Real-Time Recognition of Pointing Gestures for Robot to Robot Interaction

Polychronis Kondaxakis, Joni Pajarinen and Ville Kyrki*

***Abstract*— This paper addresses the idea of establishing symbolic communication between mobile robots through gesturing. Humans communicate using body language and gestures in addition to other linguistic modalities like prosody and text or dialog structure. This research aims to develop a pointing gesture detection system for robot to robot communication scenarios to grant robots an ability to convey object identity information without global localization of the agents. The detection is based on RGB-D and a NAO humanoid robot is used as the pointing agent in the experiments. The presented algorithms are based on PCL library. The results indicate that real-time detection of pointing gesture can be performed with little information about the embodiment of the pointing agent and that an observing agent can use the gesture detection to perform actions on the pointed targets.**

I. INTRODUCTION

Humans are species that gesture [1]. In human communication, gesturing is closely associated to verbal communication and complements spoken language by providing additional information or emphasizing specific meanings and descriptions. Furthermore, deictic gestures (pointing gestures) can help recipients to identify the target object of a conversation by guiding the recipient's gaze in the target's region.

Deictic gestures can thus be used to obtain an agreement on the symbols used in the communication between the interacting parties. In this research, we propose to use gestures to obtain the agreement on names of physical objects between two robots. In other words, gesturing is used to cooperatively anchor names (symbols) to physical objects in a heterogeneous multi-robot system. Such anchoring is necessary for the agents to be able to collaborate on solving tasks.

Therefore, this paper presents a pointing gesture detection system for robot to robot communication scenarios to grant robots an ability to convey object identity information. The detection is based on RGB-D data and analysis of the resultant point cloud. The approach offers the distinctive benefit that the communicating robots do not require a synchronized coordinate frame in order to pinpoint an object of interest in space (transmitting absolute coordinates between them). Each robot can carry its own local coordinate frame and localize objects of interest through a pointing gesture. Additionally, robots equipped with more human-like communication capabilities can more naturally blend in human environments and houses. Most of people are not accustomed with robotic devices and thus robots exhibiting familiar human-like communication behaviors could more easily blend in household environments. According to our best knowledge, this is the first work proposing to understand deictic gestures between robotic agents. The particular contribution of this work is: a) the idea of using deictic gestures for anchoring object identities, b) a description of a system for understanding robot-robot deictic gestures and c) an experimental study of the capabilities of such a system.

The paper is organized as follows: We begin by surveying the related literature in Section II. In Section III the scenario including the hardware platforms is introduced. Section IV describes the pointing gesture detection and tracking algorithm. In Section V the object extraction and segmentation algorithm is explained. Section VI illustrates the fusion of the developed algorithms and the final extraction of the object. Section VII presents real world experiments which support the effectiveness of the proposed system. Finally a conclusion is given in Section VIII.

II. RELATED WORK

There is currently many on-going research endeavors on detecting and recognizing human gestures based on data from for example visual sensors. In human-machine interaction, gesturing has been utilized as a natural interface for human operators to control complex devices such as TV sets or games consoles. In robotics, gestures have been exploited as an intuitive human-robot command interface, where the human controller directs and communicates actions to its robot counterpart through gesturing.

However, most methods geared towards human-machine interaction scenarios rely on ready-made skeletal and hand tracking libraries provided for example by OpenNI and Mirosoft SDK, which are designed to detect and track only human shapes and heights. So far and to the extent of our knowledge, there is very limited research on robot to robot interaction through body language. An example can be found in [2] where the authors present their work on language evolution using gestures in a robot to robot or human to robot scenarios. They also discuss the applicability of this framework on grounding communication modalities between robots. However, their research mainly concerns the linguistic aspects of the developing communication system that includes both voice and gestures and not the usability of a human-like gesturing system. Furthermore, according to Chapter 5 of the book [2], the authors implement very limited visual strategies to detect gestures. More specifically, deictic gestures are not explicitly detected between participating robots, in a language game experiment, but the coordinates of the pointed objects are broadcasted by the pointing robot.

*Resrach is supported by EU FP7 grant RECONFIG (no. IST-600825).

P. Kondaxakis (corresponding author), J. Pajarinen and V. Kyrki are with the Intelligent Robotics Group, Electrical Engineering and Automation Department, Aalto University, Finland. (E-mail: {polychronis.kondaxakis, joni.pajarinen, ville.kyrki}@aalto.fi).

