Did you Mean this Object?: Detecting Ambiguity in Pointing Gesture Targets

Akansel Cosgun,*Alexander J. B. Trevor and Henrik I. Christensen
Georgia Institute of Technology
Institute for Robotics and Intelligent Machines (IRIM)
{acosgun,atrevor,hic}@cc.gatech.edu

ABSTRACT

Pointing gestures are often used to refer to objects, places and people. Robot perception systems that are able to correctly interpret pointing gestures can benefit human-robot interaction. When there are several potential pointing targets in the scene, such as multiple objects, the intended target may be ambiguous. Robots that are able to detect such ambiguities can act more intelligently by requesting clarification from a user. In this work, we model the uncertainty of pointing gestures in a spherical coordinate system, use this model to determine the correct pointing target, and detect when there is ambiguity. Two pointing methods are evaluated using two skeleton tracking algorithms: elbow-hand and head-hand rays, using both OpenNI NITE and Microsoft Kinect SDK. Our evaluation shows that our model can be used to detect when there is an ambiguous pointing target, and when it is clearly determined which object was referenced.

Keywords

Pointing Gestures; Domestic robots; Joint Attention

Categories and Subject Descriptors

I.2.10 [**Artificial Intelligence**]: Vision and Scene Understanding; I.4.8 [**Image Processing and Computer Vision**]: Scene Analysis

1. INTRODUCTION

Humans and robots engaged in joint actions that involve objects will need to be able to communicate about these objects. Deictic gestures can play a crucial part such communication, especially when establishing goals and negotiating responsibilities. In previous work, we developed an approach for recognizing human deictic gestures in RGB-D sensors for annotating object models [27] and 3D maps [28], to establish common ground, and enable these labels to be referenced in tasks. In this work, we analyze the performance of our deictic gesture recognition, and present an uncertainty model that enables us to reason about ambiguity in pointing gestures, when objects are too close to one another to determine the referent.We model the uncertainty of pointing gestures in a spherical coordinate system, use this model to determine the correct pointing target, and detect when there is ambiguity. Two pointing methods are evaluated using two skeleton tracking algorithms: elbow-hand and head-hand rays, using both OpenNI NITE and Microsoft Kinect SDK [26]. A data collection with 6 users and 7 pointing targets was performed, and the data was used to model users pointing behavior. The resulting model was evaluated for its ability to distinguish between potential pointing targets, and to detect when the target is ambiguous. An example scenario is shown in Figure 1.

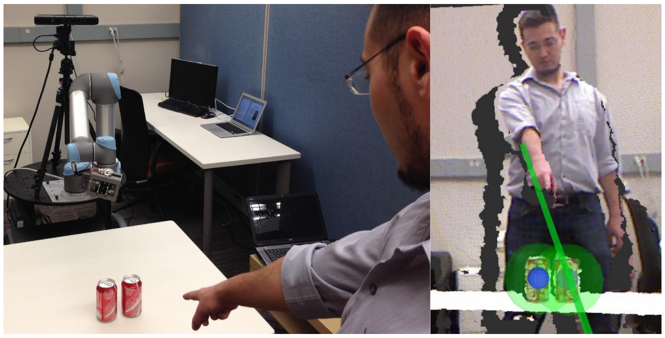


Figure 1: (Left) Our approach allows a robot to detect when there is ambiguity on the pointing gesture targets. (Right) The point cloud view from robot's perspective is shown. Both objects are identified as potential intended targets, therefore the robot decides that there is ambiguity.

After a brief survey of the literature in Section 2, our gesture recognition approach is outlined in Section 3. We then present our spherical coordinate model in Section 4 and how intended targets are determined in Section 5. Our data collection process is described in Section 6 and the results of the error analysis is reported in Section 7. We evaluate our model for two potentially ambiguous pointing targets in Section 8, before concluding in Section 9.

2. RELATED WORK

Pointing gestures are widely used in Human-Robot Interaction applications. Examples include interpretation of spoken requests [31], pick and place tasks [3], joint attention [8], referring to places [10] and objects [25], instructing [19] and providing navigation goals [24] to a robot.

Early works on recognizing pointing gestures in stereo vision utilized background subtraction [6, 13, 14]. Other popular meth-

*A. Cosgun and A. J. B. Trevor contributed equally to this work.

HRI'15 Towards a Framework for Joint Action Workshop, Mar 02, 2015, Portland, OR, USA

ods include body silhouettes [15], hand poses [12], motion analysis [20] and Hidden Markov Models [30, 2, 18, 22, 7, 1]. Matuszek *et. al* presented a method for detecting deictic gestures given a set of detected tabletop objects, by first segmenting the users hands and computing the most distal point form the user, then applying a Hierarchical Matching Pursuit (HMP) approach on these points over time [21].

After deciding if a pointing gesture occurred or not, an algorithm must estimate the direction of pointing. This is typically done by extending a ray from one body part to another. Several body part pairs are used in the literature, such as eye-fingertip [15] and shoulder-hand [11]; with the two of most commonly used methods being elbow-hand [24, 4, 3] and head-hand [2, 25] rays. Some studies found that head-hand approach is a more accurate way for estimating pointing direction than elbow-hand [23, 8]. Recent works made use of skeleton tracking data from in depth images [23, 3, 24]. Other approaches, such as measuring head orientation with a magnetic sensor [22] and pointing with a laser pointer [5, 16] is reported to have a better estimation accuracy than the body parts method, but require additional hardware. We prefer not to use additional devices in order to the interaction as natural as possible.

