authors of [7] present an algorithm for real-time detection of pointing gestures in immersive environments (cave-like ones); they define the pointing direction as the line of sight connecting the eyes and the pointing fingertips. They rely on the live video stream from 4-8 statically mounted and calibrated cameras, including 1 overhead camera. The algorithm starts with background subtraction and silhouette segmentation. Extremal points on the silhouettes corresponding to the head and hands are searched and matched among the views from different cameras. The head and one or two hands positions are then searched within the 3D points. In [8], the use of disparity maps (instead of color-based blob trackers) as a more robust technique against light changes is proposed. The system subtracts the background, analyzes the foreground pixels to break the body into parts and estimates the direction of pointing. Authors evaluate their system by enabling the users to select objects in a room or guide a cursor. A similar strategy is used in [9] to extend the recognizable gesture set. A method combining Time-of-Flight (ToF) and RGB cameras is proposed in [11]. After a calibration process, it is applied a 3D hand detection algorithm based on depth and color, on which the gesture is recognized by using a classifier that relies on a dimensionality reduction based on Average Neighborhood Margin Maximization, approximated using Haarlets. The Kinect sensor is mounted in a robot in [10] to simulate an interaction in which the robot asks for directions to people and automatically detects a 3D pointing direction by connecting the wrist with the center of the hand. The system works on a Haarlet-based hand gesture recognition strategy.

Some proposals explore multimodal systems for 3D interaction. In [12], eye gaze is used to move a rectangular overlay called 'area of interaction' in a computer screen; this area defines the boundary for pointer positioning. Hand fingertip is used to position the pointer and the presence of a second hand triggers a click event. Using the system described in [4], authors of [13] aims at providing a multi-gesture interaction system combining eye gaze, head pose/position, hand-pointing direction and mouth opening/closing. Authors claim that the maximum pointing error for a specific application is around 9 pixels.

Apart from vision-based systems, there are other proposals that instrument the 3D pointing with devices. The 'Point&Click' system [14] enables a stand-alone 'remote control' to get the control information from other devices, in order to allow operational interaction through a simple user interface. In a similar way, a laser-equipped device is proposed in [15] to retrieve a set of control commands from an object: when the object detects the laser beam, it sends the control description to the master device by using infrared. The XWand [16] is a wand-like device that enables the user to point at an object in the environment and control it using gestures and voice commands. A specific hardware solution to detect the pointing direction of a laser pointer on a screen is provided in [17]. Gloves and wearable devices have been also proposed to interpret hand gestures. The Charade interaction system [18] relies on a Data Glove to enable the user to perform 16 different gestures when working with a presentation. Pointing is one of these gestures. A recent proposal is Digits [19], a wrist-worn sensor containing an IMU and infrared camera that optically images a large part of the user's bare hand.

3D pointing estimation has many applications; among them, robot control is a widely explored one in literature. In [20], a human-robot interaction method enables the human to intuitively select a 3D location and communicate it to a robot. The human points at the position with an off-the-shelf green laser pointer, then the robot detects the resulting laser spot with an omnidirectional, catadioptric camera with a narrow-band green filter. Once detected, the robot moves its stereo pan/tilt camera to look at the detected laser spot and then estimate the 3D location of the spot relative to the robot body. Authors claim an average error of 9.75cm for 178 trials (12 objects, 5 users, 3 pointing attempts per user). The work [3] is applied to robot control in [21]. As previously mentioned [10] also focus in exploring human-robot interaction. Object or environment control is also the service scenario for many systems. For example, recent CityHome MIT project address the issue of configuring a gesture-responsive home.

Current developments on Kinect for pointing interaction are still limited. The use of this low-cost device may facilitate to build and deploy real services. For this reason, our work proposes to use Kinect capabilities to build a pointing-aware environment.

III. Service scenario setting

The service objective of this Kinect-based system is to facilitate the creation of spaces enabled with interactive pointing capabilities. These capabilities will serve to activate and manage different resources, such as smart home devices (lights, blinds, etc.) by pointing at them.