Despite the differences between human-robot and robot-robot gesturing, we next provide an overview of the works in human gesture detection. Gesture detection systems deploy a variety of visual sensors and algorithms to recognize hand moves and track their trajectories. For example, RGB-D sensors are widely used for this purpose. In [3], the authors recognize pointing gestures using Kinect based depth image and skeletal points tracking. To detect the direction of the pointing gesture, they detect and track the pointing fingertip using a minimum bounding rectangle and a Kalman filter estimator. In a similar trail of thought, the authors in [4] use the skeletal tracker provided by the Kinect SDK and a data mining classifier to recognize human gestures. Simpler sensor setups such as a simple 2D camera have been also deployed to detect human gesturing as described in [5], [6] and [7], where probabilistic and other classifier methods were used to categorize a number of gestures. Another 3D method (PinPoint system) is described in [8] and it utilizes a pair of calibrated cameras to achieve stereo vision and detect pointing gestures. Finally, a 3D time of flight (TOF) camera is utilized in [9] to detect and categorize pointing gestures based on the pointing direction for a slideshow presentation scenario.

In the more related human-robot interaction research, there are also plenty of approaches that utilize RGB-sensors to detect gestures. For example, both approaches in [10] and [11] use 3D Kinect sensor and OpenNI and NITE skeletal tracker to recognize a number of pointing and other gestures. OpenNI and NITE libraries [12] as well as Microsoft Kinect SDK [13] provide a simple ready-made skeletal tracking solution for human detection [14] and joint tracking and thus they are widely used in human-robot interaction scenarios and in gesture detection approaches. Another method that uses 3D Kinect mounted on a mobile robot is presented in [15], where the authors use the available OpenNI hand tracker to initiate pointing gesture detection. Having that natural interface, a human operator can direct the mobile robot to specific locations in space by simply gesturing. However, there are other available visual sensor setups such as in [16] and [17] where the authors of both publications use stereo vision to recognize gestures and have similar algorithmic implementation methods. On one hand, in [16] the authors use 3D particle filtering for tracking and a cascade of two hidden Markov model (HMM) for pointing direction estimation. On the other hand, in [17] a multi hypothesis tracking framework is established to find the positions in space of the respective body parts and an HMM-based classifier is trained to detect pointing gestures. Finally, there are simple 2D methods utilizing a single color camera to control robotic artifacts such as in [18], [19] and [20]. To detect deictic gestures, the authors in [18] apply a model that enables a humanoid robot to recognize the static orientation of the gesture as an edge image and movement as an optical flow. These gestures are used to influence a learning mechanism on the robot that enables to further understand its environment. In [19] deictic gestures are recognized through extending a trajectory recognition algorithm based on particle filtering with symbolic information from the objects in the vicinity of the acting hand. Finally, in [20] counting gestures are detected by a swarm of mobile robots to play games. Each robot is equipped with a statistical classifier, which is used to generate an opinion for the sensed gesture.

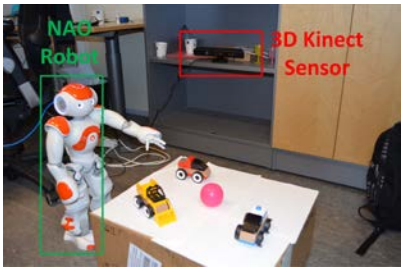

Figure 1. The NAO robot is pointing to an object of interest, in this case the red wooden toy-car and a 3D Kinect device is observing the scene.

III. System Settings

This research is conducted on a scenario where deictic gestures from a humanoid platform (in this case the Aldebaran's NAO robot) will be detected by a Kinect 3D sensor device (Figure 1). On one hand, the Kinect device is composed by a VGA color camera and a depth sensor. Depth measurements are extracted utilizing an infrared projector and a monochrome CMOS sensor that work together to "see" the room in 3D regardless of the lighting conditions. The default RGB video stream uses 8-bit VGA resolution (640×480 pixels) at 30FPS with a Bayer color filter [21]. On the other hand, NAO robot is a 25-degrees of freedom humanoid platform equipped with a colored camera sensor. The robot can perform all kinds of deictic gesture moves and can detect and home to any objects of interest in its environment. In our robot to robot interaction scenario, the Kinect sensor represents the observing robot. Moreover, pointing and observing robots are synchronized when a deictic gesture starts, to avoid potential outliers from other moving objects in the scene. Some basic assumptions made in this research are: a) The hand detection algorithm relies on movement and thus any moving object in Kinect field of view is a potential pointing arm, b) The size of the pointing arm has to be known a priori, c) Pointing and observing agents should remain stationary during gesture execution. Later-on in the Experimental Results section, the Kinect sensor will be augmented by a manipulator arm, which will demonstrate some dynamic grasping on the pointed objects. All developed algorithms have been tested on an Intel® Core™ i5-3320M CPU @ 2.60GHz×4 with 8GB of RAM memory. They are based on Point Cloud Library (PCL) [22] and divided into three distinctive modules under the ROS framework (ROS-nodes). Figure 2 presents the interactions between the three nodes and how they are connected to each other. In more details, the Hand Detection and Object Detection ROS nodes subscribe to the ROS Kinect driver Node and obtain RGB-D data. Secondly, the Find Pointed Object ROS node subscribes to the Hand Detection and Object Detection ROS nodes, which provide the pointing vector and the point clusters of the segmented objects respectively. Finally, a resulting point-cluster of the pointed object is published by the Find Pointed Object Action ROS node.