Given a pointing direction, several methods have been proposed to determine which target or object is referred by the gesture, including euclidean distance on a planar surface [5], ray-to-cluster intersection in point clouds [3, 23] and searching a region in interest around the intersection point [25]. Droeschel [7] trains a function using head-hand, elbow-hand and shoulder-hand features with Gaussian Process Regression and reports a significant improvement on pointing accuracy. Some efforts fuse speech with pointing gestures for multi-modal Human-Robot Interaction [1, 17]. Aly [1] focuses on relation between non-verbal arm gestures and para-verbal communication based on a HMM approach.

To our knowledge, only Zukerman [32] and Kowadlo [17] considered a probabilistic model for determining the referred object for a pointing gesture. The probability that the user intended an object is calculated using the 2D distance to the object and the occlusion factor. Objects that reside in a Gaussian cone emanating from the user's hand are considered as candidates in the model. The approach is implemented in [32], where it is reported that due to high variance of the gesture recognition system, the Gaussian cone typically encompassed about five objects in cluttered settings. Our work addresses the confusion in such settings. In contrast to their work, we use a 3D elliptical Gaussian cone where its shape is extracted using a prior error analysis, and use point cloud data to account for the size of the objects.

3. POINTING GESTURE RECOGNITION

Our approach to pointing gesture recognition is to use a skeleton tracking algorithm as implemented by OpenNI NITE 1.5 (OpenNI) or Microsoft Kinect SDK 1.5 (MS-SDK). Skeleton tracking software produces 3D positions for several important points on the user's body, including hands, elbows, shoulders and head. Our points of interests are user's hands, elbows, and head for the recognition of pointing gestures.

3.1 Types of Pointing Gestures

We are primarily interested in deictic gestures generated by pointing with one's arm. We consider two rays for determining the pointing direction: elbow-hand and head-hand. Both of these methods were evaluated with the two skeleton tracking implementations. For each depth frame, this yields two rays for each of the OpenNI and MS-SDK trackers:

- $\vec{v}_{eh} := \overrightarrow{p_{elbow}p_{hand}}$
- $\vec{v}_{hh} := \overrightarrow{p_{head}p_{hand}}$

3.2 Pointing Gesture Recognition

In previous work [28], we described a pointing gesture recognition method, summarized here. When a pointing gesture recognition request is received from a higher level process, the gesture is searched in a time window of T seconds. Two conditions must be met to trigger a pointing gesture:

- Forearm makes an angle more than ϕ_g with the vertical axis
- Elbow and hand points stays near-constant for duration Δt_g

The first condition requires the arm of the person to be extended, while the second ensures that the gesture is consistent for some time period. The parameters are empirically determined as: $T = 30s$, $\phi_g = 45°$ and $\Delta t_g = 0.5s$.

4. REPRESENTING POINTING DIRECTIONS

We represent a pointing ray in two angles: a "horizontal" / "azimuth" sense we denote as θ and a "vertical" / "altitude" sense we denote as ψ. We first attach a coordinate frame to the hand point, with its z-axis oriented in either Elbow-hand $\vec{v}_{eh}$ or Head-Hand $\vec{v}_{hh}$ directions. The hand was chosen as the origin for this coordinate system because both of head-hand and elbow-hand pointing methods include the user's hand. The transformation between the sensor frame and the hand frame ${}^{sensor}T_{hand}$ is calculated by using an angle-axis rotation. An illustration of the hand coordinate frame for Elbow-Hand method and corresponding angles are shown graphically in Figure 2.

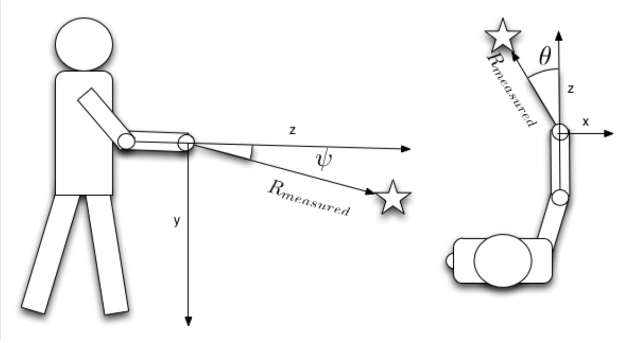


Figure 2: Vertical (ψ) and horizontal (θ) angles in spherical coordinates are illustrated. A potential intended target is shown as a star. The z-axis of the hand coordinate frame is defined by either the Elbow-Hand (this example) or Head-Hand ray.

Given this coordinate frame and a potential target point P, we first transform it to the hand frame by:

$$^{hand}p_{target} = T_{hand} * p_{target}$$

We calculate the horizontal and vertical angles for a target point as ${}^{hand}p_{target} = (x_{targ}, y_{targ}, z_{targ})$ follows:

$$[\theta_{target}\ \psi_{target}] = [atan2(x_{targ}, z_{targ})\ atan2(y_{targ}, z_{targ})]$$

Where $atan2(y, x)$ is a function returns the value of the angle $\arctan(\frac{y}{x})$ with the correct sign. This representation allows us to calculate the angular errors in our error analysis experiments in Section 7. The angles for each object is then used to find the intended target, as explained in the following section.

