APPEARANCE-BASED COGNITION OF OBJECTS POINTED OUT BY HUMAN HANDS

by

Mirhan Ürkmez

B.S., Electrical and Electronics Engineering, Boğaziçi University, 2019

B.S., Mathematics, Boğaziçi University, 2019

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electrical and Electronics Engineering

Boğaziçi University

2021

ACKNOWLEDGEMENTS

First of all, I would like to thank Prof. H. Işıl Bozma for her guidance during the thesis. She has always had a solution to the problems which helped me get through a lot of trouble. I am grateful that she always had a positive attitude and supported me.

I want to thank Prof. Yağmur Denizhan and Assoc. Prof. Erol Şahin for attending my thesis committee.

I would like to thank ISL members for their support and collaboration throughout the thesis. Both Meriç Durukan and Serhat İşcan contributed to my studies. I appreciate Mehmet Yasin Özkan's great efforts to annotate pointing direction data set.

This work was done with partial support of TÜBİTAK EEEAG Project #118E857.

Last but not least, I have to mention all my friends and my family. They listened to my complaints patiently.

ABSTRACT

APPEARANCE-BASED COGNITION OF OBJECTS POINTED OUT BY HUMAN HANDS

In this thesis, human-guided appearance-based object cognition in robots is addressed. Here, the robot observes the human pointing to the object of interest based on the incoming RGB-D data and then either recognizes or learns this object as needed. This problem is of interest because the associated learning problem does not require objects and their labels to be provided externally as is the case with supervised learning or learned objects can be more human-intuitive since the robot is not completely on its own as is the case with unsupervised learning. We propose a complete end-to-end system consisting of three stages: First, the robot determines the pointing direction. For this, it first finds hands and humans in the incoming RGB image via exploiting a state-of-the-art CNN-based detector. Following, it finds the point cloud object corresponding to the hand segment through applying a density-based segmentation algorithm on the RGB-D data and then estimates the 3D pointing direction vector from the implicit geometry of the 3D hand segment. We also introduce a RGB-D data set with varied robot-human distances and pointing gesture directions - due to the unavailability of such a data set. In the second stage, the robot determines the targeted object based on the 3D pointing direction. For this, it determines a set of candidate point cloud objects and then selects the object that is most likely to be targeted. The final stage is either to recognize the target object or to learn it as necessary. In this, its objects' memory that is organized hierarchically plays a key role. In the latter case, the new object class is added to the memory using an unsupervised learning algorithm. To the best of our knowledge, the proposed system is the first end-to-end system in which the robot's reasoning is completely autonomous. The advantages of the proposed approach are as follows: i) Applicability in a wide-range of robot-human interactions regardless of human proximity and background variability; ii) Ability to continue learning new object classes through interaction with humans.

ÖZET

ROBOTLARDA İNSANLARIN ELLE İŞARET EDİLEREK GÖSTERİLEN NESNELERİN GÖRSEL TEMELLİ BİLİŞİ

Bu tezde, bir robotun insanlarla onların nesneleri elle göstermeleri suretiyle etkileşime girerek, renk ve derinlik (RGB-D) verilerini kullanarak gösterilen nesneyi tanımaları veya öğrenmeleri konusunda çalışılmıştır. Bu problemin ilginç olmasının sebebi, tam güdümlü öğrenmeden farklı olarak görünüşlerin etiketlenmelerine gerek olmaması, ancak güdümsüz öğrenmede olduğu gibi de robotun nesnelere ait muhakemesinde tamamiyle tek başına olmaması nedeniyle nesne bilişinin daha insan-benzeri olabilmesidir. Önerilen yaklaşım üç ana aşamadan oluşmaktadır. İlk olarak, robot insanın işaret ettiği yönü belirlemeye çalışır. Bunun için robot evrişimsel sinir ağı temelli bir yaklaşımla RGB verisinden el ve insanın yerlerini ve şekillerini belirler. Ardından, bu bilgiyi derinlik verisiyle tümleştirerek ve yoğunluk bazlı bir bölütleme algoritması uygulanarak üç-boyutlu el bölütünü bulur. Bu bölütün geometrik özellikleri kullanılarak üç boyutlu işaret yönü hesaplanır. Değerlendirma amaçlı kıyaslama veri setlerinin olmaması nedeniyle, değişken robot-insan uzaklıkları ve geniş işaret yönü çeşitliliği içeren bir RGB-D veri seti hazırlanmıştır. İkinci aşamada, robot işaret edilen nesneyi belirler. Bunun için, aday nesnelere ait bölütler belirlenir ve aralarından en büyük hedef puanına sahip olan işaret edilen nesne olarak seçilir. Hedef puanı, işaret yönüne ve nesne konumlarına bağlı olarak hesaplanır. En son aşamada, bu nesnenin tanınması ve tanınamadığı durumda ise öğrenilmesi yapılır. Burada, sıradüzensel bir mimariye sahip nesneler belleği önemli bir rol oynar. Tanınamayan nesneler, bir denetimsiz öğrenme yöntemiyle yeni nesneler olarak bu belleğe eklenir. Bildiğimiz kadarıyla, önerilen yaklaşım robotun karar vermesinin tamamen otonom olduğu ilk uçtan uca sistemdir. Önerilen yaklaşım iki ana avantaja sahiptir: i) İnsan uzaklığı ve arka plan değişiminden bağımsız olarak geniş bir yelpazede robot-insan etkileşimi senaryolarında uygulanabilir olması; ii) Nesneler bilişinin insanın işaret ettiği nesneleri kapsayacak şekilde sürekli gelişebilme yetisine imkan sağlaması.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . iii
ABSTRACT . . . iv
ÖZET . . . v
LIST OF FIGURES . . . viii
LIST OF TABLES . . . x
LIST OF SYMBOLS . . . xi
LIST OF ACRONYMS/ABBREVIATIONS . . . xiv
1. INTRODUCTION . . . 1
1.1. Contributions . . . 3
2. 3D POINTING DIRECTION ESTIMATION . . . 5
2.1. Introduction . . . 5
2.2. Related Literature . . . 7
2.2.1. Hand Detection . . . 7
2.2.2. Pointing Gesture Classification . . . 7
2.2.3. 3D Hand Pointing Direction Detection . . . 8
2.2.4. Hand Pointing Direction Data Sets . . . 9
2.3. Method . . . 11
2.3.1. Hand and Human Detection . . . 11
2.3.2. Pointing Gesture Classification . . . 12
2.3.3. 3D Segmentation . . . 14
2.3.4. Pointing Direction Estimation . . . 16
2.4. Experiments . . . 19
2.4.1. Hand Detection . . . 19
2.4.2. Pointing Gesture Detection . . . 20
2.4.3. 3D Pointing Direction Data Set . . . 21
2.4.4. Pointing Direction Detection . . . 22
3. TARGET OBJECT DETECTION . . . 24
3.1. Introduction . . . 24

3.2. Related Literature . . . 25
3.3. Method . . . 26
3.4. Experimental Results . . . 28
4. OBJECT LEARNING . . . 32
4.1. Introduction . . . 32
4.2. Related Literature . . . 33
4.3. Method . . . 34
4.3.1. Deformable Sphere Approximation Descriptor . . . 34
4.3.2. Deformed Sphere Mapping . . . 34
4.3.3. Recognition & Objects' Memory . . . 36
4.3.4. Object Learning . . . 36
4.4. Experimental Results . . . 37
4.4.1. Manually Switched Learning & Recognition Modes . . . 37
5. CONCLUSION . . . 42
REFERENCES . . . 44
APPENDIX A: User's Guide . . . 52
A.1. Hardware . . . 52
A.2. Required Software . . . 52
A.3. Project Software . . . 53
A.4. Usage . . . 54
APPENDIX B: Permissions . . . 55

LIST OF FIGURES

Figure 1.1. General Approach. 3

Figure 2.1. PDE Pipeline 6

Figure 2.2. Hand and Human Detection. 12

Figure 2.3. Architecture of Pointing Gesture Classifier. 13

Figure 2.4. Samples from Pointing Gesture Data set 14

Figure 2.5. 3D Hand Region. 14

Figure 2.6. Planes Passing Through Hand Orthogonal to v_1 and v_2. 17

Figure 2.7. Detected 3D Pointing Direction. 18

Figure 2.8. A Sample Detection with YOLOv4 19

Figure 2.9. Samples From Pointing Direction Data Set 22

Figure 2.10. A PDE Failure Case. 23

Figure 3.1. Orthogonal Distance Score Failure 25

Figure 3.2. Target Object Search Neighborhoods 26

Figure 3.3. θ_1 and θ_2 for the Ketchup Bottle 31

Figure 3.4. Sample Target Object Detections on MHRI Data 31

Figure 4.1. Some Objects Learned by Pointing Gestures. 38

Figure 4.2. Objects' Memory Hierarchy After Learning 5 Objects. 38

Figure 4.3. Under-segmentation of Blue Chair. 41

Figure A.1. Turtlebot Robot Used in the Experiments. 52

Figure B.1. MHRI data set permission 55

LIST OF TABLES

Table 2.1. PDE related literature 10

Table 2.2. Parameter table 18

Table 2.3. Average precision (AP). 20

Table 2.4. Computation time. 20

Table 2.5. Accuracy ratings for pointing gesture classification 21

Table 2.6. 3D Pointing direction data set 21

Table 2.7. Accuracy results for varying thresholds and average accuracy for 30°. . 23

Table 2.8. PDE results for different parameter values 23

Table 3.1. Results on MHRI data set. 30

Table 3.2. Comparative results on MHRI data set 30

Table 4.1. 3-fold cross validation average results 40

Table 4.2. Learning validation 40

Table 4.3. Recognition performance 41

Table 4.4. Test data results when pointed object is found correctly 41

LIST OF SYMBOLS

c	HSV value
$\mathcal{D}$	Point cloud set for segments
$\mathcal{D}_o$	Object cloud
$e_{h_1 h_2}$	The vector of an orthonormal set of trigonometric functions
H_1	The number of first order harmonics
H_2	The number of second order harmonics
$\mathcal{H}$	Hand point cloud data
$\mathcal{H}_1$	Neighborhood of $\perp v_1$ plane
$\mathcal{H}_2$	Neighborhood of $\perp v_2$ plane
$I_{coh_1 h_2}$	DSA descriptor of object o
J	Gesture prediction loss function
K	Number of gesture training samples
l	Index for a segment
L	Index set of labels
M_1	Minimum neighbors for dense points
N_1	Number of planes for $\perp v_1$
N_2	Number of planes for $\perp v_2$
$\mathcal{N}$	Neighboring points
o	A segment
o_{pd}	A hand point on pointing direction
$\mathcal{O}$	Set of 3D segments
p	A point in point cloud data
$\mathcal{P}$	Point cloud data
q	A point in spherical coordinate system
Q	Point cloud in spherical coordinates
s	Pointing Direction line parameter
v_0	Principal eigenvector of Σ
v_1	Eigenvector of Σ

v_2	Eigenvector of Σ
v_{pd}	Pointing Direction
w_{a1}	First angle threshold for objects
w_{a2}	Second angle threshold for objects
y_i	True gesture label
$\hat{y}_i$	Predicted gesture output
$z_{coh_1h_2}$	h1; h2 th coefficient of DSA descriptor
α	Hand-Object apex angle threshold
γ_1	Coefficient for pan angle threshold
γ_2	Coefficient for tilt angle threshold
δ_ϕ	Pan scan resolution
δ_ψ	Tilt scan resolution
δv_1	Plane separation distance for v_1
δv_2	Plane separation distance for v_2
θ	Hand-Object apex angle
λ_0	Principal eigenvalue of Σ
λ_1	Eigenvalue of Σ
λ_2	Eigenvalue of Σ
ρ	Depth value
ρ_{co}	Deformed sphere map
σ_1	Distance threshold for plane $\perp v_1$
σ_2	Distance threshold for plane $\perp v_2$
Σ	Covariance matrix of hand data
τ_ϕ	Normalization parameter for pan angle
τ_ψ	Normalization parameter for tilt angle
τ_ρ	Normalization parameter for depth value
τ_{p_1}	Strict threshold for depth value
τ_{p_2}	Loose threshold for depth value
τ_{hsv}	Threshold for HSV
ϕ	Pan angle

ψ	Tilt angle

LIST OF ACRONYMS/ABBREVIATIONS

2D	Two Dimensional
3D	Three Dimensional
AP	Average Precision
CNN	Convolutional Neural Network
DSA	Deformable Sphere Approximation
DTFS	Double Trigonometric Fourier Series
fps	frame per second
HRI	Human-robot Interaction
HSV	Hue Saturation Value
k-NN	k-nearest Neighbors
LBP	Local Binary Patterns
PDE	Pointing Direction Estimation
R-CNN	Region Based Convolutional Neural Networks
RGB	Red Green Blue
RGB-D	Red Green Blue Depth
ROI	Region of Interest
SSD	Single Shot Detector
SVM	Support Vector Machines
YOLOv4	You Only Look Once Version 4

1. INTRODUCTION

A large number of mobile robot applications requires the robots to be capable of recognizing the objects in their surroundings based on their appearances. For example, service robots may need to recognize objects such as home appliances or tools depending on their purpose of use. Due to the large number of objects and the variability of their appearances, object learning needs to be continual and incremental. The proposed approaches can be categorized to be either supervised or unsupervised. In the former, appearances need to be associated with labels. For example, most current state-of-art convolutional neural networks (CNNs) need large data sets for training. However, creating data sets for each object is quite expensive. Furthermore, the addition of new object classes to CNN object detectors cannot be done continually and incrementally. On the other hand, with unsupervised approaches, the robot is completely on its own. While the learning of objects can potentially be continual and incremental, the resulting object concepts need not to be 'human-like'.