IV. Arm Detection Component

This section describes the implementation of the arm

detection and tracking node. This node subscribes to the ROS Kinect node, which provides the RGB-D point cloud data as obtained by the Kinect sensor. After processing these data, the node publishes the detected pointing vector as two distinctive points that represent the beginning and the end of the vector. Upon detection of a deictic gesture, the pointing vector is constantly provided to the subscribed modules even if the pointing arm remains stationary. The algorithm is divided into three parts. The first part segments the point cloud to extract moving points, that is, areas with spatial changes in the environment. The second part assigns a bounding box according to the dimensions of the arm that needs to be detected. The third part extracts the pointing vector. Figure 3A below visualizes the aligned bounding box and the corresponding pointing vector on an unsegmented point cloud obtained by the Kinect device. In the current setup this algorithm performs at ≈3.3Hz update rate. Therefore, although it detects the pointing arm independently of its moving speed, there are speed restrictions on NAO's pointing action. This is due to the detection algorithm's relatively low update frequency. However, it is extremely robust on slow pointing arm rotation rates.

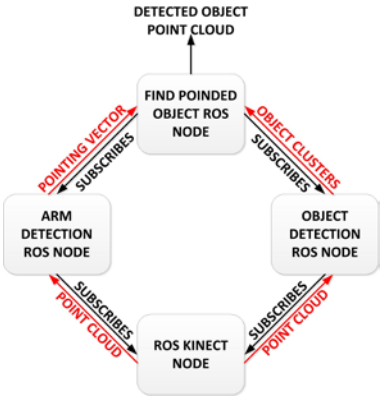

Figure 2. Pointing gesture detection algorithm developed under the ROS framework.

A. *Spatial Change Detection*

The first step is to detect spatial changes between multiple consecutive unorganized point clouds that could vary in size, resolution, density and point ordering. To achieve real-time implementation, we first reduce the number of input points by downsampling the raw point cloud data. This is achieved by utilizing a VoxelGrid filter from PCL. This filter approximates all available points in each 3D cell of the created voxel-grid with their centroid. Next, we create an octree data structure for organizing the sparse 3D downsampled data. Spatial changes are identified by recursively comparing the tree structures of the octrees and locating differences in voxel configuration. Then the algorithm processes only these differences between two consecutive point clouds by comparing all octree's leaf nodes that did not exist in previous buffer. Finally, we further remove outlier data by utilizing a *RadiusOutlier* filter from PCL. This filter removes all points in its input cloud that do not have a minimum number of neighbors within a certain range. The remaining point data are further segmented into clusters using the Euclidean Cluster Extraction method from PCL. This method is described in [23] and it uses the filtered point cloud (from the *RadiusOutlier* filter) and a Kd-tree structure for finding nearest neighbors of every point in the cloud. For each point, the neighboring points are evaluated according to their Euclidean distance. Points positioned inside a sphere with a particular radius, with distance values below a certain threshold, are collected into the same cluster.

B. *Bounding Box*

The second step of the proposed algorithm is to initially assign a 3D rectangular bounding box to the most appropriate cluster extracted by the Euclidean segmentation algorithm and then calculate the relative pointing vector. The bounding box technique provides a simple but yet efficient solution for real-time arm detection and it simultaneously avoids any heavy processing model registration methods. It also provides a level of modularity and re-configurability in case different arm shapes and sizes need to be detected. More specifically, we manually assign threshold values for the *max* and *min* side sizes of a 3D rectangular box that represents the dimensions of the arm we want to detect, and we compute the best fit to the available clusters based on the following criteria: $\mathbf{dim}_{min} < \mathbf{dim}_i < \mathbf{dim}_{max}$, where $\mathbf{dim}$ is the dimensional vector $(width, hight, length)$ and i is the number of a particular cluster. Here, NAO robot uses its fully extended arm to point at objects. The *min* and *max* arm dimensions are modeled as (0.02m, 0.06m, 0.2m) and (0.08m, 0.09m, 0.3m) respectively.