5. DETERMINING INTENDED TARGET

We propose a probabilistic approach to determine the referred target by using statistical data from previous pointing gesture observations. We observed that head-hand and elbow-hand methods, implemented using two skeleton trackers, returned different angle errors in spherical coordinates. Our approach relies on learning statistics of each of these approaches, and compensating the error when the target object is searched for. First, given a set of prior pointing observations, we calculate the mean and variance of the vertical and horizontal angle errors for each pointing method. This analysis will be presented in Section 7. Given an input gesture, we apply correction to the pointing direction and find the Mahalanobis distance to each object in the scene.

When a pointing gesture is recognized, and the angle pair

$$[\theta_{target} \ \psi_{target}]$$

is found as described in the previous section, we first apply a correction by subtracting the mean terms from measured angles:

$$[\theta_{cor} \ \ \psi_{cor}] = [\theta_{target} - \mu_\theta \quad \psi_{target} - \mu_\psi]$$

We also compute a covariance matrix for angle errors in this spherical coordinate system:

$$S_{type} = \begin{bmatrix} \sigma_\theta & 0 \\ 0 & \sigma_\psi \end{bmatrix}$$

We get the values for $\mu_\theta, \mu_\psi, \sigma_\theta, \sigma_\psi$ from Tables 1 and 2 for the corresponding gesture type and target. We then compute the mahalanobis distance to the target by:

$$D_{mah}(target, method) = \sqrt{[\theta_{cor} \ \psi_{cor}]^T S_{method}^{-1} [\theta_{cor} \ \psi_{cor}]}$$

We use D_{mah} to estimate which target or object is intended. We consider two use cases: the objects are represented as a 3D point or a point cloud. For point targets, we first filter out targets that have a Mahalanobis distance larger than a threshold $D_{mah} > D_{thresh}$. If none of the targets has a D_{mah} lower than the threshold, then we say the user did not point to any targets. If there are multiple targets that has $D_{mah} <= D_{thresh}$, then we determine ambiguity by employing a ratio test. The ratio of the least D_{mah} and the second-least D_{mah} among all targets is compared with a threshold to determine if there is ambiguity. If the ratio is higher than a threshold, then the robot can ask the person to solve the ambiguity.

If the target objects are represented as point clouds, we then compute the horizontal and vertical angles for every point in the point cloud and find the minimum mahalanobis distance among all. The distance to an object is then represented by this minimum value. Usage of the point cloud instead of the centroid for determining the intended object has several advantages. First, it yields better estimations due to the coverage of the full point cloud. Second, it takes into account the size of the object. For example, if a person is pointing to a chair or door, it is very unlikely that he/she will target the centroid of it. If the point cloud is used, then we can easily tell that the object is targeted.

6. DATA COLLECTION

To evaluate the accuracy of pointing gestures, we created a test environment with 7 targets placed on planar surfaces in view of a Kinect sensor (Figure 3). Depth data was collected from six users, who pointed at each of the seven targets with their right arm while standing at 2 meters away from the sensor. Targets 1 through 4 were on tables positioned around the user, while targets 5 through 7 were located on a wall to the user's right. Our use case is on a mobile robot platform capable of positioning itself relative to the user. For this reason, we can assume that the user is always centered in the image, as the robot can easily rotate to face the user and can position itself at a desired distance from the user.

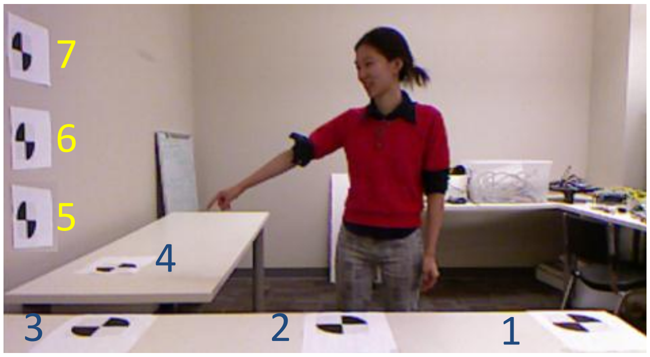


Figure 3: Our study involved 6 users that pointed to 7 targets while being recorded using 30 frames per target.

6.1 Ground Truth Target Positions

We make use of plane extraction technique in a point cloud to have an accurate ground truth measurement. First, the two table and wall planes are extracted from the point cloud data using the planar segmentation technique described in [29]. We then find the pixel coordinates of corners on targets in RGB images, using Harris corner detection [9], which produces calculated corners in image coordinates with sub-pixel accuracy. The pixel coordinate corresponding to each target defines a ray in 3D space relative to the Kinect's RGB camera. These rays are then intersected with the planes detected from the depth data, yielding the 3D coordinates of the targets.