In this thesis, a third approach which is in-between these two approaches is considered - namely human-guided appearance-based objects' learning through pointing interaction. In this problem, a human points to an object and the robot either recognizes the object or realizes that it need to learn it as a new object class. In the latter case, it then expects the human to show a small set of different views of the object which it then uses to learn the object class. Pointing is selected as our interaction method since it is known to be one of the most commonly used gestures for communication [1]. This approach is advantageous because each robot can be trained by its user for any specific goal so that the resulting object knowledge is human-like.

Object cognition through pointing interaction is a difficult problem since it consists of various components such as hand and human detection, pointing direction estimation, selection of targeted object, object recognition and object learning. The proposed approach consists of three stages as shown on Figure 1.1: i) Pointing direction estimation; ii) Target object determination and iii) Recognition or learning as needed.

The first stage aims to estimate the three-dimensional (3D) pointing direction. There has been extensive work on pointing direction estimation. However, many of them are not convenient for robot applications due to restrictions such as requiring the hand to be dominant object in the scene or outputting only 2D pointing directions. The proposed method aims to reliably determine the 3D pointing directions under wide-ranging scenarios. First, hand and human regions in the input RGB image are detected. This is a critical stage as reliable detection is essential to correct direction estimation and the robot being able to determine the target object. Fortunately, there are quite successful convolutional neural network (CNN) based object detectors that can be used for this task. The YOLOv4 along with the publicly available human and hand data sets are used to create the hand and human detector [2]. Next, the 3D hand segment is obtained through merging the RGB segment data and depth data and then using a density based segmentation algorithm. The intrinsic geometry of the pointing gesture is then exploited to find the pointing direction.

The second stage is to use the 3D pointing direction in order to find the targeted object. For this task, there are score based methods in the literature such that each object in the scene is assigned a score depending on pointing direction and the object with the best score is selected. A similar approach is also adopted here with hierarchical ordering. First, objects that are located within the close neighborhood of the pointing direction line are determined. This is followed by assigning a target score depending on their location with respect to the pointing direction. Finally, the object with the highest score is selected. If there are not any object close enough to the line, then the neighborhood is expanded and the whole process is repeated.

After finding the targeted object, the robot tries to classify this object based on its point cloud data. First, it encodes the respective point cloud data using deformed sphere approximation representation [3]. Then, it associates this descriptor with the knowledge existing in its objects' memory. The objects' memory has a tree organized structure in which terminal nodes correspond to distinct object classes. A hierarchical incremental clustering method is used to construct a tree structure. In this structure, each parent node is associated with a set of classifiers as to assign a new input to one of the child nodes.

The robot searches a top-down search in its place memory using these classifiers in order to determine whether it can recognize the target object through traversing down the hierarchy as long as classification performance is not sufficient or until either a terminal node is reached. The former case implies no recognition and the robot expects human to show the object from different viewpoints as to form its knowledge of this object. It then adds this to its existing objects' memory using incremental clustering method.

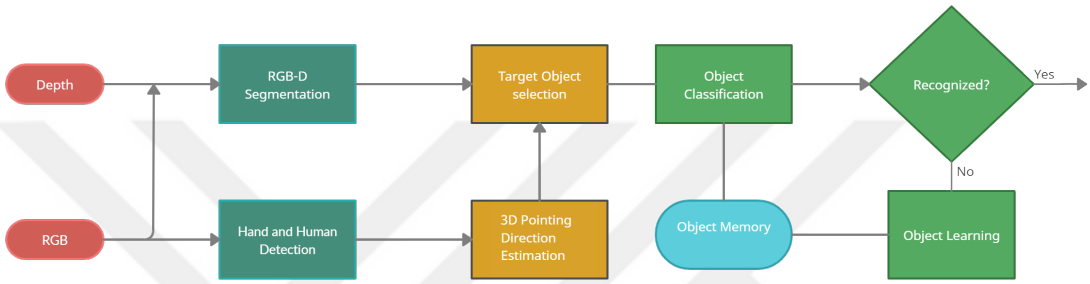

Figure 1.1. General Approach.

1.1. Contributions

The major contributions of the thesis are as follows:

- A new approach to the estimation of 3D pointing direction: Our proposed approach to the detection of 3D pointing direction from RGB-D images is suitable for a wide range of scenarios in which hand distance, background variability or the detectability of specific human parts all may vary. Our only assumption is that person performing pointing gesture should be in the range of depth sensor which is around 4.5 m according to sensor specifications. To the best of our knowledge, this is the first framework than can accurately predict the 3D pointing direction from a single RGB-D image with the least amount of assumptions.
- A novel benchmark RGB-D data set with 3D pointing direction annotations: To the best of our knowledge there is no existing benchmark for the 3D pointing directions targeted in this work, so we have built a new data set that will also be publicly available.

This data set consists of RGB-D images of human with various pointing gestures, - incorporating challenges such as varying robot-human distance and large pointing gesture variability.

- A new approach to target object detection: The method is more complex than previous works and it is shown to perform well on a publicly available benchmark.
- A novel end-to-end approach to human-guided appearance-based objects' cognition in robots: The proposed method is completely autonomous and enables the robot to learn or recognize objects that are of interest for humans.

2. 3D POINTING DIRECTION ESTIMATION

2.1. Introduction

A pointing gesture typically specifies a direction from a person - indicating a location, a thing or another person. As such, it can be used to direct the robot's attention to a place or an object of interest. Thus, the ability to reliably and quickly detect the pointing direction is an important tool in the HRI repertoire. It requires the robot to detect human hand reliably and to estimate its pointing direction. This has proven to be a complex task as i) hand appearances vary in shape and orientation; and ii) human-robot interaction scenarios are wide-ranging with respect to hand proximity, background variability or the detectability of specific human parts.

There has been extensive work on this both within machine vision and robotics areas. Many work explore close-range interactions and thus assume the hand to be the dominant object in the incoming sensory data. However, in many HRI applications such as service robotics, such an assumption does not hold and the robot needs to estimate the pointing direction from a wide range of distances so that detecting the pointing direction becomes much harder. The problem is exacerbated by background variability that makes the detection of both hand and other body parts more difficult. Interestingly, neural networks have not been effective for this problem due to the absence of a large 3D pointing direction data set. Recent work have addressed these challenges, however the proposed approaches have limitations as they find two-dimensional (2D) pointing directions [4] or require specific body parts to be detected [5, 6] or output quantized 3D directions [6]. Thus, they are not applicable to tasks in which the nature of human-robot interaction requires a fine-grained 3D direction estimate - with the additional need that the estimation must be possible even if specific body parts are not easily detectable. In this work, we want our robot to be capable of detecting the 3D pointing direction precisely - regardless of hand distance, background variability and the detectability of specific body parts.

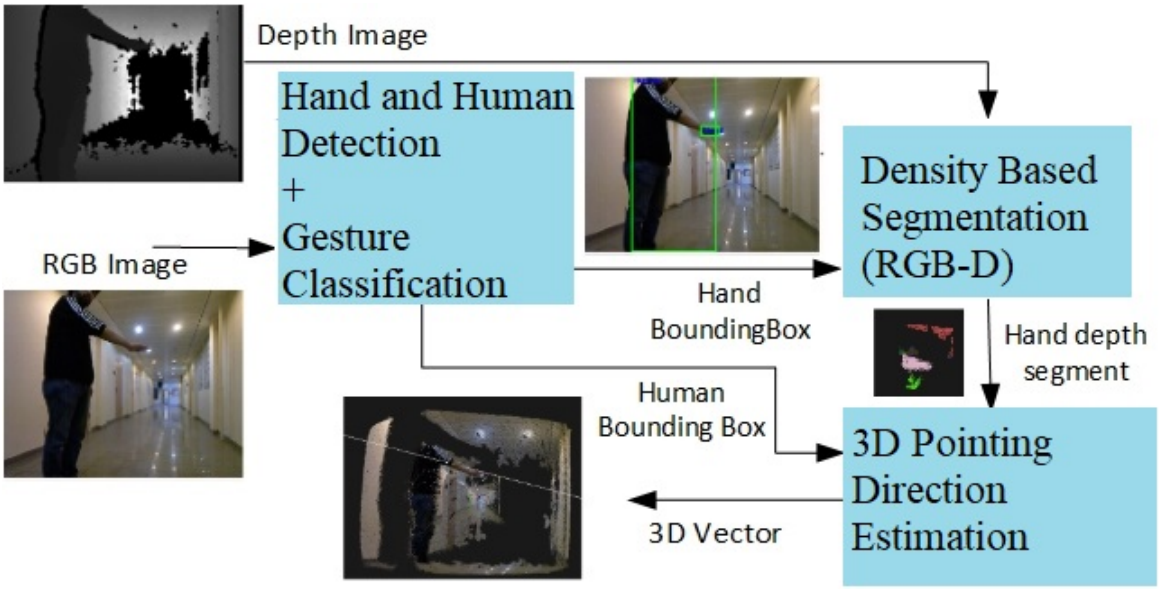

Figure 2.1. Pipeline of our system for the detection of 3D pointing direction. First, the robot generates hand and human regions in the RGB image using a specially trained CNN. Then, hand region is classified into pointing or non-pointing gesture classes. Robot then applies a density-based segmentation on the RGB-D data of the resulting hand bounding box. The 3D pointing direction vector is computed from the implicit geometry of the hand depth segment. As such, the robot uses only RGB data, only depth data or both together as needed.

To tackle these issues, we present a novel approach with a pipeline of three stages as shown in Figure 2.1. First, the robot uses a network that is built on a deep convolutional neural network in order to find hands and humans in the incoming RGB image. The 3D hand segment is then determined by applying a density-based segmentation algorithm on the corresponding RGB-D data of the hand. Finally, the 3D pointing direction vector is computed from the geometry implicit in the hand segment's depth data.

The outline of the rest of the chapter is as follows: Related literature is reviewed in Section 2.2. The proposed method is presented in Section 2.3. Experimental results on comparative hand detection performance and 3D pointing direction detection are discussed in Section 2.4.

2.2. Related Literature

Most of the proposed approaches for detecting the 3D pointing direction take advantage of the extensive work done on hand detection. First, we present a brief review of this work. Next, we review work on the detection of 3D hand pointing direction. Finally, we briefly review the available data sets.