To assign a bounding box in each available cluster we execute the following steps:

1. Compute the centroid of each available cluster $\mathbf{x}_{cent_i} = \left(x_{cent_i}, y_{cent_i}, z_{cent_i}\right)$.
2. Determine orientation by calculating the eigenvectors of the covariance matrix of the cluster points so that the eigenvectors $\mathbf{eigDx} = (\bar{\boldsymbol{e}}_1, \bar{\boldsymbol{e}}_2, \bar{\boldsymbol{e}}_3)$ are used to obtain the rotational matrix ${}^{w}\mathbf{R}_l = (\bar{\boldsymbol{e}}_1 | \bar{\boldsymbol{e}}_2 | \bar{\boldsymbol{e}}_1 \times \bar{\boldsymbol{e}}_2)$, ($w$ = world frame, l = local frame) with $\bar{\boldsymbol{e}}_1 \times \bar{\boldsymbol{e}}_2 = \pm \bar{\boldsymbol{e}}_3$.
3. Transfer the points of each cluster i to the local frame of that cluster by:

$$ {}^{l}\mathbf{P}_i = {}^{w}\mathbf{R}_l^T \left({}^{w}\mathbf{P}_i - \mathbf{x}_{cent_i} \right) \quad (1) $$

4. Compute the max, the min and the center of the diagonal points.

$$ \mathbf{x}_{mean\_diag_i} = 0.5 \times \left(\mathbf{x}_{max_i} + \mathbf{x}_{min_i} \right) \quad (2) $$

5. Given a box centered at the origin with size $\left(x_{max_i} - x_{min_i},\ y_{max_i} - y_{min_i},\ z_{max_i} - z_{min_i}\right)$ we apply the following transformation to the box center coordinates to place it in the right position and orientation in space:

<u>*Translation*</u>

$$ \mathbf{x}_{box\_center} = {}^{w}\mathbf{R}_l \mathbf{x}_{mean\_diag_i} + \mathbf{x}_{cent_i} \quad (3) $$

<u>*Rotation*</u>

$$ {}^{w}\mathbf{R}_l = (\bar{\boldsymbol{e}}_1 | \bar{\boldsymbol{e}}_2 | \bar{\boldsymbol{e}}_1 \times \bar{\boldsymbol{e}}_2) \quad (4) $$

C. Pointing Vector Extraction

Having isolated NAO's pointing arm by using the bounding box technique, we next extract the pointing vector. To achieve this, we first calculate the center points of the two far sides of the rectangular bounding box. These two sides are located one near the detected arm's fingertips and the other near the shoulder of the robot as shown in Figure 3A. The vector is defined by these two center points. To obtain the pointing direction, we take into account two vectors from two consecutive time instants (t and t-1) when the robot starts pointing at something. Next, we project these two lines into a 2D plane by ignoring the z-axis direction and we find their 2D intersection. Finally, we compare the 2D Euclidean distance between this point and the two center points defining the vector line at time t. The center point with the smaller distance from the intersection point is defined as the back point of the vector and the other as the front vector point. Therefore, the arm is pointing towards the front center point direction.

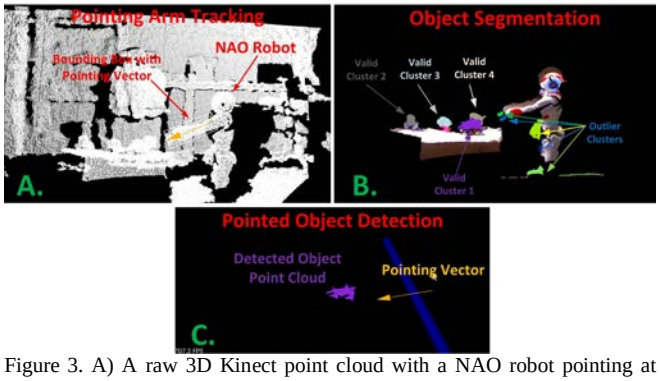

Figure 3. A) A raw 3D Kinect point cloud with a NAO robot pointing at objects. NAO pointing arm is detected by fitting a bounding box to it and the pointing vector is extracted. B) The "Object Detection" ROS node segments the raw point cloud data from the Kinect sensor and outputs point cloud clusters of potential objects. C) The red wooden toy-car is detected by the merging "Find Pointed Object" component.