6.2 Skeleton Data Capture

In order to be able to do a fair comparison between MS-SDK and OpenNI skeleton trackers, we used the same dataset for both. MS-SDK and OpenNI use different device drivers, therefore can not be directly used on the live depth stream at the same time. Because of this, we extract the skeleton data offline in multiple stages. The pipeline for the data capture procedure is illustrated in Figure 4. We first save the depth streams as .xed files using Microsoft Kinect Studio program. The acquired .xed file is converted to .oni in a OpenNI recorder application by streaming the depth stream to through Kinect Studio. The .oni is then played back in skeleton

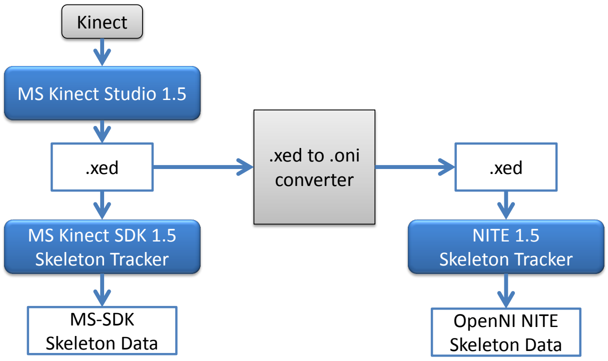

Kinect
MS Kinect Studio 1.5
.xed
.xed to .oni converter
.xed
MS Kinect SDK 1.5 Skeleton Tracker
NITE 1.5 Skeleton Tracker
MS-SDK Skeleton Data
OpenNI NITE Skeleton Data

Figure 4: Data capturing pipeline for error analysis.

tracking application in OpenNI, which writes the OpenNI skeleton data to a .txt file. MS-SDK skeleton data is obtained by playing back the original .xed in the skeleton tracking application.

6.3 Pointing Gesture Annotation

To factor out the effectiveness of our pointing gesture recognition method described in Section 3.2, we manually labeled when each pointing gesture began for data collection. Starting from the onset of the pointing gesture as annotated by the experimenter, the following 30 sensor frames were used as the pointing gesture. This corresponds to a recording of 1 second in the Kinect sensor stream. For each frame, we extracted skeleton data using both the MS-SDK and the OpenNI.

7. ERROR ANALYSIS

The four rays corresponding to the four different pointing approaches described in Section 3.1 were used for our error analysis. As described in Section 6.1, the ground truth target positions are available. We computed two types of errors for each gesture and target:

- Euclidean error of ray intersections with target planes (Figure 5)
- Angular error in spherical coordinates (Tables 1,2 and Figure 6)

We elaborate our error analysis subsequent sections:

7.1 Euclidean error

Given a ray $\vec{v}$ in the sensors frame from one of the pointing gesture approaches, and a ground truth target point p_{target} lying on a target planar surface, the ray-plane intersection between $\vec{v}$ and plane was computed for each ray, resulting in a 3D point lying on the plane. Figure 5 shows the 2D projections for all 30 measurements from each subject (shown in different colors) and each target. For ease of display, the 3D intersection coordinates with the target planes are displayed in a 2D coordinate system attached to the target plane, with the ground truth target point as the origin.

As can be seen in Figure 5, the pointing gesture intersection points were fairly consistent across all users, but varied per target location. The elbow-hand method produced similar results for MS-SDK and OpenNI. The same is true for the head-hand method. It is also noteworthy that the data for each target tends to be quite clustered for all methods, and typically not centered on the target location.

7.2 Angular Error

We computed the mean and standard deviations of the angular errors in the spherical coordinate system for each pointing gesture method and target. Section 4 describes in detail how the angles (θ, ψ) are found. The mean and standard deviation analyses are given in Tables 1 and 2. The aggregate errors are also displayed as a box plot in Figure 6.

Several observations can be made from these results. The data from the elbow-hand pointing method reports that users typically point about 11 degrees to the left of the intended target direction, and about 9 above the target direction. Similarly, the data from the head-hand pointing method reports that users typically point about 2 degrees to the left of the intended pointing direction, but with a higher standard deviation than the elbow-hand method. On average, the vertical angle ψ was about 5 degrees below the intended direction for the OpenNI tracker and 10 degrees below for the MS-