2.2.1. Hand Detection

Hand detection plays a critical role in the robustness of the 3D pointing direction estimation. Early work on hand detection have mostly used skin color [7–9] or shape features via boosted classifiers [10, 11]. Context information from human body parts has also been utilized [12–14]. A combination of skin color, shape and context have been used in [15]. However, the performance of these approaches tends to be lower if the robot is to operate in a wide-range of scenarios with hand appearances varying greatly with respect to color, shape and size. Recently, CNN based detectors have been addressing these issues. A multi-scale Faster R-CNN architecture is proposed in [16], however the algorithm is slow for real-time applications. Faster R-CNN and skin segmentation are combined in [17]. A rotation estimator CNN is used to in [18]. A Region-based Fully Convolutional Network is presented in [19]. However, their performances on benchmark data sets such as Oxford hand data set have been reported to be low - possibly due to the lack of training data. Mask R-CNN is used to reliably predict a bounding box segmentation of hands in cluttered environments [4], however it is slow for real-time applications. Yang et al. have developed a modified MobileNet with SSD as a real-time hand detector [20]. A Faster R-CNN based detector that is trained along with a CNN that can also reconstruct hands in the ROIs is presented in [21], however the required computational resources are very high for real-time applicability.

2.2.2. Pointing Gesture Classification

In the recent years, CNN based methods have dominated hand gesture classification. Most of these methods take RGB images as input.

However, there are also works that use depth data as well [22]. Background removal is a common practice among the classification methods. Segmentation is achieved using methods such as skin color [23] or watershed algorithm [24]. In most work, the input of CNNs are directly RGB images [25, 26]. However, other sources of information such as local binary patterns [24] or hand edge images [27] are used as well. There are also works targeting RGB videos [26]. Key frames are extracted for a better accuracy in [28] using image entropy.

2.2.3. 3D Hand Pointing Direction Detection

In most work, hand detection and 3D hand pointing direction estimation are done separately. For example, once the hand regions are determined, the direction estimation problem is posed as an object detection problem based on the similarity of features such as hue and surface normals [29]. The 3D pointing direction is estimated through interpolating SVM scores of different direction classes [30]. Recently, end-to-end systems are also being developed [31]. The applicability of these methods vary depending on the sensing, hand range, background variability, whether they require the detection of specific body parts and whether a continuous or quantized 3D direction estimate is output. A list of the proposed approaches along with how they fare with respect to these issues is presented in Table 2.1.

The sensor type is important because the type of data (RGB image, depth, RGB-D) determines what can be computed from the data. For example, typically 2D fingertip positions can be found with a RGB camera [32–35]. While there are approaches that infer 3D pointing direction from the 2D position data [31], their accuracy is limited as there is no distance information in 2-D data. One remedy is to use multiple RGB cameras with different locations [36]. However, this is restrictive in mobile robotics applications. The 3D pointing direction estimation cannot be done reliably with RGB data. Thus, it has become evident that depth data as obtained from stereo cameras [9], depth cameras [37], time-of-flight cameras [38], or RGB-D sensor such as Kinect [39] is integral to the precise detection. There are also work that assume the availability of Kinect skeletal data [40, 41]; however these approaches are only applicable if there is a movement of the pointing hand. From the robotics perspective, it is important to detect hands in unconstrained conditions [42].

The remaining issues pertain to the applicability of the method in different HRI interaction scenarios. In some works, the hand is assumed to occupy a large part of the robot's sensory viewing volume and hence hand detection is rather easy [34–36]. These approaches are not applicable in many HRI applications since they require the hand to be the closest object to camera. Hence, approaches that are applicable with varying hand distances are preferred. Another issue is regarding the background variability. In some work, the background is either assumed to be fixed or a simple background [29, 36]. Again, it is not possible to use these approaches in arbitrary settings. Another issue is that some approaches use the relative geometry between the hand and another body part to determine the 3D pointing direction. For example, the estimation is computed from the line from face to hand or elbow to hand [38], wrist to hand [39] or head to hand [5]. Thus, the detection of these body parts needs to be both possible and reliable. In addition, some approaches require specific hand topologies - such as the back of the hand facing upwards [30] or person-specific fist models [37]. As such, these methods are not practical in most HRI tasks. Finally, in some methods, the 3D pointing direction is estimated in a quantized manner - such as the 26 direction quanta [6]. As such, the resulting estimates turn out be very coarse due to the quantization.

2.2.4. Hand Pointing Direction Data Sets

Most of the data sets on human pointing direction are based on close range and/or egocentric RGB images of the hand [33, 34]. As such, the hand is the dominant object in the image. Hence, the data set is not suitable for testing performance with varying hand distance. An RGB data set with 2D pointing directions is given in [6], but the annotations are such that edge directions are quantized into 8 levels. They also provide a RGB-D data set for 3D pointing directions, however, the annotations provide 3D edge directions that are quantized into 26 directions. A 3D pointing direction data set is given in [29]. However, it's not suitable to assess performances wide-ranging scenarios since all the images are from a fixed table setup.

Table 2.1. Related work. Cont. 3D refers to whether the output is a continuous or quantized 3D vector.

Method	**Input**	**Varying Distance**	**Background variability**	**Body parts**	**Cont. 3D**
[9]	2 RGB	✓	✓	Head	✓
[36]	2 RGB	✗	✗	-	✓
[39]	RGB-D	✓	✓	Legs & face	✓
[38]	ToF	✓	✓	Head & torso	✓
[40]	RGB-D	✓	✓	Skelatal	✓
[30]	1-2 RGB	✓	✗	-	✓
[41]	RGB-D	✓	✓	Skelatal	✓
[29]	RGB-D	✗	✗	-	✓
[31]	RGB	✓	✗	-	✓
[37]	D	✓	✗	-	✓
[43]	RGB-D	✗	✗	Skelatal	✓
[5]	RGB-D	✓	✓	Face	✓
[6]	RGB-D	✓	✓	-	✗
Proposed Work	RGB-D	✓	✓	-	✓

2.3. Method

We propose a three-stage 3D hand pointing direction detector. It operates on the RGB-D data coming from Kinect. First, hands and human are detected using a newly trained CNN running on the RGB image. Next, the point cloud object corresponding to the 3D hand segment is found. Finally, using the 3D hand segment, the 3D pointing direction is computed. The direction is parametrized by $(v_{pd}, o_{pd},)$ where $v_{pd} \in R^3$ corresponds to the orientation and $o_{pd} \in R^3$ corresponds to the 3D point it passes through. Hence, the direction is defined by the line $o_{pd} + sv_{pd}$ with $s \in R$.

2.3.1. Hand and Human Detection

The processing starts with the detection of hand and human regions from the RGB image. The latter is needed as a reference for positive direction. However, no specific human part is required to be seen by the RGB-D sensor - as this would have imposed restrictions on our target scenarios. Unfortunately, there is no available hand and human region detector.

For this, the learning network architecture is selected. For this, we have decided to use the YOLOv4 network - since it is a robust and fast state-of-the-art CNN. In particular, it is known to achieve 43.5% AP on MS COCO data set with over 60 fps on Tesla V100 [2]. Its input is a $416 \times 416 \times 3$ RGB image. A larger image size may lead to better accuracy, but the network would slow down. The output is a set of 2D bounding boxes corresponding to the detected objects. The YoloV4 network is retrained or used in three stages as to detect only hands and humans. This is done using the Darknet framework [44]. In all, its architecture remains the same except the layers before Yolo prediction layer. This is necessary in order to change number of classes YOLOv4 can detect.

Unfortunately, it is not possible to directly retrain the YOLOv4 network since there is no available data set with both hand and human annotations. However, there are data sets that provide each annotation separately. One example is the COCO-hand data set and COCO-human data set [45]. A three-step process is applied.

First, a hand detector is created using the COCO-Hand data set [4]. Next, the resulting hand detector is run on human images of COCO data set as to create an annotated data set with hand and human labels. This data set is then used to train the network once more. Hence, the network learns to detect both hands and humans. Altogether 73672 frames are used in learning. An example of the resulting hand and human detection is given in Figure 2.2.

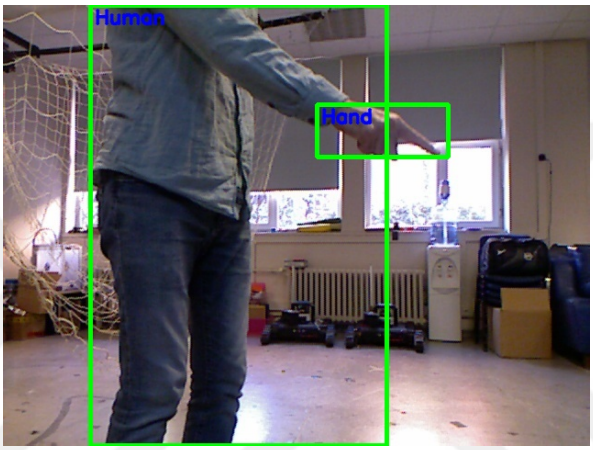

Figure 2.2. Hand and Human Detection.

2.3.2. Pointing Gesture Classification

After finding hand bounding box, the robot needs to determine if the hand is performing a pointing gesture. Similar to most work, a CNN is used to classify hand regions. The classifier outputs whether the input gesture is a pointing gesture or not. The architecture of our classifier as given in Figure 2.3 is inspired by that of [46]. Differing from it, a normalization layer is added in order to speed up the training. Here, categorical cross-entropy loss function given by

$$J = -\frac{1}{K}\sum_{i=1}^{K}\left[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right]. \tag{2.1}$$

where y_i is the true label, $\hat{y}_i$ is the predicted output. Training is done for 20 epochs.

We created a RGB data set consisting of both pointing and non-pointing gestures. The data is obtained from the robot's Kinect sensor.

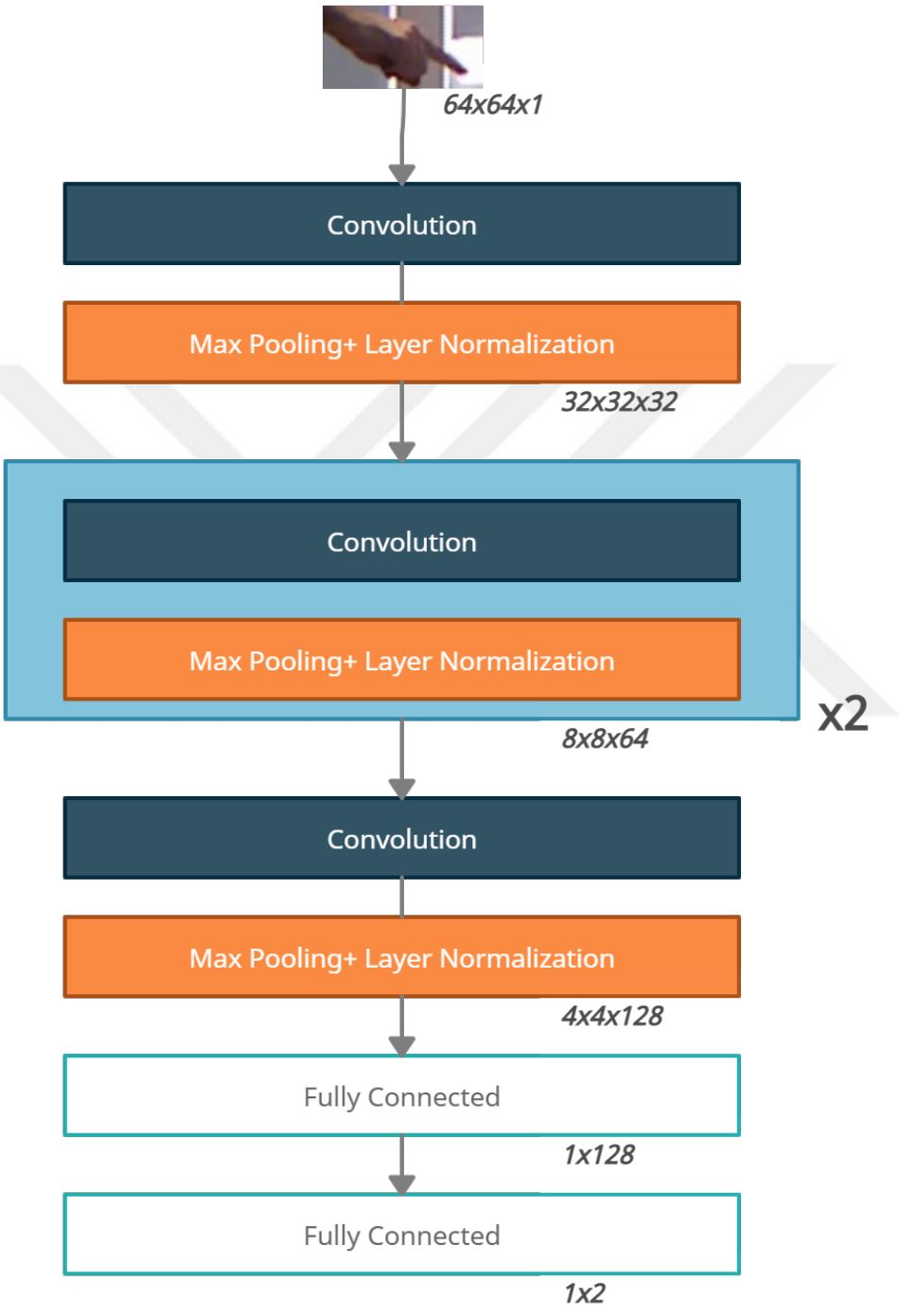

Figure 2.3. Architecture of Pointing Gesture Classifier.