Kinect 1.0 devices have optimal visibility in a range of 0.8-4 meters from the sensor. To enable pointing, the system has to determine the position of the users' arms, which derives into a practical operation range of 1.2 to 3.5 meters. Our experimental setting is a room of 19 m^2 (4.9x3.9 m.), which has been equipped with 2 Kinect sensors obliquely situated at 2.4 m. high in opposite corners of the same wall (Figure 1); in these conditions, the user can move within an operating area of 2.5x3 m. approximately. The placement of the devices maximizes the operating area when compared to the two-walls parallel and one-wall perpendicular alternatives.

In order to obtain the needed features from the Kinect cameras, a calibration process has been previously completed to obtain the camera intrinsic and extrinsic parameters. The accurate calibration of the camera enables to estimate the required distances in the real world from the captured images. Intrinsic parameters (e.g. principal axis, optical center and focal distance) are related to the internal geometry and the optical features of the camera and remain constant if the features and relative positions of the optics and the imaging sensor do not vary. External parameters estimate the position and the orientation of the camera within the scene coordinate system (translation vector and rotation matrix). Kinect cameras are CCD enabled (Charged Coupled Devices), thus the intrinsic parameters define the coordinates in the reference frame of the camera. The system requires referencing the sensing devices and the smart objects to a global reference system. Each object in the environment will be assigned with an invariant position, independently of the Kinect device to be used in that moment. The mentioned global reference system will be obtained through the extrinsic parameters of the calibration process.

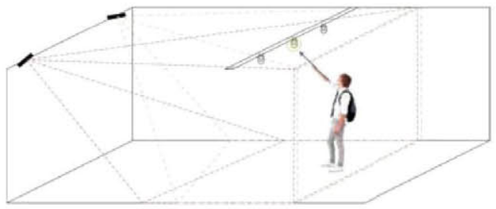


Fig. 1. Service concept and sensor deployment.

In our case, the system has been calibrated by using the open tool Camera Calibration Toolbox for MATLAB (CCT). Its workings are described e.g. at [22][23]. The calibration pattern that has been used is a chessboard of 4x4 80 mm-sided squares. The two Kinects have taken 34 images with the calibration pattern in different positions of the operational space in order to extract the intrinsic parameters. Afterwards, the coordinate origin for the new reference system has been placed on the floor, in the most distant line of sight of the Kinect devices (Figure 2a). With this information, extrinsic parameters have been calculated.

IV. System Modules

On the calibrated cameras, we have deployed a tracker that enables to continuously position the user. On the user standing still in a position, the system is prepared to detect pointing scenes and trigger the subsequent actions.

A. Tracking and user management module

When the interaction application is launched, the system waits until a user shows up in area of coverage of at least one Kinect device. On this 'new user' presence event, the logic starts acquiring the position of the user's head (Kinect 1.0 device is ready to detect 20 points/joints in the user's body). If the user's position falls into the operating space, a coordinates transformation process is initiated (as said in Section III.b, the coordinates of the initial positions are referred to the Kinect reference system and need to be transformed into the global reference system) and the data of the recognized user are stored in a list that enables user management.

The maximum number of users that are detected by the Kinect 1.0 is 6. In this first prototype of the pointing system, a single user is enabled to interact with the objects in the space in a given slot. Thus, when a new user is detected by one of the Kinect sensors, it is needed to figure out if s/he is an existent user previously registered by the other Kinect device or a new user. In the first case, the system performs a user comparison, basically estimating the difference between the user's position that each devices provide. If the distance is smaller than a 3D threshold (600, 600, 800mm.), then the system enables joint detection for the two devices. Apart from the 'new user' events, the system needs to manage 'user lost' events, in order to update the user list, which is decremented only when the two devices lose the user from their vision. For each frame (Kinect 1.0 works at 30fps), a positioning update signal is emitted and the transformation of the position to the global reference system, completed. A user track is depicted in Figure 2a.

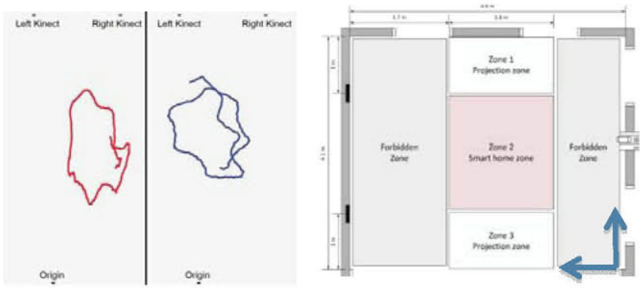


Fig. 2. a) Track acquired by both Kinects. b) activation zones by object type.