V. Object Detection Component

This ROS node is responsible for segmenting the available point cloud and creating clusters of points that represent objects. It subscribes to the ROS Kinect node to obtain the raw point cloud data and it publishes the extracted point cloud clusters for later use.

The developed algorithm is also based on the PCL library. Firstly, we compute normals for an organized point cloud using integral images. To achieve this we utilize a covariance matrix method, which creates nine integral images to compute the normal for a specific point from the covariance matrix of its local neighborhood. Following that, the obtained normals are inserted into the *organized multiplane segmentation* functionality of PCL to segment the raw point cloud data into clusters. This algorithm finds all planes present in the input cloud, and outputs the plane equations, as well as point clouds corresponding to the inliers of each detected plane. It uses the 2D image information to detect segment boundaries and a flood fill technique to connect pixels. Following a common approach, we first segment planes from the point cloud, then mask the segmented planes, and then employ an Euclidean point similarity measure to extract objects. It is worth mentioning here that the *organized multiplane segmentation* algorithm used in this work, operates efficiently in real time, making it ideal for the developed system at hand. In the current setup, the segmentation algorithm runs at ≈0.95Hz. Figure 3B visualizes the output of the Object Detection ROS node that has detected a number of point cloud clusters marked with different colors. Furthermore, a number of outlier clusters have been detected, which will be removed later on the Find Pointed Object integration component.

VI. Find Pointed Object Component

For extracting the pointed object information, a third ROS node has been built to manipulate and combine the output from the two previous nodes. As Figure 2 demonstrates, this ROS node subscribes to both Hand Detection node and Object Detection node and then receives the pointing vector information and the detected object point-cloud clusters respectively. The implemented algorithm in this node, firstly selects the point-cloud cluster with the shortest vertical projection distance of its centroid to the pointing vector line. Next, it further filters out possible outliers (wrong clusters) by utilizing a 3D grid map technique.

A. Object Cluster Selection

The criteria to select the most relevant point-cloud cluster from the available ones, provided by the Object Detection ROS node, is the shortest vertical projection distance of its centroid to the pointing vector line. Assuming that we have the front and back pointing-vector points ($\mathbf{x}_f$ and $\mathbf{x}_b$) as well as the centroid points of each cluster i, this distance is calculated using the following equation:

$$d = \sqrt{\frac{\left|\mathbf{x}_f - \mathbf{x}_{cent_i}\right|^2 \left|\mathbf{x}_b - \mathbf{x}_f\right|^2 - \left[\left(\mathbf{x}_f - \mathbf{x}_{cent_i}\right)\cdot\left(\mathbf{x}_b - \mathbf{x}_f\right)\right]^2}{\left|\mathbf{x}_b - \mathbf{x}_f\right|^2}} \quad (5)$$

Also, the sign of the dot product $\left(\mathbf{x}_f - \mathbf{x}_b\right)\cdot\left(\mathbf{x}_{cent_i} - \mathbf{x}_f\right)$ provides information regarding the projected location of the centroid to the pointing vector. If the sign is positive, the projection lies after the front vector point and all relative clusters will be evaluated as possible pointed objects. On the other hand, if the sign is negative all objects represented by detected clusters will be disregarded as outliers.

B. 3D Grid Map Filtering

To obtain a consistent object selection mechanism, we implement a time-delay filter to remove any further outliers that might occur. These outliers are caused by small variations on the computed vertical projection distances among consecutive algorithmic iterations. The proposed technique is inspired by authors' previous work in [23]. It can be understood as a 3D grid map where a counter in each cell totals possible object centroid hits. In each algorithm iteration, the object's centroid point is associated with a particular cell in the grid-map and a cell counter increases linearly until it reaches a maximum value (7 in this case). The general rule here is that the dominant pointed object is

the one whose centroid falls inside a grid cell with the most counts in it. Also, for every iteration a linear forgetting factor decreases the counts inside the cells that have not been updated with a new centroid point. If at a particular time instant an outlier object appears, with its centroid point falling inside a cell with a lesser count-value, the algorithm takes into account the last centroid-verified object point cloud. Therefore, the object appearing most is assumed to be the final pointed object of interest.