Users in front of the robot perform pointing and non-pointing gestures at varying distances in the range 0.5-5 m. Following, the hand detector as developed previously is applied on the images in order to extract the hand bounding boxes. Finally, hand images are labeled as pointing or non-pointing manually. In total, there are 2557 pointing, 3156 non-pointing hands. The data set includes images with from varying illumination conditions, different hand distances. Some samples from the data set are given in Figure 2.4. The data set is divided into training, validation and test data sets.

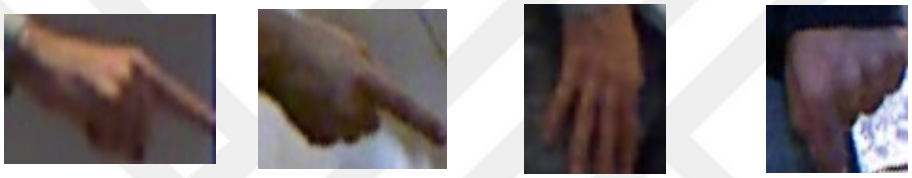

Figure 2.4. Samples from Pointing Gesture Data set.

2.3.3. 3D Segmentation

The goal of the next stage is to determine the 3D hand segment in the RGB-D data. First, the bounding box of the hand (as determined in the previous stage) is mapped to a 3D volume - considering the RGB-D data. This requires the RGB data and depth data to be calibrated. The calibration is achieved using a method as prescribed in [47]. The result is a 3D hand region as shown in Figure 2.5.

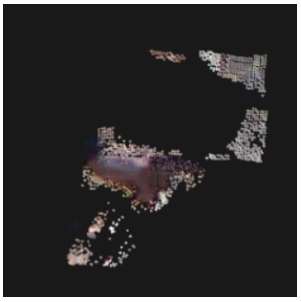

Figure 2.5. 3D Hand Region.

Following, a density-based segmentation algorithm is applied to the 3D points that lie within this volume in order to find the 3D hand segment. For this, we use a modified version of density based clustering algorithm [3]. As the original algorithm is designed for point cloud segmentation, it is modified to consider both depth and RGB data. The algorithm is based on clustering adjacent 3D points. Let $\mathcal{P}$ refer to the RGB-D data. Each point is associated with an segment label. Let L denote the index set of labels. As such, we can determine the set of 3D segments $\mathcal{O}$ where each segment $o \in \mathcal{O}$ is determined by the RGB-D data $\mathcal{D} \subset \mathcal{P}$ having the same label $l \in L$. The algorithm uses the fact that the sensory geometry is defined by a spherical coordinate system with the origin at Kinect optical center with known pan angle resolution δ_ϕ and tilt angle resolution δ_ψ. Let Q be the corresponding point cloud data expressed in the spherical coordinates. Each point $p \in \mathcal{P}$ is associated with a pan angle $\phi \in [0, 2\pi)$, tilt angle $\psi \in [0, \pi)$, depth $\rho(\phi, \psi)$ and HSV data $c(\phi, \psi)$. A connected component algorithm is applied in the spherical coordinate system. Two metrics are used for determining if two points are adjacent. First, proximity in the spherical coordinate system is checked. Hence, for a point $q \in Q$ where $q = \begin{bmatrix} \phi & \psi & \rho(\phi, \psi) \end{bmatrix}^T$, its spherical distance $d(q, q')$ to another point q' is calculated as

$$d(q, q') = \frac{(\phi - \phi')^2}{\tau_\phi^2} + \frac{(\psi - \psi')^2}{\tau_\psi^2} + \frac{(\rho - \rho')^2}{\tau_\rho^2} \tag{2.2}$$

where $q' = \begin{bmatrix} \phi' & \psi' & \rho(\phi', \psi') \end{bmatrix}^T$. The parameters $\tau_\phi, \tau_\psi, \tau_\rho > 0$ are determined considering the respective resolutions - namely $\tau_\phi = \gamma_1 \delta\phi$ and $\tau_\psi = \gamma_2 \delta\psi$ where γ_i, $i = 1, 2$ are determined empirically. The parameter τ_p is set depending on how much depth variation is allowed. Second, color similarity is checked. This is simply defined by the L_2 norm of the difference between their respective HSV values $c(q)$ and $c(q')$. A point q' is considered to be in the neighborhood $\mathcal{N}(q)$ of q:

$$q' \in \mathcal{N}(q) \text{ iff } \begin{cases} d(q, q') \leq \tau_{p_1} \text{or} \\ \tau_{p_1} < d(q, q') \leq \tau_{p_2} \text{ and } \|c(q) - c(q')\| \leq \tau_{hsv}. \end{cases} \tag{2.3}$$

The cardinality of the neighboring points should satisfy a minimum neighbor criterion - namely $|\mathcal{N}(q)| > M_1$ where $M_1 > 0$ denotes the minimum number of neighbors. In this case, q is taken as a core point for a new segment. The segment is expanded repeating this process for all neighboring points. The parameter M_1 is set to be a percentage of maximum cardinality possible. The computational complexity of this algorithm is kept constant by using an indexed organization of the RGB-D data.

Finally, the 3D hand segment is found based on the observation that the hand occupies the biggest area in the 3D hand volume and is also one of the closest objects therein. Thus, each segment can be associated with a score that measures how large it is and how close it is. In these calculations, the points whose depth readings are outside the Kinect depth sensing range of $0.5m - 5m$ are not included. The segment with the highest score is assumed to be the hand segment.

2.3.4. Pointing Direction Estimation

The geometry common to all hand pointing gestures is that a big part of the hand is physically gathered around a direction [48]. This is typically done either through extending the index finger or flexing the remaining fingers into the palm with possibly with the thumb to the side. Our estimation process considers this and consists of three steps.

The pointing direction is inferred from the geometry of the 3D hand segment. For this, the covariance matrix Σ is constructed. Let its eigenvalues be denoted by $|\lambda_0| > |\lambda_1| > |\lambda_2|$ with corresponding eigenvectors are v_0, v_1, v_2. We expect the final estimation to be close to the principal eigenvector v_0. However, it needs to fine-tuned considering the hand pointing geometry.

The pointing part of the hand is determined through finding the two planes whose normals are orthogonal to v_0 and whose neighborhoods contain the largest number of 3D hand segment points. These points will correspond to the pointing part of the hand. The first plane has normal in the v_1 direction while the other has normal in v_2 direction.

First, the two outermost planes in the v_1 direction containing hand segment data are determined. Following N_1 orthogonal planes are placed at equidistant locations along v_1. Let the plane separation distance be denoted by $\delta v_1 > 0$. A point $h \in \mathcal{H}$ is considered to be in the neighborhood of a plane if its distance to the plane is less than $\frac{\delta v_1}{\sigma_1}$. Finally, the plane with the largest neighborhood cardinality is determined. Let the respective neighboring points be denoted by $\mathcal{H}_1 \subset \mathcal{H}$. Following, the process is repeated for the v_2 direction - considering now the set of points $\mathcal{H}_1$. In this case, N_2 planes are placed at equidistant locations with respect to the vector v_2. For each plane, the neighboring points are determined by comparing their distance to $\frac{\delta v_2}{\sigma_2}$. The final set of points $\mathcal{H}_2$ consists of 3D points corresponding to the pointing part of the hand. The orientation of these points is our final pointing direction estimation v_{pd}. The 3D point o_{pd} is selected randomly from $\mathcal{H}_2$. For example, in the classical pointing gesture which consists of a fist and a pointing index finger, the final points will be those corresponding to the index finger. For such a 3D hand segment, the two planes with the largest neighborhoods are shown in Figure 2.6. The intersection (blue points) of these neighborhoods is observed to be parallel to the pointing direction. The corresponding 3D pointing direction (white line) can be seen in Figure 2.7.

After finding the line, it remains to determine which end of the line is the pointing part. We used human detection to solve this issue. If the vector from hand center to one end of the line makes a positive angle with the vector from human center to hand center, then the corresponding end of the line is taken as pointing side. If not, the other part is selected.

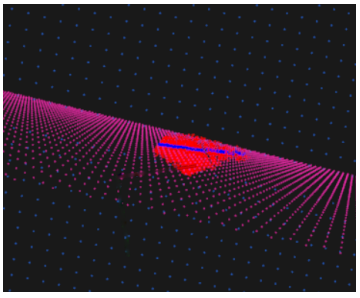

Figure 2.6. Planes Passing Through Hand Orthogonal to v_1 and v_2.

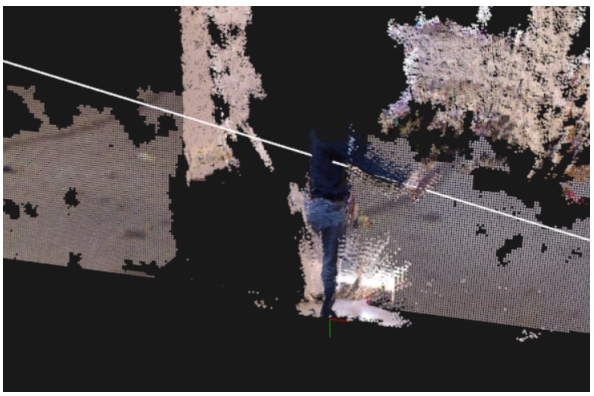

Figure 2.7. Detected 3D Pointing Direction.

Table 2.2. Parameter table.

Parameter	Meaning	Section	Value
τ_ϕ	Pan normalization	2.3.3	3
τ_ψ	Tilt normalization	2.3.3	3
τ_ρ	Radius normalization	2.3.3	0.05
M_1	Neighborhood threshold	2.3.3	10
τ_{p_1}	Strict spherical dist. threshold	2.3.3	0.8
τ_{p_2}	Loose spherical dist. threshold	2.3.3	1.1
τ_{hsv}	HSV distance threshold	2.3.3	0.5
N_1	Number of planes $\perp v_1$	2.3.4	3
N_2	Number of planes $\perp v_2$	2.3.4	2
σ_1	Distance threshold for plane $\perp v_1$	2.3.4	2
σ_2	Distance threshold for plane $\perp v_2$	2.3.4	2

2.4. Experiments

In this section, we present our experimental results regarding hand detection and pointing direction estimation. First, we compare the performance of our hand detection method with the state-of-art hand detection methods. Next, we evaluate the reliability of pointing gesture detection. Finally, we study the performance of our pointing direction estimation algorithm using this data set - including a study of performance sensitivity to the parameter values. The parameter settings are as given in Table 2.2. The average execution time of our overall method is 0.19 sec on a computer with 2.59 GHz CPU and GTX 1650 though it may change between 0.1 sec and 0.5 sec depending on the size of the hand bounding box.

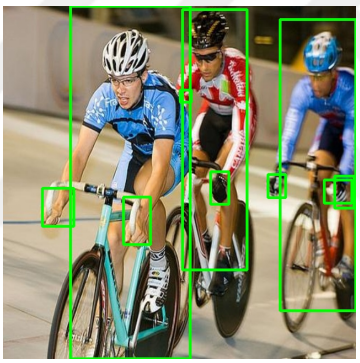

Figure 2.8. A Sample Detection with YOLOv4 [15].