B. Pointing interaction and object selection

Our first deployment of the pointing system makes possible to interact with two different types of devices: 9 smart led lights (in 3 rows) and 1 blind, and 4 on-wall projected contents (Figure 4a). The actions to be performed on the objects are two-state type (on/off, up/down, show/no show) and depend on their previous state (e.g. switch the pointed light on if it is off and viceversa), thus not additional gesture grammar is needed for this first version of the system.

Depending on the region of the operating space, the user will be capable of interacting with the different resources. Due to service reasons, the central area of the operating space (zone 2) is configured to interact with the led lights, while the lateral areas (zone 1, 3) are suited to interact with the projected contents (Figure 2b). As it is afterwards explained, the pointing vector is extracted from the user's arm position. To simplify the initial setting, all the resources can be activated by the user's right arm when in the zones 2 and 3, while the left arm is required to activate actions in zone 1.

The global reference system is set at the left down corner of the room (Fig. 2b). Whenever a user is detected within an activation zone, Kinect estimates the position of the elbow and wrist joints for the activation arm. Those coordinates are transformed into the global reference system: the user's elbow position is then (x_e, y_e, x_e), the wrist position (x_w, y_w, x_w) and the object position (x_o, y_o, x_o). Then, the normalized pointing vector elbow-wrist $\overline{v_{ew}}$ is calculated as:

$$\overline{v_{ew}} = \frac{(x_w - x_e, y_w - y_e, z_w - z_e)}{\sqrt{(x_w - x_e)^2 + (y_w - y_e)^2 + (y_w - y_e)^2}} \qquad (1)$$

This done, object filtering is performed. A new reference system is set at the user's elbow and the wrist and objects' positions, recalculated with respect to it (Fig. 3b). This way, the wrist remains situated in one of the eight cube regions in which the space is naturally divided. All the objects that are not within the target region are discarded as candidate objects. When the wrist position is in the boundaries of two cubes, objects within both spaces are considered.

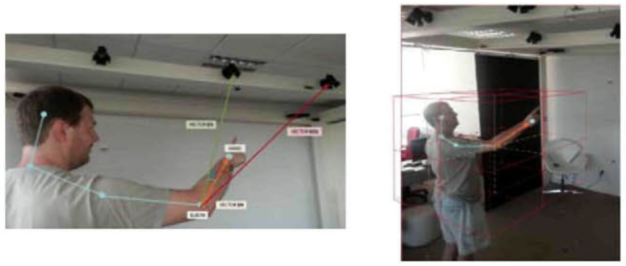


Fig. 3. a) Object selection vectors; b) Cubic filtering.

For each object that remains in the candidate list, the inner product of $\overline{v_{ew}}$ and $\overline{v_{eo}}$ is calculated (the inner product would be 1 if the two vectors are perfectly aligned).

$$\overline{v_{eo}} = \frac{(x_o-x_e, y_o-y_e, z_o-z_e)}{\sqrt{(x_o-x_e)^2+(y_o-y_e)^2+(z_o-z_e)^2}} \qquad (2)$$

$$\overline{v_{ew}} \cdot \overline{v_{eo}} > T_a \qquad (3)$$

Then, it is checked if the inner product is over an 'acceptance' threshold T_a. If so, the object is kept within the candidate list. T_a has been empirically determined and depends on the user's pointing habits and the position of the target resources. In our trial specific setting, it varies between 0.98 and 0.96 (equivalent to an angle between vectors in 11º-16º range, Section V.B). After this second filtering, the final stage determines that the target object is the one offering a minimum angular difference between $\overline{v_{ew}}$ and $\overline{v_{eo}}$ (the greater inner product).

Once the object/resource is chosen, the system communicates to the infrastructure to perform the required actions. To manage the led lights and the blind, a socket is used to send the event to an Arduino controller. In order to manage the resource projection, a DLNA (Digital Living Network Alliance) content management system is used.