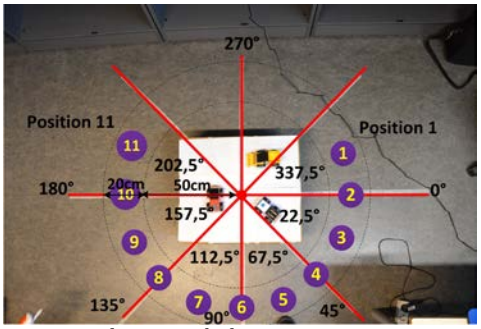

Figure 4. Experimental setup with distinctive pointing positions for NAO robot. The space is divided clockwise into eleven segments (1 to 11) starting from 0° to 337.5°.

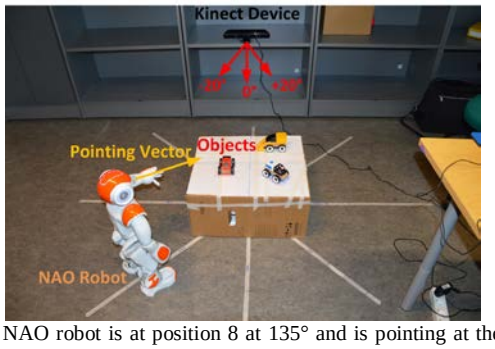

Figure 5. NAO robot is at position 8 at 135° and is pointing at the red car object. The Kinect device observes the objects at 0°.

VII. Experiments and Results

To evaluate the performance of the proposed algorithm we completed a series of real world experiments. Here, the NAO robot is placed on a number of predefined positions and performs a pointing gesture at three objects (IKEA wooden toy cars). These objects are placed in fixed locations on top of a cardboard box serving as a table. A Kinect sensor is at a fixed position in space, observing the box with the objects at 1m distance and at 270° according to Figure 4. Eleven discrete pointing positions have been defined around the box at approximately 60 cm distance from the center of the box (purple spots at Figure 4). In each position, NAO executes a separate pointing gesture for each object. For example, in Figure 5 NAO is located at position 8 (135°) and it points towards the red wooden car. Moreover, during this experiment the Kinect sensor was facing the paper box with the objects on top from three different locations. For positions 1 to 3 the Kinect was placed at +20°, for positions 4 to 8 at 0° and for positions 9 to 11 it had a -20° orientation (Figure 5). These orientations provided the Kinect device with the proper field of view to observe the NAO pointing gestures at all positions.

The results of the experiment are illustrated in Table I below. In each spot, NAO performs a pointing gesture ten times for each object. This pointing gesture repeatability assesses the robustness of the algorithm. According to the results in this table, the proposed system is accurately recognizing most of the pointed objects from all predefined positions. However, there are position-object combinations that the current implementation could not detect. These outliers occur whenever there are occlusions of objects and more specifically they appear at positions 1 (red car), 3 (red car), 5 to 9 (yellow car), 10 (blue car). Here, we define that an object is occluded when there is another blocking object before, at the same pointing vector direction. For example, in position 3 we say that the red car is occluded by the blue car. At this position, the system assumes that the blue car is the pointed object, because the arm of the pointing robot is almost horizontal. Although such situations are extremely difficult to overcome, some improvements in: a) the accuracy of the pointing arm detection mechanism, b) the pointing precision of robot's arm, c) dynamic change in pointing position to avoid occlusions, could provide adequate solutions to the problem.

TABLE I. NAO Pointing Experimental Results

Kinect Orientation	NAO Position	Objects	Arm	Success Rate	Percentage
+20°	1 (337,5°)	Red	Left	0/10	0%
		Blue	Left	10/10	100%
		Yellow	Right	9/10	90%
	2 (0°)	Red	Left	7/10	70%
		Blue	Left	10/10	100%
		Yellow	Right	10/10	100%
	3 (22,5°)	Red	Left	0/10	0%
		Blue	Left	10/10	100%
		Yellow	Right	10/10	100%
0°	4 (45°)	Red	Left	10/10	100%
		Blue	Right	10/10	100%
		Yellow	Right	10/10	100%
	5 (67,5°)	Red	Left	10/10	100%
		Blue	Right	10/10	100%
		Yellow	Right	0/10	0%
	6 (90°)	Red	Right	10/10	100%
		Blue	Left	10/10	100%
		Yellow	Left	0/10	0%
	7 (112,5°)	Red	Left	10/10	100%
		Blue	Right	9/10	90%
		Yellow	Left	0/10	0%
	8 (135°)	Red	Left	10/10	100%
		Blue	Left	10/10	100%
		Yellow	Left	1/10	10%
-20°	9 (157,5°)	Red	Left	10/10	100%
		Blue	Right	9/10	90%
		Yellow	Left	0/10	0%
	10 (180°)	Red	Left	8/10	80%
		Blue	Right	0/10	0%
		Yellow	Left	8/10	80%
	11 (202,5°)	Red	Left	10/10	100%
		Blue	Left	9/10	90%
		Yellow	Left	10/10	100%
		Overall Succes		240/330	≈73%