2.4.1. Hand Detection

Hand detection experiments are conducted using the benchmark Oxford hand data set. A sample detection of the proposed YOLOv4-based model is shown on Figure 2.8. Average precision and speed of hand detection are compared with the previous approaches in Table 2.3. Computation time of the algorithms is given in Table 2.4. It is observed that our model achieves a good balance between average precision and run-time. While the approach of [21] is the only one that has better average precision, its run-time is significantly higher than our detector. On the other hand, the fastest model [20] has significantly lower precision. Hence, the experimental evaluation of our proposed YOLOv4-based model indicate that that we achieve hand detection performance comparable with the state-of-the-art methods.

Table 2.3. Average precision (AP).

Method	AP
Faster R-CNN + skin [17]	49.51%
RPN [18]	57.7%
RPN + Rotation Estimator [18]	58.1%
MS-RFCN [19]	75.1%
Hand-CNN [4]	78.8%
MobileNet-SSD [20]	83.2%
Faster R-CNN + GAN [21]	87.6%
Proposed YOLOv4-based model	**84.78%**

Table 2.4. Computation time.

Method	Time (sec.)	Hardware
RPN [18]	0.1	Titan X
RPN + Rotation Estimator [18]	1	Titan X
MobileNet-SSD [20]	0.0072	Titan X
Faster R-CNN + GAN [21]	0.1121	GTX1080Ti
Proposed YOLOv4-based model	**0.032**	GTX 970

2.4.2. Pointing Gesture Detection

Experimental results on pointing gesture detection are given on Table 2.5. As explained, the data set has been divided into learning, validation and test sets. The accuracy of detection of positive samples is 94.5% while that negative samples is 94%. As it can be seen, we have reached satisfactory results considering that pointing gestures have been made at a wide range of distances to the camera. Still, gesture classification data set needs to be expanded to include various scenarios to have a more robust classifier.

Table 2.5. Accuracy ratings for pointing gesture classification.

Data	# Pointing Images	Accuracy	# Non-Pointing Images	Accuracy	# Total	Accuracy
Training	1860	99.9%	2253	98.8%	4113	99.3 %
Validation	441	95.0%	587	92.5%	1079	93.5 %
Test	256	94.5 %	316	93.7 %	572	94.0 %

Table 2.6. 3D Pointing direction data set.

Range	Hand Distance range (m)	# of Images
Close	0.5 - 1	41
Mid	1 - 1.7	169
Far	1-7 - 4.5	77

2.4.3. 3D Pointing Direction Data Set

The 3D pointing direction data set is introduced as a benchmark for future studies. To the best of our knowledge, there are no such data sets with continuous 3D pointing direction annotations in varying backgrounds. The data is obtained with a Kinect sensor. The rotation and translation parameters between RGB and depth sensors and the camera parameters that map 2D RGB-D data to 3D point cloud data along are provided. However, while saving depth images, we wrote them to ".jpg" files which are unable to hold 16 bit pixel information. So, the written depth pixels are in 8 bit format, which causes them to be more noisy then usual Kinect depth readings.

The data set consists of 287 RGB-D images with varying hand pointing gesture scenarios as viewed by a mobile robot. The viewing distances ranges from close to far as shown in Table 2.6. We have also tried to include various pointing gestures that may be encountered in real-life applications. For example, the pointing finger is not always parallel to arm as one would expect. RGB and depth samples from data set are given in Figure 2.9. Image labels contain 3D pointing directions. The pointing hand locations in the RGB images are also provided for studies that focus purely on the performance of the direction estimation.

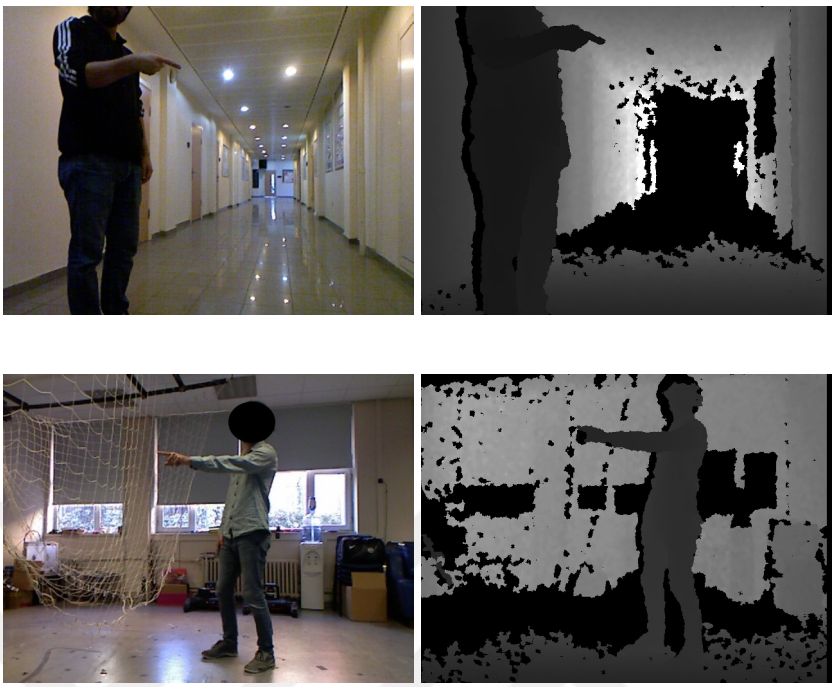

Figure 2.9. RGB and Depth Samples from Pointing Direction Data Set.

2.4.4. Pointing Direction Detection

In this section, we study the performance of 3D pointing direction detection. The evaluation is done using the benchmark 3D pointing direction data set. We study the accuracy of the detected 3D pointing direction in different distance scenarios and that choice of parameter values are not critical for the performance of the algorithm.

The estimation error is defined as the angle between the estimated direction and the ground truth direction. Overall, average estimation error is 14.34° as seen in Table 2.7. We also consider accuracy performance considering three different thresholds 10° to 30°. Estimations with errors larger than the given threshold are considered to be failures. The resulting accuracy rates are seen in Table 2.7. It is observed that 87.1% accuracy is achieved for 30°.

Finally, the effect of parameter selection on the 3D pointing direction performance is studied.

Table 2.7. Accuracy results for varying thresholds and average accuracy for 30°.

Average angle error	<10°	<20°	<30°
14.34°	27.8%	66.6%	**87.1%**

Table 2.8. Accuracy rates and average angle errors for different parameter selections. Parameters are given in the order : (N_2, σ_2) , (N_1, σ_1).

Parameters	<10°	<20°	<30°	Average angle error
(3,2),(2,2)	27.8%	66.6%	87.1%	14.34°
(4,3),(2,2)	28.6%	61.7%	87.8%	14.72°
(4,3),(3,3)	29.6%	62.4%	86.8%	14.97°

There are 5 parameters to consider: compression parameter τ_c, (n, σ) values for planes parallel to v_3 and (n, σ) values for planes parallel to v_2. The results for different set of parameters are given in Table 2.8. It is observed that similar accuracy rates are obtained. Hence, variations in the used values of these parameters do not seem to affect the resulting accuracy.

We have also investigated failure cases - namely the angle error is more than 30°. For example, a failure case is shown in Figure 2.10. Failure tend to occur when the pointing direction is towards to the robot. This is attributed to the fact that only the front part of the hand is seen by depth sensor. As such, the clustering of hand points around another direction is more dominant than that of the pointing direction.

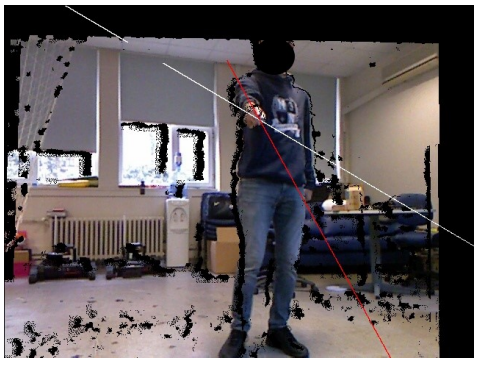

Figure 2.10. A PDE Failure Case.

3. TARGET OBJECT DETECTION

3.1. Introduction

After finding the 3D pointing direction, the next stage is to have the robot determine the pointed object. This problem has been addressed in some of the related work such as [29]. The typical approach is to assign a score to each object and then select the object with the best score. Although this approach seems reasonable, it is not easy to find a reliable single score for the target object detection. One common approach is to use the angle between the pointing direction and the line from hand to object [41]. However, such a score fails in many scenarios such as another object intersecting with pointing direction. Another score formulation is based on the orthogonal distance of the object to the pointing direction. However, in this case, objects that are close to the hand becomes a problem. An example is shown in Figure 3.1. The user is pointing to the 'plate'but the cans's orthogonal distance to the pointing direction is small as well. These suggest that the score formulation should include various criteria.

For this method, a novel reasoning method is proposed. In this method, the robot first considers the cone with the pointing direction set as its axis and with a small apex angle. All point cloud objects within this volume are then determined. If no objects are found, then the apex angle is enlargened until some objects are determined. The objects are then associated with a target score that is constructed in a weighted manner considering several criteria such as angle, distance to the hand and another metric to eliminate objects that are close to the hand but not in the way of pointing direction. In the remaining part of this chapter, related literature is presented followed by our approach to the target object detection problem. Finally, experimental results are reported.



Figure 3.1. An Example Where Orthogonal Distance Score May Fail to Detect Correct Object [49].

3.2. Related Literature

In the literature, target object detection algorithms are generally presented along with the pointing direction estimation. The pointed object is determined assuming that a pointing direction is found. Most commonly, a score is assigned to each object and the object with the top score is set to be target object.

Commonly, the score is formulated based on the angle between the pointing direction and the line from elbow to object [41] - assuming that the pointing direction is along the line from the elbow to the wrist. In some work, the target object is found from the 2D pointing direction. For example, scores are defined for the bounding boxes associated with the objects based on the ratio of the length of intersection of pointing direction and object to the length of diagonal of object bounding box [50]. It is observed purely that angle based methods tend to perform poorly when there are multiple objects intersecting with pointing direction.

Similar to the angle score, the distance of the objects to the pointing direction has also been considered. For example, each point in 3D space is assigned a score based on its distance to the pointing direction. Object score is computed by summing the scores of all points belonging to it. The object with the highest score is selected [51].

In [29], the target object is selected considering the weighted sum of the distances of an object to all potential directions with each weight being equal to the confidence score of the direction. These methods tend to favor objects close to the hand and may yield wrong detections when the target object is not necessarily the closest object.

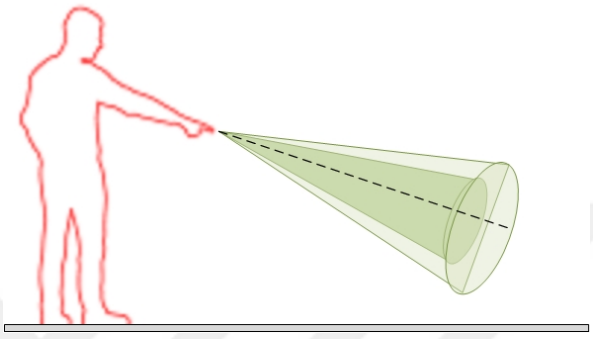

Figure 3.2. The target object is first searched in a smaller neighborhood and then in a larger neighborhood.

3.3. Method

In the proposed approach, the robot first determines all the point cloud objects in the scene. This is achieved using the point cloud segmentation algorithm as presented in Section 2.3.3. As this algorithm assumes that objects are located on ground and separately from each other, a ground removal algorithm is applied prior to segmentation [52]. As such, it is not possible to segment objects that are located on another object such as objects sitting on a table top.