V. System Evaluation

Two tests have been performed to adjust the experimental parameters and evaluate the system performance. In the first test, an experienced user has cooperated i) to model the system sensitivity to the interaction position, ii) to tune the acceptance threshold for resource filtering and iii) to estimate the response time. In the second test, 8 non-experienced users have tried the system, performing a set of interaction tasks. The number of participants provides a reasonable cost/benefit ratio, being enough to collect insights on the system's main performance and usability aspects [24]. The independent variables of the tests are the reference positions and the target objects/resources to point, and the dependent ones, the time response and the accuracy. Details and results are following gathered.

A. System sensitivity depending on the pointing location.

This experiment was designed to test the influence of the distance between the user and the Kinect devices in the resource selection process. Firstly, five different positions (Figure 4a) in the central area of the interactive room were used as reference testing points. From each position, 3 different led lights were controlled (one light per row). For each pair (reference position, object), 10 measures were collected: in total 50 iterations/light were analyzed (Figure 4b). Results show that the accuracy in the object selection strategy not only depends on the user position, but on the relative position of the resource with respect to the user. Pointing resources behind the user position may make the arm colocation unnatural, deriving this fact in an increased error. Moreover, if the user is located far away from the Kinect devices (e.g. position 4 or 5), detection is affected. The optimal positions are those intermediate (position 2 or 3), as the user is in the optimal detection area for the Kinect devices and the arm position is natural. In the case of the 4 projected resources, three different positions in each of the lateral areas were used to perform an equivalent evaluation (a total of 60 iterations per projected object were obtained). In this case, the reference points were optimally situated, thus providing a success rate close to 100%.

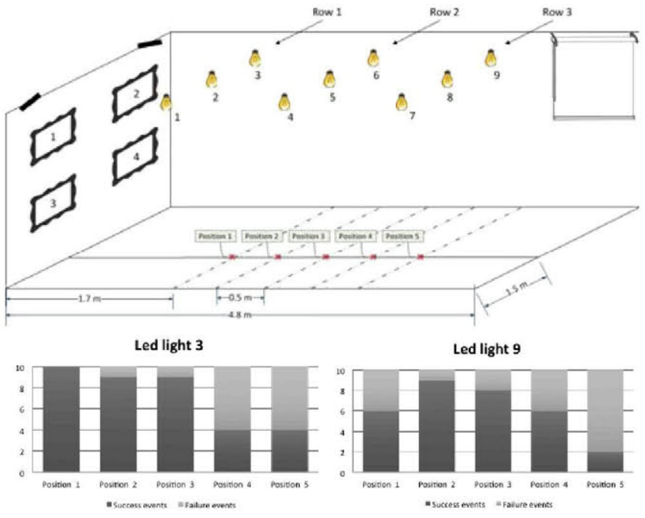


Fig. 4. a) Experiment setting with led lights, blind and projected paintings, and b) Distance effect, example for 2 led lights from different positions.

B. Estimation of the acceptance threshold.

As previously said, in the target selection process, two filters are applied. The second filter is tuned taking into consideration the acceptance threshold that defines the maximum allowed angle between the pointing vector $\overline{v_{ew}}$ and the elbow-resource vector $\overline{v_{eo}}$. The threshold has been adjusted from the data collected in an experiment in which the user freely walked around the activation zones, randomly pointing at different objects until 65 measures/resource were gathered. For each object, the success/failure rate was calculated for different thresholds (Figure 5a), finally configuring the system with data in Table I. Errors can be classified as Type I, when an unwanted object is selected, and Type II, when no object is selected. As it can be seen at Figure 5b, the system acts in a conservative way: most of the errors are Type II.

TABLE I. SUCCESS/FAILURE RATE FOR THE SELECTED THRESHOLDS.