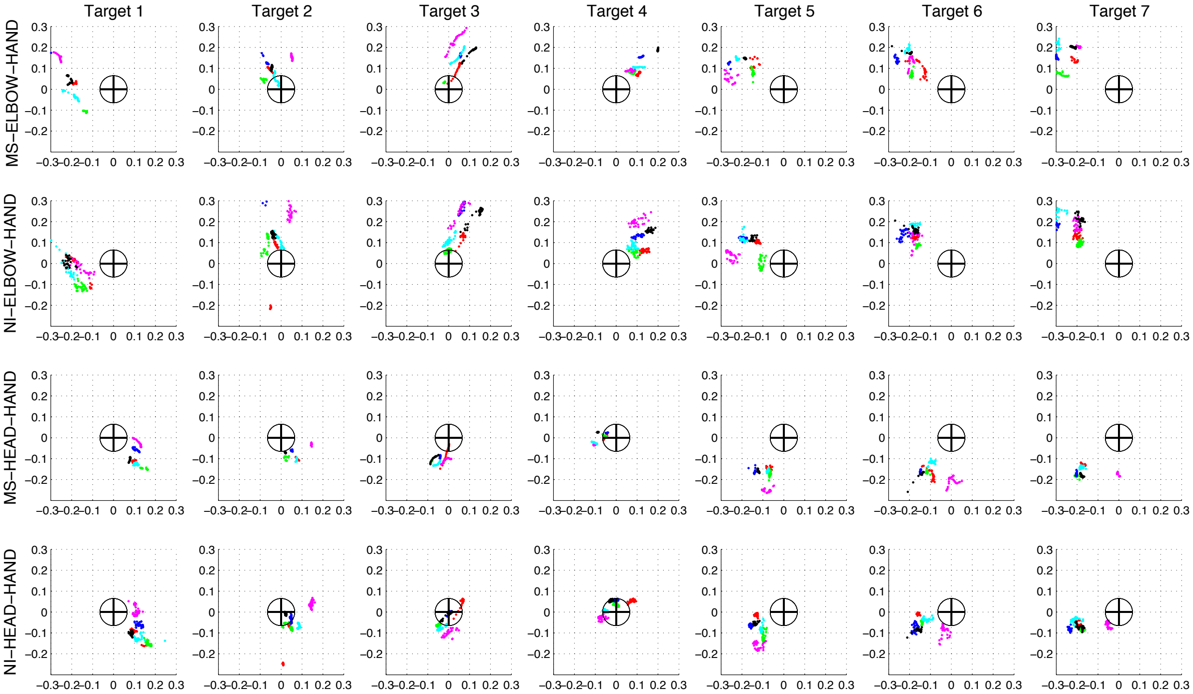


Figure 5: Euclidean distance error in cartesian coordinates for each method and target. The gesture ray intersection points in centimeters with the target plane, are shown here for each target (T1-T7) as the columns. Each subject's points are shown in separate colors. There are 30 points from each subject, corresponding to the 30 frames recorded for the pointing gesture at each target. Axes are shown in centimeters. The circle drawn in the center of each plot has the same diameter (13 cm) as the physical target objects used.

	Target 1				Target 2				Target 3				Target 4			
	θ		ψ		θ		ψ		θ		ψ		θ		ψ	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
MS-Elbow-Hand	-15.7	2.9	5.5	6.1	-4.3	6.6	7.8	3.1	-3.7	2.4	7.4	3.1	-3.6	1.9	11.5	2.9
NI-Elbow-Hand	-16.4	2.9	4.3	7.0	-3.8	6.6	11.3	10.9	-4.8	2.6	9.7	3.4	-4.0	4.7	12.4	2.5
MS-Head-Hand	7.7	2.6	-12.0	5.3	10.8	6.4	-9.1	3.2	2.2	2.0	-8.3	3.2	-4.0	1.7	-4.3	3.2
NI-Head-Hand	8.5	2.6	-11.7	6.1	10.2	6.7	-5.7	8.0	2.0	2.3	-2.9	4.7	-3.2	2.5	1.45	4.8

Table 1: μ and σ angular errors in degrees for each of Targets 1-4 (Figure 3), for each pointing method.

	Target 5				Target 6				Target 7				ALL TARGETS			
	θ		ψ		θ		ψ		θ		ψ		θ		ψ	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
MS-Elbow-Hand	-14.5	4.1	11.6	3.7	-16.6	3.3	9.9	3.7	-20.8	3.7	5.7	3.4	-11.3	7.7	8.5	4.5
NI-Elbow-Hand	-12.9	4.2	9.7	5.1	-16.2	2.6	11.7	3.7	-20.2	4.5	8.1	-3.0	-11.2	7.6	9.6	6.3
MS-Head-Hand	-9.4	1.9	-11.6	3.0	-7.1	4.7	-13.8	1.8	-8.0	5.5	-15.6	-2.9	-1.1	8.5	-10.7	4.8
NI-Head-Hand	-11.5	1.5	-4.7	4.8	-11.2	4.8	-4.9	2.9	-12.3	5.2	-8.9	-2.5	-2.4	9.6	-5.3	6.4

Table 2: μ and σ of angular error in degrees for Targets 5-7 (Figure 3), for each pointing method. The aggregate μ and σ is also shown.

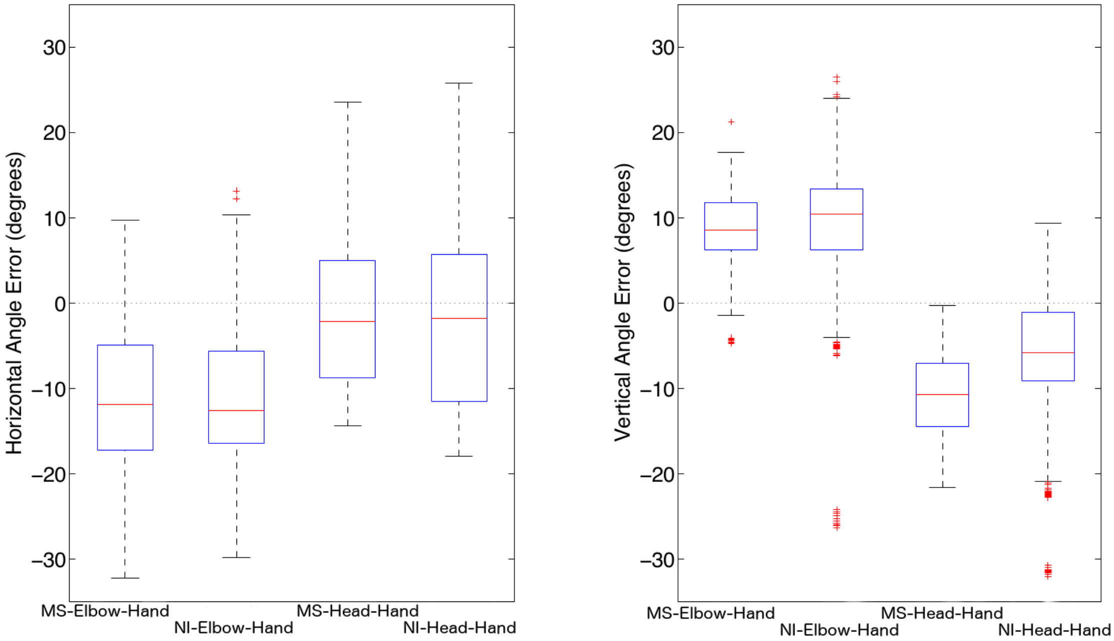


Figure 6: Box plots of the errors in spherical coordinates θ and ψ for each pointing method.