Following, the robot finds the pointed object via doing a search among the resulting point cloud objects. Our approach to the problem tries to mimic the human way of finding the pointed object. Let $p_1 \in R^3$ and $p_2 \in R^3$ denote two outermost hand points in the pointing direction. When people want to identify the targeted object, they firstly look for objects that are in the close neighborhood of pointing direction as shown in Figure 3.2. This corresponds to a cone with apex angle α_1. An object is deemed to be in this neighborhood if it contains at least one point that falls within this volume.

If it cannot find any object within this volume, then the search volume is expanded to a cone with a larger apex angle α_2. Similarly, this is determined by checking for at least one point overlap. Let $\mathcal{S}$ denote the index of point cloud objects.

The robot then computes a 'targetness' score for each object. The scores are constructed using three different measures:

- The angle between the pointing direction and the line connecting p_1 to the object: $\theta_1(i)$
- The smallest distance of the object to the hand $d(i)$
- The angle between the pointing direction and the line connecting p_2 to the object: $\theta_2(i)$

For $i \in \mathcal{S}$, if $\theta_1(i) < \alpha_1$, this suggests that the object i is the smaller volume region. Then let $s_1 : \mathcal{S} \rightarrow R^{\geq 0}$ denote the corresponding function that is defined as follows:

$$s_1(i) = d(i) \sin \theta_1^3(i) \sin(\theta_2(i) - \theta_1(i))^2. \tag{3.1}$$

In this case, the robot needs to select an object that is close to the hand and has a small angle with pointing direction. The first two terms check for this. The last term is to eliminate objects that are close to hand but does not make small enough angle with the direction line. If the difference between them is large, we can assume the particular object is not intersecting with pointing direction. An example of such an object is given in Figure 3.3. As it can be seen, difference between θ_1 and θ_2 for ketchup bottle is large, which causes $s_1(i)$ of the object to be large. Therefore, ketchup bottle will not be selected.

If $\alpha_1 < \theta_1(i) < \alpha_2$, then the object is in the larger search region. Let $s_2 : \mathcal{S} \rightarrow R^{\geq 0}$ denote the corresponding score map. It is defined as follows:

$$s_2(i) = d(i) * \sin |\theta_1(i) - \theta_2(i)|^3 . \tag{3.2}$$

In this score, θ_1 since it is not a reliable metric. The summary of our approach is given in Algorithm 1.

Algorithm 1 Find pointed object

for $i \in \mathcal{S}$ **do**
  $\theta_1(i) = calcAng(i, p_1)$;
  $\theta_2(i) = calcAng(i, p_2)$;
  if $\theta_1(i) < \alpha_1$ **then**
    $s_1(i) = d(i) \sin \theta_1(i)^3 \sin(\theta_1(i) - \theta_2(i))^2$;
    $closeScores.add(s_1(i))$;
  end if
  if $\theta_1(i) < \alpha_2$ **then**
    $s_2(i) = d(i) \sin |\theta_1(i) - \theta_2(i)|^3$;
    $restScores.add(s_2(i))$;
  end if
end for
if $closeObjs.size() > 0$ **then**
  return $\arg\min_i closeScores$
else
  return $\arg\min_i restScores$
end if

3.4. Experimental Results

This section presents experimental evaluation of the proposed target object selection method. In the experiments, $\alpha_1 = 15^\circ$ and $\alpha_2 = 30^\circ$ based on experimental observations. The experiments are done using the Multimodal Human-Robot Interaction data set (MHRI) presented in [49]. The data set consists of video frames of 10 users pointing to 10 different objects in varying scenarios. The environment is rather restricted with the users pointing to a variety of objects on a table.

Furthermore, pointing gestures occur in only a small subset of the frames in each sequence - as the scenes before and after the pointing gestures are made are also recorded. For testing, only the frames containing the pointing gestures are used. Here, ground truth pertaining to the target objects is provided. A sample scene is as shown Figure 3.4.

Similar to [49], the goal is to only evaluate the target object detection algorithm. Thus, table removal is done and then segmentation is applied on the resulting scene data. The parameters of the ground segmentation algorithm are manually adjusted in each frame as to correctly remove the table pixels. Even so, only 45 out of 100 videos can be used - since in the rest it is not possible to completely remove the table. As the number of frames with pointing gestures is not the same across all the videos, a fixed number of frames per video is used in the experimental evaluation. Here, this is arbitrarily set to be 4. Hence, it total, the evaluation is done considering $45 \times 5 = 225$ frames. Let it be noted that in [49], only 90 frames are considered.

The results of our experiments on the data set are given in Table 3.1. Target object detection is taken to be correct if it actually overlaps with the ground truth target object. For example, in some frames, it is observed that the part of the targeted object is removed after applying ground segmentation. An example of such a case is given in Figure 3.4. Also, part of the table pixels remained even after ground segmentation and segmented as part of the target object. We considered them correct if no other object in the table setup is not included in the particular segment. Overall, target object is found correctly in 72.4%. As it can be seen from the table, we did not include frames from User 10. That's because User 10 was touching the target objects, which is not our target scenario i.e. segmentation does not work. Uneven distribution of data taken from other users stems from similar reasons.

The results are compared with those presented in [50]. The results are given in Table 3.2. Since there are some images in the data set that can not be used for testing , one has to select which images to use manually. Also, to remove the effects of the segmentation on results, one has to choose correctly segmented images. These are done both in our work and in [50] did.

Table 3.1. Results on MHRI data set.

Users	**# Frames**	**Successful Frames**	**Success Rate (%)**
User 1	25	19	68.0
User 2	25	15	60.0
User 3	35	16	45.7
User 4	35	29	82.8
User 5	25	24	96.0
User 6	15	5	33.3
User 7	15	15	100.0
User 8	35	25	71.4
User 9	15	15	100.0
Total	225	163	72.4

Table 3.2. Comparative results on MHRI data set.

Method	**Success Rate (%)**
[50]	46.7
Proposed Method	72.4



Figure 3.3. θ_1 and θ_2 for the Ketchup Bottle [49].

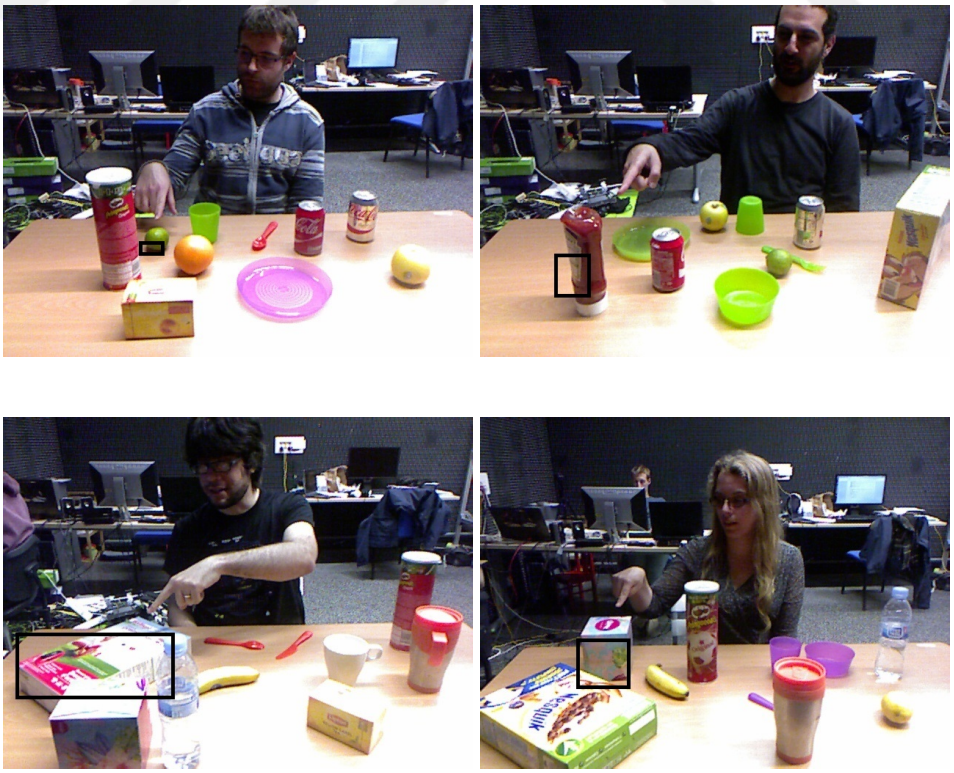

Figure 3.4. Sample Target Object Detections on MHRI Data [49].

4. OBJECT LEARNING

4.1. Introduction

The final stage of the proposed approach is classifying or learning the pointed objects as necessary. After target object detection, the robot needs to either recognize the object as belonging to a previously learned class or to use the object for learning if it can not be classified. When the target object is not recognized, we assume that the robot can get multiple views of the object. This can be achieved either through having the human can show the object from different angles or through the robot actively exploring the object from different angles. In this way, the robot will have the knowledge of which objects are from the same class during learning. Thus, this problem can be seen as a semi-supervised incremental learning problem. This is a challenging problem as robot needs to learn an accurate representation of a class from a small amount of data.

This is done in three steps. First, the robot encodes the respective point cloud data using deformed sphere approximation representation [3]. Then, it associates this descriptor with the knowledge existing in its objects' memory. The objects' memory has a tree organized structure in which terminal nodes correspond to distinct object classes. In this structure, each parent node is associated with a set of classifiers to assign a new input to one of the child nodes. The robot conducts a top-down search in its objects' memory in order to determine whether it can recognize the target object or not. This is accomplished through traversing down the memory hierarchy and using the classifiers associated with the reached node to decide on which child node to move to. This is repeated until either classifiers' responses are all low or or a terminal node is reached. The former case implies no recognition and the robot expects human to show the object from different viewpoints as to form its knowledge of this object. It then adds this to its existing objects' memory using an incremental hierarchical clustering method.

The outline of this chapter is as follows: Related literature is discussed in Section 4.2. The proposed approach is presented in Section 4.3. Experimental results are then reported in Section 4.4.

4.2. Related Literature

In the literature, incremental learning of objects' classes has been studied extensively. Most work consider learning without any human interaction. Some of these work only allow the learned class models to be updated and hence do not enable the learning of new classes [53, 54]. As this is rather restrictive, a plethora of approaches that enable continual learning has been developed. These are categorized depending whether supervision (labels) is required or not and whether the method is offline or online.

As an example, with supervised offline learning methods, CNNs are retrained in a fast and efficient way when new training data appears [55] or new nodes are added to the output layer of the architecture for new classes [56]. The problem of having unbalanced proportion of learning examples per class is addressed by introducing a regularized least squares method [57]. The supervision is done through checking the correctness of recognition decisions by a human operator with incorrect decisions being fed back to the Hough forest detector for updating [54]. Alternatively, unsupervised incremental approaches to learning have also been developed. As explained, these methods are advantageous as they do not require label to be externally provided. For example, a probabilistic decision making strategy is proposed for classifying objects into robot body part, human body part or manipulable objects [58].

There are also work that use incremental learning and propose end-to-end systems in the context of objects' learning through pointing gestures. For example, the robot recognizes or learns the objects pointed and labeled by a human based the similarity of their spin image descriptors [59]. However, its application requires the usage of skeleton tracking library for finding the 3D pointing direction. Furthermore, target objects need to be labeled by the human operator and hence requires supervision. In [50], pointing direction is found without tracking.

However the estimation is done in 2D which can possibly lead to wrong target detection and thus low recognition accuracy. Hence, both learning and recognition are done with small variations in objects' appearances.

4.3. Method

The proposed approach consists of the following steps:

- First, the target point cloud object is encoded internally using the Deformable Sphere Approximation (DSA) Descriptor [3].
- Next, it is associated with the existing objects' memory.
- In case of no association, the robot learns the pointed object through getting N_s different views of the object as pointed by the human and then adding the resulting knowledge to its objects' memory.

4.3.1. Deformable Sphere Approximation Descriptor

The DSA descriptor is based on the approximation of double trigonometric Fourier series (DTFS) representation of the point cloud data. The first step is to encode the original object cloud $\mathcal{D}_o \subset R^3$ data via deforming a sphere with radius ρ_0. Following, the coefficients of the double trigonometric series representation of the deformed sphere are found. The DSA descriptor is obtained from the rotational invariants of these coefficients [60]. In the rest of this section, the descriptor is briefly explained for completeness. For details, the interested reader is kindly referred to [3].