Object	Rate for selected threshold			
	T_a	α	*Success rate*	*Error rate*
Light 1	0.97	14°	76%	24%
Light 2	0.97	14°	91.4%	8.6%
Light 3	0.97	14°	100%	0%
Light 4	0.965	15°	100%	0%
Light 5	0.97	14°	91.4%	8.6%
Light 6	0.965	15°	90%	10%
Light 7	0.97	14°	100%	0%
Light 8	0.97	14°	80%	20%
Light 9	0.965	15°	90%	10%
Proj. image 1	0.97	14°	88.6%	11.4%
Proj. image 2	0.965	15°	91.4%	8.6%
Proj. image 3	0.98	11.5°	87.1%	12.9%
Proj. image 4	0.97	14°	84.3%	15.7%

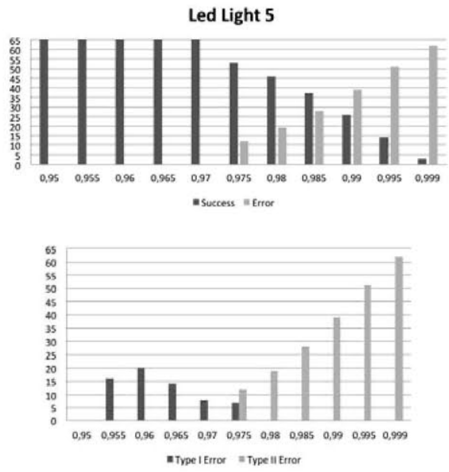
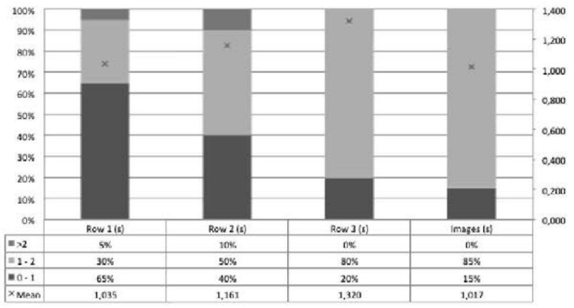


Fig. 5. a) Success/failure rate distribution for different thresholds for led light no. 5. These results mask Type I errors when another light in the same raw (light 4 or 6) is selected. b) Distribution of Type I and Type II errors for led light no. 5 from position 3.

C. *Analysis of the system response time.*

The time between the pointing action and the object response has been measured from a random sequence of actions within the activation space. The tester walked around until she completed 60 interactions with the led lights (20 per row) and 20 interactions with the four projected images. Timestamps in the code show that the pointing calculation process (since Kinect data are received until the pointed object is decided) is basically instantaneous (1 ms).

Operating results that include the complete process – since the user points at the object until the expected object response takes place - are summarized in Figure 6. The interaction duration was estimated off-line from the captured video frames. For led lights, the mean activation time is 1171 ms (σ=496 ms) from the pointing action to the light response. This is due basically to the infrastructure delay: communication with the Arduino-based infrastructure that manage the smart home actuators takes around 300 ms and the row of led lights is blocked for 1 second, to avoid consecutive detection of the same interaction gesture (and subsequently, an unwanted second action over the same device). Thus, if the user wants to switch the light on and off, it will take 0.3s+1s+0.3s=1.6s.



	Row 1 (s)	Row 2 (s)	Row 3 (s)	Images (s)
▨ >2	5%	10%	0%	0%
▨ 1 - 2	30%	50%	80%	85%
■ 0 - 1	65%	40%	20%	15%
× Mean	1,035	1,161	1,320	1,017

Fig. 6. Response time.

With respect to the projected images, the mean projection time is 1017 ms (σ=51ms); the lag is due to the DLNA standard infrastructure that is used to project the image on pointing detection.

D. *Non-experienced user tests.*

With all the information collected above, the system parameters were tuned to provide the optimal user experience and 8 users (ages between 23 and 31) were invited to try the system within June 2014. None of these users had previous knowledge about the system, thus the objective was to compare the time response and the accuracy in untrained users. A facilitator guided the users throughout the experiment.