SDK tracker, with a higher standard deviation than the elbow-hand methods. The disparity between the two skeleton trackers for this pointing method is because they report different points for the head position, with the MS-SDK head position typically being reported higher than the OpenNI head position. The overall performance of the OpenNI and MS-SDK skeleton trackers, however, is fairly similar, with the MS-SDK having slightly less variation for our dataset.

As can be seen in the aggregate box plot in Figure 6, the horizontal angle θ has a higher variation than the vertical angle ψ. Examining the errors for individual target locations shows that this error changes significantly with the target location. As future work, it would be interesting to collect data for a higher density of target locations to attempt to parameterize any angular correction that might be applied.

8. EVALUATION

Using the error analysis and pointing gesture model we presented in previous sections, we conducted an experiment to determine how our approach distinguished two potentially ambiguous pointing target objects. We use the results of the angular error analysis results and not the euclidean error analysis for the remainder of the paper because of our angular representation of pointing gestures.

8.1 Object Separation Test

The setup consisted of a table between the robot and the person and two coke cans on the table (Figure 8) where the separation between objects was varied. To detect the table plane and segment out the objects on top of it, we used the segmentation approach presented in [29]. The objects centroid positions, along with their point clouds were calculated in real-time using our segmentation

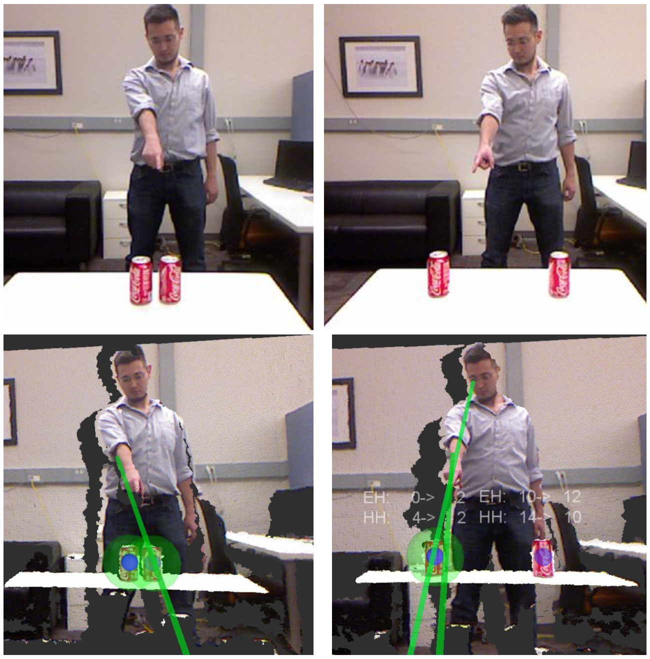


Figure 8: Example scenarios from the object separation test is shown. Our experiments covered separations between 2cm (left images) and 30cm (right images). The object is comfortably distinguished for the 30cm case, whereas the intended target is ambiguous when the targets are 2cm apart. Second row shows the point cloud from Kinect's view. Green lines show the Elbow-Hand and Head-Hand directions whereas green circles show the objects that are within the threshold $D_{mah} < 2$.

algorithm. The separation between objects were varied with 1 cm increments from 2-15 cm and with 5 cm increments between 15-30 cm. We could not conduct the experiment below 2 cm separation because of the limitations of our perception system. We used the OpenNI skeleton tracker because rest of our system is based in Linux, and we already found that performance difference with MS-SDK for pointing angle errors is insignificant. The experiment was conducted with one user, who was not in the training dataset. For each separation, the user performed 5 pointing gestures to the object on the right and 5 to the object on the left. The person pointed to one of the objects and the Mahalanobis distance D_{mah} to the intended object and the other object is calculated using the approach in Section 5. We used the mean and standard deviation values of Target 2 (Figure 3) for this experiment because the objects were located between the robot and the person.