4.3.2. Deformed Sphere Mapping

The first step is to map the object cloud data is mapped to a deformed sphere. For this, the object's point cloud data $\mathcal{D}_o$ is the transformed so that its origin is at μ_o - the mean of $\mathcal{D}_o$. The transformed data is then used to deform a 2-sphere S^2 with radius ρ_0.

For each point $p \in \mathcal{D}_o$, define its spherical coordinates as $f(p) = [f_1(p) \quad f_2(p)]^T$ with $f_1(p) \in [-\pi, \pi]$ and $f_2(p) \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ being pan and tilt angles respectively and $r(p)$ is the radial distance. The deformation map $\rho_o : S^2 \to R$ is defined as follows:

$$\rho_o(f) = \begin{cases} \rho_0 + r(p) & \exists p \in \mathcal{D}'_o \;\; s.t. \;\; f(p) = f \\ \rho_0. \end{cases} \tag{4.1}$$

The deformed sphere is then approximated using double trigonometric Fourier series:

$$\rho_o(f) = \sum_{h_1=0}^{H_1-1} \sum_{h_2=0}^{H_2-1} \lambda_{h_1 h_2} z_{o h_1 h_2}(c) e_{h_1 h_2}(f) \tag{4.2}$$

where H_1 and H_2 denotes the number of harmonics and the parameters $\lambda_{h_1 h_2}$ are calculated as

$$\lambda_{h_1 h_2} = \begin{cases} 0.25 & if \;\; h_1 = 0, h_2 = 0 \\ 0.5 & if \;\; h_1 > 0, h_2 = 0 \quad or \quad h_1 = 0, h_2 > 0 \\ 1 & if \;\; h_1 > 0, h_2 > 0. \end{cases} \tag{4.3}$$

For each (h_1, h_2) pair, $e_{h_1 h_2}(f) \in R^4$ is of the form :

$$e_{h_1 h_2}(f) = \begin{bmatrix} cos(h_1 f_1) cos(2 h_2 f_2) \\ sin(h_1 f_1) cos(2 h_2 f_2) \\ cos(h_1 f_1) sin(2 h_2 f_2) \\ sin(h_1 f_1) sin(2 h_2 f_2) \end{bmatrix}. \tag{4.4}$$

The vector $z_{o h_1 h_2} \in R^4$ is defined as:

$$z_{o h_1 h_2} = \frac{2}{\pi^2} \begin{bmatrix} \int_{-\pi}^{\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \rho_o(f) cos(h_1 f_1) cos(2 h_2 f_2) df_1 df_2 \\ \int_{-\pi}^{\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \rho_o(f) sin(h_1 f_1) cos(2 h_2 f_2) df_1 df_2 \\ \int_{-\pi}^{\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \rho_o(f) cos(h_1 f_1) sin(2 h_2 f_2) df_1 df_2 \\ \int_{-\pi}^{\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \rho_o(f) sin(h_1 f_1) sin(2 h_2 f_2) df_1 df_2 \end{bmatrix}. \tag{4.5}$$

Finally, the DSA descriptor $I_o \in R^{H_1 H_2}$ is constructed as

$$I_{oh_1h_2} = z^T_{oh_1h_2} z_{oh_1h_2}. \tag{4.6}$$

4.3.3. Recognition & Objects' Memory

The robot first attempts to recognize the target object. This is done via forming its DSA descriptor and then relating it with its objects' memory. The robot retains its knowledge of objects' classes in its objects' memory. The objects' memory is an hierarchically organized structure with terminal nodes corresponding to distinct objects. All nodes except the leaf nodes are associated with a set of support vector machine (SVM) classifiers pertaining the node's children nodes. As such, the objects' memory enables the robot to associate the incoming objects data with its existing knowledge.

Memory association is done by starting at the root note and traversing down the hierarchy. At each node $\mathcal{N}$ at each level, the robot uses the SVM classifiers associated with that node as to decide which node to proceed to at the next level. It chooses to move to the node $\mathcal{N}' \in C(\mathcal{N})$ among the children $C(\mathcal{N})$ nodes of node $\mathcal{N}$ with the highest classifier response $d(\mathcal{N}')$ - pending that its classification response is sufficiently high:

$$\mathcal{N}' \in \arg \max_{\mathcal{N}^c \in C(\mathcal{N})} d(\mathcal{N}^c) \qquad d(\mathcal{N}) \geq \tau_r. \tag{4.7}$$

This is repeated until either a terminal node is reached or the classifier responses of all children nodes are low. In the latter case, the object is deemed to be 'unrecognized'. In that case, the robot goes into learning mode in order to learn the target object.

4.3.4. Object Learning

In learning, the robot then assumes that the human will point to N_s different views of the same object. As such, its learning set will consist of a set of object's appearances.

Typically, N_s tends to vary between 18 and 26. The robot uses an incremental agglomerative clustering method SLINK algorithm to construct its objects' memory [61]. When evolving the memory, the robot can consider two alternatives in regards to each new and unknown object class. It can be placed as a single node or alternatively as a set of clusters. In the former, all samples of a class are learned together. In the second method, the robot learns each class as a set of N_c subclasses - each corresponding to one different canonical view of the object. The robot determines these views on its own using again a hierarchical clustering method [62].

4.4. Experimental Results

In this section, we present the results of object cognition experiments. In these experiments, it is assumed that there is only one hand present in the scene and it is performing a pointing gesture. In all the results, the robot is completely on its own with respect to the first two stages - namely finding the pointing direction and then finding the target objects. Some of the pointed classes are shown in Figure 4.1. When learning, the human points the object with different views and the robot applies incremental learning using the data provided. The number of samples per class is $N_s \in \{18, 28\}$. The number of subcategories is $N_c = 3$. ν-type SVM is used. In the direct approach we set $\nu = 0.1$. In the sub-categorical approach, since inter-class variance is smaller compared to the direct approach, a smaller $\nu = 0.05$ value is used during training. For recognition, the SVM classifiers are subjected to threshold $\tau_r = 0.7$.

4.4.1. Manually Switched Learning & Recognition Modes

In the experiments, the robot's operation mode is manually first set to be learning and then recognition. In the learning mode, a human points to each object from different perspectives and the robots incrementally evolves its objects' memory. With each new class, its structure is updated and SVM classifiers associated with the changed nodes are then re-trained. The resulting objects' memory is as shown in Figure 4.2.

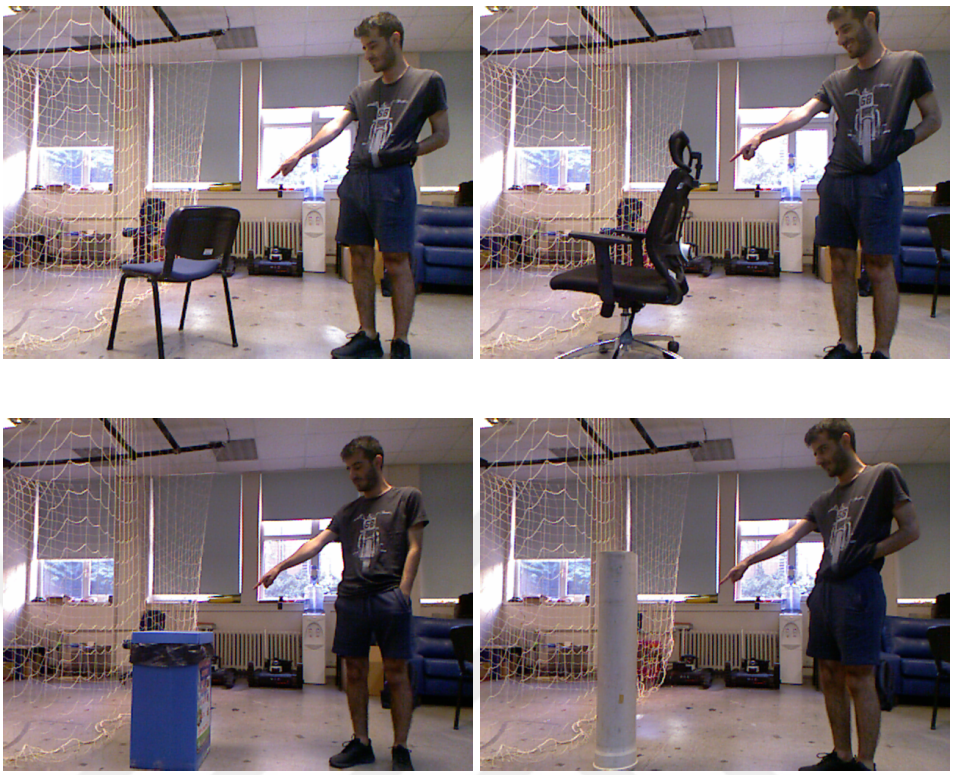

Figure 4.1. Some Objects Learned by Pointing Gestures.

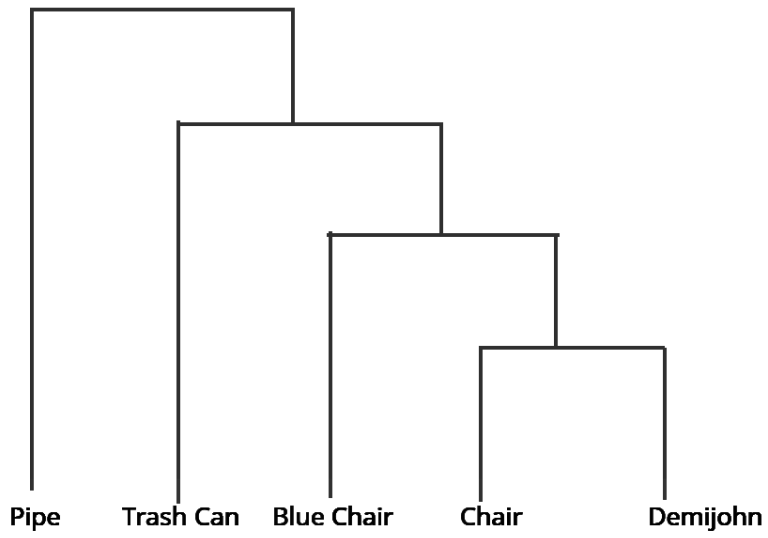

Figure 4.2. Objects' Memory Hierarchy After Learning 5 Objects.

The robot's operation is then switched to the recognition mode. In this mode, the robot views a scene in which a human points to an object and makes a recognition decision.

This classification is correct if the pointed object is both determined and classified correctly. Recognition performance is evaluated considering both one-class and subclasses approaches. Prior to recognition performance tests, learning validation is done using learning samples. The results are shown in Table 4.2 for both approaches. The object classes shown by O_i represent following classes in order: Blue chair, chair, trash can, pipe, demijohn.

Following, recognition performance in new scenes is evaluated. The results are presented in Table 4.3. It is observed that average accuracy is around 44% with no subclasses while it is slightly lower (39.8%) with subclasses. In both, worst performance occurs with objects O_1 and O_2. We attribute this to two reasons: First, their appearances change completely when looked from different perspectives compared to other objects in the data set. Hence, the learning set needs to be larger. Secondly, the segmentation of these objects tend to problematic. For example, the upper part of the blue chair is sometimes found as a different object as seen in Figure 4.3. As a result, learning is harder for blue chair objects. In general, classification without subclasses is observed to perform better that with subclasses. This is not surprising since we are dealing with a small amount of data. As we divide classes into sub-categories, number of instances in each cluster get smaller, which makes learning task harder. If we had more data for each object, subclasses approach would be expected to have better performance. The success of the algorithm on objects likeO_3 and O_4 can could be explained by their simpler geometry with appearances not changing much when looked from different perspectives. Hence a small number of samples suffices for reliable learning. Finally, it should be noted that while overall performance around 40% needs to be improved, it is still much better than around 10% average accuracy reported in [50].

In order to increase number of test samples, we used a testing method similar to k-fold cross validation. Samples of the each class in the test data is split into 2 data sets. In total we have 3 sets, 2 sets coming from original test data and our original training data. In each of the 3-fold cross validation test phases, one of the 3 sets are used to train the robot while the remaining two is used for testing. Average results of 3 testing phases are give in Table 4.1.