According to Table I the overall success rate of the experiment is approximately 73%, which indicates that the developed system operates adequately.

In a final demonstration of the algorithm's functionality, we implemented an interactive scenario where a Kinova JACO arm was used to grasp and transfer the pointed object. The relative pose between the arm and Kinect was calibrated. To control the JACO arm a separate ROS node is implemented. This module subscribes to the Find Pointed Object ROS component and obtains the centroid of the detected pointed object. Upon receiving the centroid, the

arm aligns its gripper to this point and performs a predefined pick up and place sequence. The demonstration is shown on the accompanying video. It provides a practical proof that the proposed system offers a future framework on multi robot interaction scenarios. In such scenarios gestures and active body language between robots will play a key role in the manipulation of their environment.

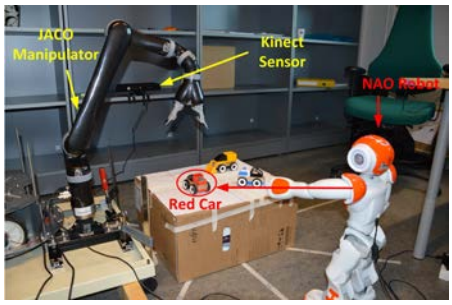

Figure 8. Robot to robot interaction scenario where NAO points at red wooden car and Kinova's JACO arm grasps and relocates the object.

VIII. CONCLUSIONS

This paper presents the idea that deictic gesturing can be used as a mechanism for transferring identity and class information of objects between robots. The idea is demonstrated by developing a real-time pointing gesture detection system as well as a pointed object segmentation and recognition mechanism. The algorithm utilizes a point cloud from an RGB-D sensor.

From the experimental results we conclude that the algorithm is effectively detecting pointed objects from the humanoid robot NAO. It has detected most of the objects correctly with a success rate of approximately 73%.

In the future, we are planning to implement a more sophisticated probabilistic approach to overcome occlusions. In this forthcoming framework, probabilities will be assigned at the pointed objects and they will be updated in a recursive manner according to measured features such as vertical distance from pointing vector, distance from pointing robot, pointed objects surface area, etc. Furthermore, a future implementation of a prediction-estimation technique such as Kalman filters or Particle filters could increase the accuracy of the current pointing arm detection algorithm and improve overall system robustness.

REFERENCES

[1] M. M. Louwerse and A. Bangerter, "Focusing Attention with Deictic Gestures and Linguistic Explessions" In *The Annual Conference of Cognitive Science Society*, Stresa, Italy, July 21 – 23: Erlbaum, 2005, pp. 1331 -1336.

[2] L. Steels and M. Hild, *Language Grounding in Robots*, New York: Springer, 2012.

[3] P. Jing and G. Ye-peng, "Human-computer Interaction using Pointing Gesture based on an Adaptive Virtual Touch Screen" In *International Journal of Signal Processing, Image Processing and Pattern Recognition*, Volume 6, Issue 4, August, 2013, pp. 81-91.

[4] O. Patsadu, C. Nukoolkit and B. Watanapa, "Human Gesture Recognition Using Kinect Camera" In *2012 Ninth International Joint Conference on Computer Science and Software Engineering (JCSSE12)*, Bangkok, Tailand, May 30 -June 1, 2012, pp. 28 – 32.

[5] X. Zabulis, H. Baltzakis, A. Argyros, "Vision-based Hand Gesture Recognition for Human-Computer Interaction", In *The Universal Access Handbook*, Human Factors and Ergonomics. Lawrence Erlbaum Associates, Inc. (LEA), 2009.

[6] M. Sigalas, H. Baltzakis and P. Trahanias, "Gesture recognition based on arm tracking for human-robot interaction", In *The IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS10)*, Taipei, Taiwan, October 18-22, 2010, pp. 5424 – 5429.