Users were firstly asked to point at the four projected resources from one position in each of the two activations zones (8 interaction attempts). Success and failures were annotated, together with the error types when needed and the response time. A total of 64 attempts were gathered to obtain 53 success iterations (83%). The mean number of success iterations per participant was 6.6/8 (σ=1.3). No Type I errors were obtained; the average number of Type II errors per user was 1.4 (σ=1.3). Then, users were asked to point at each led light in each row (9 led lights) from positions 2 and 4, in a random order defined by the facilitator. From position 2, 72 iterations were needed to reach 36 success events (50%), while from position 4, with the same number of iterations, 55 success events were obtained (76%). On position 2, the mean number of success iterations was 4.5/8 (σ=1.6), while on position 4, it increased up to 6.9 (σ=0.83). Type I errors represented 33% on position 2 and 29% on position 4. When considering the average number of errors per user, Type II errors double Type I in both positions (position 2: 3 vs. 1.5; position 4: 1.5 vs. 0.6).

Results show, once more, that the relative position from the interaction location to the devices is a relevant factor for success. Additionally, the pointing gesture differs between users, thus accuracy is reduced by this fact. Data show that users perform differently: success rate among users vary from 54% to 85%.

After this first round of the experiments, participants were trained by the facilitator about the best manner to interact with the system regarding the arm position, and the tests were repeated with the same dynamics. The objective of this second round was to evaluate the learning effect but, although minimal enhancement in the success/error rate was achieved, it was not possible to extract conclusive results.

VI. Conclusions and further work

The described work presents a preliminary deployment of a low-cost system that enables to build a pointing-aware interaction space with easy-to-deploy Kinect 1.0 devices. The system has been prototyped and validated to an extent that shows the feasibility of the concept, although there are many open issues to bring the deployment to a fully responsive, universal and multiuser solution.

The system is currently being ported to integrate the Kinect 2.0 device (launched in July 2014). This new device is more accurate and sensitive and provides a wider vision field. It can work consistently in different illumination conditions. The SDK includes the possibility of accurately tracking the user's fingers, which can be useful to both enhance the inference of the pointing vector or to recognize grammars of gestures to perform actions on the objects, this way enhancing the system expressiveness. A problem to solve with Kinect 2.0 is the devices synchronization, as no more than one device is simultaneously supported nowadays.

We are also working to provide more freedom and adapt the system to different pointing habits: the pointing vector can be obtained by fusing different type of inputs that can be retrieved from Kinect 2 (fingers position, gaze, etc.) or other complementary systems that may help when the user moves to border coverage areas. In future settings, the user should be able to freely modify the body pose, the arm to use and the attitude (sitting, standing, walking, etc.). Another interesting issue is how to handle multiuser responses, when different users are interacting in the same space. This situation is perfectly feasible for potential gaming or educative services.

A deep study to evaluate the user experience with pointing interaction in different service settings will be accomplished over evolved prototypes. It will enable measuring the system performance against other interaction options, the learning curve or the hindrances for user adoption.

Acknowledgment

This work has been supported by the CDTI CENIT THOFU Programme. Additionally, authors want to thank the eight volunteers of the ETSIT - UPM that have kindly provided their help for the completion of the study.

References

[1] G. Imai. "Gestures: Body language and nonverbal communication." Retrieved Oct 2005.

[2] R.A. Bolt. "Put-that-there: Voice and gesture at the graphics interface". Vol. 14, no. 3. ACM, 1980.

[3] K. Nickel and R. Stiefelhagen. "Real-time recognition of 3d-pointing gestures for human-machine-interaction." In Pattern Recognition, pp. 557-565. Springer Berlin Heidelberg, 2003.

[4] H. Kaoning, S. Canavan, Y. Lijun, "Hand Pointing Estimation for Human Computer Interaction Based on Two Orthogonal-Views," Intl. Conference on Pattern Recognition, pp.3760-3763, 23-26 Aug. 2010.

[5] J. Segen, S. Kumar, "Human-computer interaction using gesture recognition and 3D hand tracking," 1998 International Conference on Image Processing, pp.188-192 vol.3, 4-7 Oct 1998.

[6] G. Yepeng, Z. Mingen, "Real-time 3D pointing gesture recognition for natural HCI," 7th World Congress on Intelligent Control and Automation, pp.2433-2436, 25-27 June 2008.

[7] R. Kehl, L. Van Gool, "Real-time pointing gesture recognition for an immersive environment", IEEE International Conference on Automatic Face and Gesture Recognition, pp. 577-582, 17-19 May 2004.