8.2 Results and Discussion

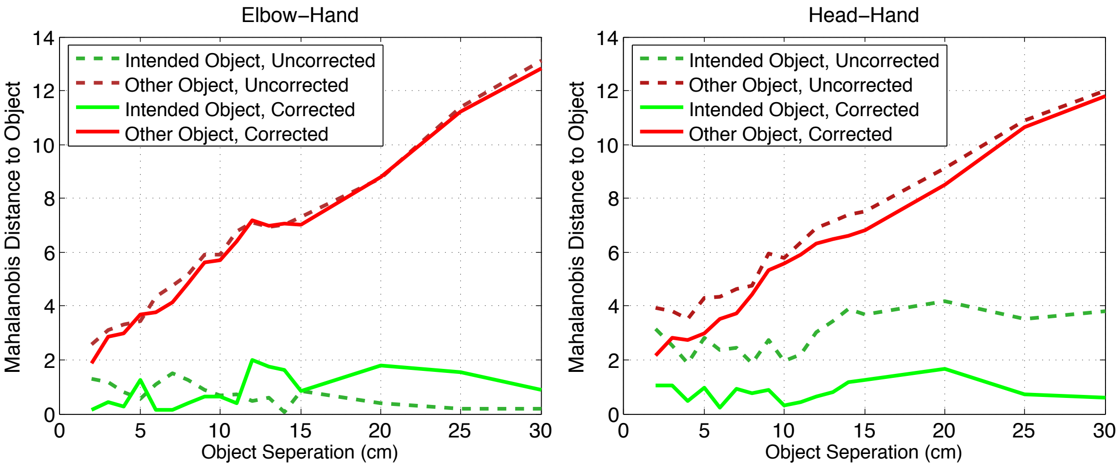
The results of the object separation experiment is given for Elbow-Hand (Figure 7(a)) and Head-Hand (Figure 7(b)) methods. The graphs plot object separation versus the Mahalanobis distance for the pointed object and the other object for corrected and uncorrected pointing direction. There are several observations we make by looking at these results.

First, the Mahalanobis distance D_{mah} for the intended object was always lower than the other object. The corrected D_{mah} for both Elbow-Hand and Head-Hand methods for the intended object was always below 2, therefore selecting the threshold $D_{thres} = 2$ is a reasonable choice. We notice that some distances for the unintended object at 2cm separation is also below $D_{mah} < 2$. Therefore, when the objects are 2 cm apart, then the pointing target becomes ambiguous for this setup. For separations of 3cm or more, D_{mah} of the unintended object is always over the threshold, therefore there is no ambiguity.

Second, correction significantly improved Head-Hand accuracy at all separations, slightly improved Elbow-Hand between 2-12cm but slightly worsened Elbow-Hand after 12cm. We attribute this to the fact that the angles we receive is heavily user-dependent and can have a significant variance across methods as showed in Figure 6. Moreover, the user was not in the training set.

Third, the Mahalanobis distance stayed generally constant for the intended object, which was expected. It linearly increased with separation distance for the other object.

Fourth, patterns for both methods are fairly similar to each other, other than Head-Hand uncorrected distances being higher than Elbow-Hand.



(a) (b)

Figure 7: Resulting Mahalanabis distances of pointing targets from the Object Separation Test is shown for a) Elbow-Hand and b) Head-Hand pointing methods. Intended object are shown in green and other object is in red. Solid lines show distances after correction is applied. Less Mahalanobis distance for intended object is better for reducing ambiguity.

9. CONCLUSIONS

Humans and robots engaged in joint actions that involve objects will need to be able to communicate about these objects. Deictic gestures can play a crucial part such communication, especially when establishing goals and negotiating responsibilities. A pointing target detection approach that returns a single hypothesis can lead to failures due to perception error. On the other hand, estimation of a pointing likelihood of nearby objects can give valuable insight to a robot. For example, a robot can ask the user to disambiguate the objects if it perceives more than one object that has a significant likelihood of being referred to.

In this work, we model the uncertainty of pointing gestures in a spherical coordinate system, use this model to determine the correct pointing target, and detect when there is ambiguity. Two pointing methods are evaluated using two skeleton tracking algorithms: Elbow-Hand and Head-Hand rays, using both OpenNI NITE and Microsoft Kinect SDK. A data collection with 6 users and 7 pointing targets was performed, and the data was used to model users pointing behavior. The resulting model was evaluated for its ability to distinguish between potential pointing targets, and to detect when the target is ambiguous. Our evaluation showed that in a scenario where the separation between two objects were varied, our system was able to identify that there is ambiguity for 2 cm separation and comfortably distinguished the intended object for 3 cm or more separation.

Acknowledgement

This work was made possible through financial support from the BMW corporation and the Boeing corporation.

References

[1] A. Aly and A. Tapus. An integrated model of speech to arm gestures mapping in human-robot interaction. In *Information Control Problems in Manufacturing*, volume 14, pages 817–822, 2012.

[2] 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.

[3] N. Blodow, Z.-C. Marton, D. Pangercic, T. Rühr, M. Tenorth, and M. Beetz. Inferring generalized pick-and-place tasks from pointing gestures. In *IEEE International Conference on Robotics and Automation (ICRA), Workshop on Semantic Perception, Mapping and Exploration*, 2011.

[4] A. G. Brooks and C. Breazeal. Working with robots and objects: Revisiting deictic reference for achieving spatial common ground. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 297–304. ACM, 2006.

[5] K. Cheng and M. Takatsuka. Hand pointing accuracy for vision-based interactive systems. In *Human-Computer Interaction–INTERACT 2009*, pages 13–16. Springer, 2009.

[6] R. Cipolla and N. J. Hollinghurst. Human-robot interface by pointing with uncalibrated stereo vision. *Image and Vision Computing*, 14(3):171–178, 1996.

[7] D. Droeschel, J. Stuckler, and S. Behnke. Learning to interpret pointing gestures with a time-of-flight camera. In *Human-Robot Interaction (HRI), 2011 6th ACM/IEEE International Conference on*, pages 481–488. IEEE, 2011.