[7] T. Axenbeck, M. Bennewitz, S. Behnke and W. Burgard, "Recognizing Complex, Parameterized Gestures from Monocular Image Sequences", In *8th IEEE/RAS International Conference on Humanoid Robots, Humanoids 2008*, Daejeon, Korea, December 1-3, 2008, pp. 687 – 692.

[8] P. Matikainen, P. Pillai, L. Mummert, R. Sukthankar, "Prop-Free Pointing Detection in Dynamic Cluttered Environments", In *IEEE International Conference on Automatic Face and Gesture Recognition*, Santa Barbara, CA, USA, March 21 - 25, 2011, pp. 374-381.

[9] M. Haker, M. Böhme, T. Martinetz and E. Barth, "Deictic gestures with a time-of-flight camera", In *GW09 Proceedings of the 8th International Workshop on Gesture in Embodied Communication and Human-Computer Interaction*, Bielefeld, Germany, February 25-27, 2009, Volume 5934, pp. 110-121.

[10] C. P. Quintero, R. T. Fomena, A. Shademan, N. Woolleb, T. Dick and M. Jagersand, "SEPO: Selecting by Pointing as an Intuitive Human-Robot Command Interface", In *IEEE International Conference on Robotics and Automation (ICRA13)*, Karlsruhe, Germany, May 6 – 10, 2013, pp. 1158 – 1163.

[11] N. S. Mohd Nor, Y. Maeda and M. Mizukawa, "Pointing Angle Estimation for human – Robot Interface", In *Journal of Advances in Computer Networks*, Volume 1, Issue 2, June, 2013, pp. 75 – 78.

[12] OpenNI NITE 2.2, URL: http://www.openni.org/files/nite, 2013.

[13] Microsoft Kinect SDK 1.8, URL: http://www.microsoft.com/en-us/kinectforwindowsdev/start.aspx, 2013.

[14] J. Shotton, A. W. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, Richard Moore, Alex Kipman, Andrew Blake, "Real-Time Pose Recognition in Parts from Single Depth Images" In *IEEE Computer Vision and Pattern Recognition (CVPR11)*, Colorado Springs, USA, June 21-23, 2011, pp. 1297 – 1304.

[15] M. Van den Bergh, D. Carton, R. De Nijs, N. Mitsou, C. Landsiedel, K. Kuehnlenz, D. Wollherr, L. V. Gool and M. Buss, "Real-time 3D Hand Gesture Interaction with a Robot for Understanding Directions from Humans", In *20th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, Atlanta, Georgia, USA, 31 July - 3 August, 2011, pp. 357 -362.

[16] C. B. Park and S. W. Lee, "Real-time 3D pointing gesture recognition for mobile robots with cascade HMM and particle filter", In *Journal of Image and Vision Computing*, Volume 29, Issue 1, January, 2011, pp. 51–63.

[17] K. Nickel and R. Stiefelhagen, "Visual Recognition of Pointing Gestures for Human-Robot Interaction", In *Journal of Image and Vision Computing*, Volume 25, Issue 12, December, 2007, pp. 1875–1884.

[18] Y. Nagai, "Learning to Comprehend Deictic Gestures in Robots and Human Infants" In *Proceedings of the 14th International Workshop on Robot and Human Interactive Communication (RO-MAN)*, Nashville, Tennessee, USA, 13-15 August, 2005, pp. 217-222.

[19] N. Hofemann, J. Fritsch and G. Segerer, "Recognition of Deictic Gestures with Context", In *proceedings of Pattern Recognition, 26th DAGM Symposium*, Tübingen, Germany, August 30 - September 1, 2004, pp. 334 -341.

[20] A. Giusti, J. Nagi, L. Gambardella and G. A. Di Caro, "Cooperative Sensing and Recognition by a Swarm of Mobile Robots", In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS12)*, Vilamoura, Algarve, Portugal, October 7-12, 2012, pp. 551 – 558.

[21] URL: http://en.wikipedia.org/wiki/Kinect.

[22] Point Cloud Library (PCL 1.7), URL: http://pointclouds.org/, 2013.

[23] R. B. Rusu, "Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments", *PhD dissertation*, Computer Science department, Technische Universitaet Muenchen, Germany, October, 2009.

[24] P. Kondaxakis and H. Baltzakis, "Multiple-Target Classification and Tracking for Mobile Robots Using a 2D Laser Range Scanner", In *International Journal of Humanoid Robotics*, Volume 9, Issue 3, August, 2012, pp. 1250025-1 - 1250025-23.