[8] N. Jojic, B. Brumitt, B. Meyers, S. Harris and T. Huang. "Detection and estimation of pointing gestures in dense disparity maps." Intl. Conf. on Automatic Face and Gesture Recognition, pp. 468-475. IEEE, 2000.

[9] L. Xia, K. Fujimura. "Hand gesture recognition using depth data." Intl. Conf. on Automatic Face and Gesture Recog. pp. 529-534. IEEE, 2004.

[10] M. Van den Bergh, D. Carton, R. De Nijs, N. Mitsou, C. Landsiedel, K. Kuehnlenz, D. Wollherr, L. Van Gool, and M. Buss. "Real-time 3D hand gesture interaction with a robot for understanding directions from humans." In RO-MAN IEEE, pp. 357-362. IEEE, 2011.

[11] M. Van den Bergh and L. Van Gool. "Combining RGB and ToF cameras for real-time 3D hand gesture interaction." In IEEE Workshop on Applications of Computer Vision, pp. 66-72. IEEE, 2011.

[12] N.-K. Chuan and A. Sivaji, "Combining eye gaze and hand tracking for pointer control in HCI: Developing a more robust and accurate interaction system for pointer positioning and clicking". IEEE Colloquium on Humanities, Science and Engineering (CHUSER), pp. 172-176, 3-4 Dec. 2012.

[13] M.J. Reale, S. Canavan, L. Yin, K. Hu and T. Hung. "A multi-gesture interaction system using a 3-D Iris disk model for gaze estimation and an active appearance model for 3-D hand pointing." IEEE Transactions on Multimedia, 13, no. 3, pp. 474-486, 2011.

[14] M. Beigl, "Point & Click – Interaction in Smart Environments". Procs. of the First Int. Symposium on Handheld and Ubiquitous Computing, LNCS 1707, Springer-Verlag, pp. 311-313, 1999.

[15] G. Broll, M. Paolucci, M. Wagner, E. Rukzio, A. Schmidt, H. Hubmann. "Perci: Pervasive Service Interaction with the Internet of Things". IEEE Internet Computing, vol. 13, no. 6, pp. 74-81, 2009.

[16] A. Wilson, S. Shafer. "XWand: UI for intelligent spaces." Proc. Conference on Human factors in Comp. Sys., pp. 545-552. ACM, 2003.

[17] D. Pasquariello, M. Vissenberg, G.J. Destura, "Remote-Touch: A Laser Input User–Display Interaction Technology," Journal of Display Technology, vol. 4, no.1, pp.39-46, March 2008.

[18] T. Baudel, M. Beaudouin-Lafon. "Charade: remote control of objects using free-hand gestures." Comm. of the ACM 36, 7, 28-35, 1993.

[19] D. Kim, O. Hilliges, S. Izadi, A.D. Butler, J. Chen, I. Oikonomidis, and P. Olivier. "Digits: freehand 3D interactions anywhere using a wrist-worn gloveless sensor." Proc. of the 25th annual ACM symposium on User interface software and technology, pp. 167-176. ACM, 2012.

[20] C.C. Kemp, C.D. Anderson, H. Nguyen, A.J. Trevor, X. Zhe Xu, "A point-and-click interface for the real world: Laser designation of objects for mobile manipulation," 3rd ACM/IEEE International Conference on Human-Robot Interaction (HRI), pp.241-248, 12-15 March 2008.

[21] K. Nickel and R. Stiefelhagen. "Real-time person tracking and pointing gesture recognition for human-robot interaction." Computer Vision in Human-Comp. Interaction, pp. 28-38. Springer Berlin Heidelberg, 2004.

[22] J. Heikkila, O. Silvén, "A four-step camera calibration procedure with implicit image correction", Procs. IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE, 1997.

[23] J.-Y. Bouguet, Jean-Yves. "Complete camera calibration toolbox for matlab", 2004, http://www.vision.caltech.edu/bouguetj/calib_doc/.

[24] J.M. Christian Bastien, "Usability testing: a review of some methodological and technical aspects of the method", Intl. J. of Medical Informatics, Vol. 79, Is. 4, pp. 18-23, 2010.