[8] D. Droeschel, J. Stuckler, D. Holz, and S. Behnke. Towards joint attention for a domestic service robot-person awareness and gesture recognition using time-of-flight cameras. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1205–1210. IEEE, 2011.

[9] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK, 1988.

[10] Y. Hato, S. Satake, T. Kanda, M. Imai, and N. Hagita. Pointing to space: modeling of deictic interaction referring to regions. In *Proceedings of the 5th ACM/IEEE international conference on Human-robot interaction*, pages 301–308. IEEE Press, 2010.

[11] E. Hosoya, H. Sato, M. Kitabata, I. Harada, H. Nojima, and A. Onozawa. Arm-pointer: 3d pointing interface for real-world interaction. In *Computer Vision in Human-Computer Interaction*, pages 72–82. Springer, 2004.

[12] K. Hu, S. Canavan, and L. Yin. Hand pointing estimation for human computer interaction based on two orthogonal-views. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 3760–3763. IEEE, 2010.

[13] N. Jojic, B. Brumitt, B. Meyers, S. Harris, and T. Huang. Detection and estimation of pointing gestures in dense disparity maps. In *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*, pages 468–475. IEEE, 2000.

[14] R. E. Kahn and M. J. Swain. Understanding people pointing: The perseus system. In *Computer Vision, 1995. Proceedings., International Symposium on*, pages 569–574. IEEE, 1995.

[15] R. Kehl and L. Van Gool. Real-time pointing gesture recognition for an immersive environment. In *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, pages 577–582. IEEE, 2004.

[16] C. C. Kemp, C. D. Anderson, H. Nguyen, A. J. Trevor, and Z. Xu. A point-and-click interface for the real world: laser designation of objects for mobile manipulation. In *Human-Robot Interaction (HRI), 2008 3rd ACM/IEEE International Conference on*, pages 241–248. IEEE, 2008.

[17] G. Kowadlo, P. Ye, and I. Zukerman. Influence of gestural salience on the interpretation of spoken requests. In *INTERSPEECH*, pages 2034–2037, 2010.

[18] Z. Li, N. Hofemann, J. Fritsch, and G. Sagerer. Hierarchical modeling and recognition of manipulative gesture. In *Proc. of the Workshop on Modeling People and Human Interaction at the IEEE Int. Conf. on Computer Vision*, volume 77, 2005.

[19] C. Martin, F.-F. Steege, and H.-M. Gross. Estimation of pointing poses for visually instructing mobile robots under real world conditions. *Robotics and Autonomous Systems*, 58(2):174–185, 2010.

[20] P. Matikainen, P. Pillai, L. Mummert, R. Sukthankar, and M. Hebert. Prop-free pointing detection in dynamic cluttered environments. In *Automatic Face & Gesture Recognition and Workshops (FG 2011), 2011 IEEE International Conference on*, pages 374–381. IEEE, 2011.

[21] C. Matuszek, L. Bo, L. Zettlemoyer, and D. Fox. Learning from unscripted deictic gesture and language for human-robot interactions. 2014.

[22] K. Nickel and R. Stiefelhagen. Pointing gesture recognition based on 3d-tracking of face, hands and head orientation. In *Proceedings of the 5th international conference on Multimodal interfaces*, pages 140–146. ACM, 2003.

[23] C. P. Quintero, R. T. Fomena, A. Shademan, N. Wolleb, T. Dick, and M. Jagersand. Sepo: Selecting by pointing as an intuitive human-robot command interface. In *IEEE Int. Conference of Robotics and Automation, Karslruhe, Germany*, 2013.

[24] S. S. Raza Abidi, M. Williams, and B. Johnston. Human pointing as a robot directive. In *Proceedings of the 8th ACM/IEEE international conference on Human-robot interaction*, pages 67–68. IEEE Press, 2013.

[25] J. Schmidt, N. Hofemann, A. Haasch, J. Fritsch, and G. Sagerer. Interacting with a mobile robot: Evaluating gestural object references. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 3804–3809. IEEE, 2008.

[26] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1297–1304. IEEE, 2011.

[27] A. J. Trevor, J. G. Rogers III, A. Cosgun, and H. I. Christensen. Interactive object modeling & labeling for service robots. In *Proceedings of the 8th ACM/IEEE international conference on Human-robot interaction*, pages 421–422. IEEE Press, 2013.

[28] A. J. B. Trevor, A. Cosgun, J. Kumar, and H. I. Christensen. Interactive map labeling for service robots. In *IROS 2012 Workshop on Active Semantic Perception*, 2012.

[29] A. J. B. Trevor, S. Gedikli, R. Rusu, and H. I. Christensen. Efficient organized point cloud segmentation with connected components. In *ICRA Workshop on Semantic Perception Mapping and Exploration*, 2013.

[30] A. D. Wilson and A. F. Bobick. Parametric hidden markov models for gesture recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(9):884–900, 1999.

[31] I. Zukerman, G. Kowadlo, and P. Ye. Interpreting pointing gestures and spoken requests: a probabilistic, salience-based approach. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 1558–1566. Association for Computational Linguistics, 2010.

[32] I. Zukerman, A. Mani, Z. Li, and R. Jarvi. Speaking and pointing?from simulations to the laborator. *Knowledge and Reasoning in Practical Dialogue Systems*, page 58, 2011.