Overall, the success of the one-class approach is close to the success of the algorithm when original training set is used. However, overall success of the subclasses approach increased from 39.8% to 47.4%. This might be related to the fact that the data distributions of the 2 sets coming from original test set are similar.

Table 4.1. 3-fold cross validation average results.

	No subclasses	**Subclasses**
Class	**Accuracy**($\%$)	**Accuracy** ($\%$)
O_1	39.6	36.8
O_2	26.1	19.2
O_3	61.8	60.8
O_4	72.7	93.2
O_5	29.3	26.9
Overall	45.9	47.4

Table 4.2. Learning validation.

		No subclasses		**Subclasses**	
Class	**# Scenes**	**# Correct Recognitions**	**Accuracy**($\%$)	**# Correct Recognitions**	**Accuracy** ($\%$)
O_1	24	24	100	20	83.3
O_2	26	25	96.2	22	84.6
O_3	24	24	100	23	95.6
O_4	21	21	100	21	100
O_5	18	8	100	8	100

Finally, in order to isolate the performance of correct target detection with that recognition, we also consider recognition performance with only correctly found target objects. The results are shown in Table 4.4. The performance with O_5 object is observed to increase significantly.

Table 4.3. Recognition performance.

		No subclasses		**Subclasses**	
O_1	45	10	22.2	6	13.3
O_2	46	7	15.2	4	8.7
O_3	35	17	48.5	22	62.9
O_4	38	34	89.8	34	89.8
O_5	23	8	34.8	8	34.8
Unknown	25	10	40.0	13	52.0
Overall	206	91	44.1	82	39.8

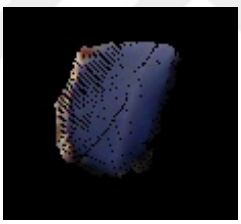

Figure 4.3. Under-segmentation of Blue Chair.

Table 4.4. Test data results when pointed object is found correctly.

		No subclasses		**Subclasses**	
Class	**# Scenes**	**# Correct Recognitions**	**Accuracy(%)**	**# Correct Recognitions**	**Accuracy (%)**
O_1	40	10	25	6	15
O_2	40	7	17.5	4	10
O_3	28	22	78.9	17	60.1
O_4	34	34	100	34	100
O_5	11	8	72.3	8	72.3
Unknown	19	10	52.6	13	68.4
Overall	172	91	52.9	82	47.6

5. CONCLUSION

This thesis provides an end-to-end approach to robot learning through human-robot interaction. Human pointing gestures are chosen as a means to communicate with the robot as they are naturally used between humans as well. The proposed approach consists of three stages: 3D pointing direction estimation, target object detection and object learning. The 3D pointing direction estimation is accomplished using a novel approach. Differing from previous work, we do not require any special background or the detectability of specific human parts. Rather, the only assumption is that the person performing the pointing gesture should be withing the coverage of the RGB-D sensor. Experimental results that are conducted with a mobile robot endowed with a RGB-D camera in a wide-ranging scenarios demonstrate that exhibits both accuracy and robustness to variations in the hand proximity and pointing directions. As part of this work, we also release a new RGB-D data set of with precise 3D pointing direction annotations that can be used as a benchmark. As part of pointing direction estimation, a pointing gesture classifier is trained. It classification accuracy is evaluated on a new data set of pointing gestures. For target object detection, a score-based approach is proposed - similar to related work. However, differing from them, the score depends on a variety of different criteria. Experiments on a benchmark data set show that the accuracy of the proposed method is comparatively higher than those of previous work. Finally, the robot either recognizes or learns the pointed object as necessary. This is done in three steps. First, the robot encodes the respective point cloud data using deformed sphere approximation representation. Then, it associates this descriptor with the knowledge existing in its objects' memory through traversing down the memory hierarchy and using the classifiers associated with the reached node to decide on which child node to move to. This is repeated until either classifiers' responses are all low or or a terminal node is reached. The former case implies no recognition and the robot expects human to show the object from different viewpoints as to form its knowledge of this object. It then adds this to its existing objects' memory using an incremental hierarchical clustering method.

For future work, information from successive frames can be utilized to improve the pointing direction detection accuracy through temporal processing. Also, the pointing direction data set can be expanded accordingly. The pointing gesture classification network is successful in learning the given data but our training set does not fully cover real-world scenarios. New data would need to be added to training set to include more scenarios. Object learning results need to be improved for objects whose view change substantially when perspective changes. To do so, sub-categorical approach should be used. Robot will need to acquire more data from the object for a robust learning in the sub-categorical approach. This could be achieved by defining a second object from the same class. That is, algorithm could be improved so that robot scans two objects pointed by human to learn a class. Another extension would be addition of other human related cues - such as natural language labels during learning. Right now, the robot recognizes an object, but does not have a label or name for it. If the robot is made to recognize human language, then communication and learning can both become easier.

REFERENCES

1. Kita, S., *Pointing: Where language, Culture and Cognition Meet*, Psychology Press, 2003.

2. Bochkovskiy, A., C.-Y. Wang and H. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection", *ArXiv*, Vol. abs/2004.10934, 2020.

3. Durukan, M., *Depth-Based Scene Mapping Through Spatio-Temporal Knowledge Integration*, Master's Thesis, Bogazici University, 2021.

4. Narasimhaswamy, S., Z. Wei, Y. Wang, J. Zhang and M. Hoai, "Contextual Attention for Hand Detection in the Wild", *International Conference on Computer Vision*, 2019.

5. Azari, B., A. Lim and R. Vaughan, "Commodifying Pointing in HRI: Simple and Fast Pointing Gesture Detection from RGB-D Images", *16th Conference on Computer and Robot Vision*, pp. 174–180, 2019.

6. Barbed, O. L., P. Azagra, L. Teixeira, M. Chli, J. Civera and A. C. Murillo, "Fine Grained Pointing Recognition for Natural Drone Guidance", *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 4480–4488, 2020.

7. Wei Wang and Jing Pan, "Hand Segmentation Using Skin Color and Background Information", *2012 International Conference on Machine Learning and Cybernetics*, Vol. 4, pp. 1487–1492, 2012.

8. Dawod, A. Y., J. Abdullah and M. J. Alam, "Adaptive Skin Color Model for Hand Segmentation", *2010 International Conference on Computer Applications and Industrial Electronics*, pp. 486–489, 2010.

9. Nickel, K. and R. Stiefelhagen, "Visual Recognition of Pointing Gestures for Human-Robot Interaction", *Image and Vision Computing*, Vol. 25, pp. 1875–1884, 2007.

10. Eng-Jon Ong and R. Bowden, “A Boosted Classifier Tree for Hand Shape Detection”, *Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004. Proceedings.*, pp. 889–894, 2004.

11. Kolsch, M. and M. Turk, “Robust Hand Detection”, *Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004. Proceedings.*, pp. 614–619, 2004.

12. Karlinsky, L., M. Dinerstein, D. Harari and S. Ullman, “The Chains Model for Detecting Parts by Their Context”, *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 25–32, 2010.

13. Kumar, M. P., A. Zisserman and P. H. S. Torr, “Efficient Discriminative Learning of Parts-Based Models”, *2009 IEEE 12th International Conference on Computer Vision*, pp. 552–559, 2009.

14. Buehler, P., M. Everingham, D. Huttenlocher and A. Zisserman, “Long Term Arm and Hand Tracking for Continuous Sign Language TV Broadcasts”, *Proceedings of the British Machine Vision Conference*, pp. 110.1–110.10, BMVA Press, 2008, doi:10.5244/C.22.110.

15. Arpit Mittal, A. Z. and P. Torr, “Hand detection using multiple proposals”, *Proceedings of the British Machine Vision Conference*, pp. 75.1–75.11, BMVA Press, 2011.

16. Le, T. H. N., Y. Zheng, C. Zhu, K. Luu and M. Savvides, “Multiple Scale Faster-RCNN Approach to Driver’s Cell-Phone Usage and Hands on Steering Wheel Detection”, *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 46–53, 2016.

17. Roy, K., A. Mohanty and R. R. Sahay, “Deep Learning Based Hand Detection in Cluttered Environment Using Skin Segmentation”, *2017 IEEE International Conference on Computer Vision Workshops*, pp. 640–649, 2017.

18. Deng, X., Y. Zhang, S. Yang, P. Tan, L. Chang, Y. Yuan and H. Wang, “Joint Hand Detection and Rotation Estimation Using CNN”, *IEEE Transactions on Image Processing*, Vol. 27, pp. 1888–1900, 2018.

19. Le, T., K. G. Quach, C. Zhu, C. N. Duong, K. Luu and M. Savvides, “Robust Hand Detection and Classification in Vehicles and in the Wild”, *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1203–1210, 2017.

20. Yang, L., Z. Qi, Z. Liu, H. Liu, M. Ling, L. Shi and X. Liu, “An Embedded Implementation of CNN-Based Hand Detection and Orientation Estimation Algorithm”, *Machine Vision and Applications*, pp. 1–12, 2019.

21. Xu, C., W. Cai, Y. Li, J. Zhou and L. Wei, “Accurate Hand Detection from Single-Color Images by Reconstructing Hand Appearances”, *Sensors (Basel, Switzerland)*, Vol. 20, 2020.

22. Li, G., H. Tang, Y. Sun, J. Kong, G. Jiang, D. Jiang, B. Tao, S. Xu and H. Liu, “Hand Gesture Recognition Based on Convolution Neural Network”, *Cluster Computing*, pp. 1–11, 2017.

23. Chung, H.-Y., Y.-L. Chung and W.-F. Tsai, “An Efficient Hand Gesture Recognition System Based on Deep CNN”, *2019 IEEE International Conference on Industrial Technology*, pp. 853–858, 2019.

24. Dong, X., Y. Xu, Z. Xu, J. Huang, J.-D. Lu, C. Zhang and L. Lu, “A Static Hand Gesture Recognition Model Based on the Improved Centroid Watershed Algorithm and a Dual-Channel CNN”, *2018 24th International Conference on Automation and Computing*, pp. 1–6, 2018.

25. Islam, M. Z., M. S. Hossain, R. ul Islam and K. Andersson, “Static Hand Gesture Recognition Using Convolutional Neural Network with Data Augmentation”, *2019 Joint 8th International Conference on Informatics, Electronics & Vision and 2019 3rd Interna-*

tional Conference on Imaging, Vision & Pattern Recognition, pp. 324–329, 2019.

26. Köpüklü, O., A. Gunduz, N. Kose and G. Rigoll, "Real-time Hand Gesture Detection and Classification Using Convolutional Neural Networks", *2019 14th IEEE International Conference on Automatic Face & Gesture Recognition*, pp. 1–8, 2019.

27. Wu, X., "A Hand Gesture Recognition Algorithm Based on DC-CNN", *Multimedia Tools and Applications*, Vol. 79, pp. 9193–9205, 2019.

28. Tang, H., H. Liu, W. Xiao and N. Sebe, "Fast and Robust Dynamic Hand Gesture Recognition via Key Frames Extraction and Feature Fusion", *Neurocomputing*, Vol. 331, pp. 424–433, 2019.

29. Shukla, D., Ö. Erkent and J. Piater, "Probabilistic Detection of Pointing Directions for Human-Robot Interaction", *2015 International Conference on Digital Image Computing: Techniques and Applications*, pp. 1–8, 2015.

30. Fujita, D. and T. Komuro, "Three-Dimensional Hand Pointing Recognition Using Two Cameras by Interpolation and Integration of Classification Scores", *ECCV Workshops*, 2014.

31. Jaiswal, S., P. Mishra and G. C. Nandi, "Deep Learning Based Command Pointing Direction Estimation Using a Single RGB Camera", *2018 5th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering*, pp. 1–6, 2018.

32. Bhuyan, M. K., D. R. Neog and M. K. Kar, "Fingertip Detection for Hand Pose Recognition", *International Journal of Advanced Trends in Computer Science and Engineering*, Vol. 4, No. 3, pp. 501–511, 2012.

33. Huang, Y., X. Liu, X. Zhang and L. Jin, "A Pointing Gesture Based Egocentric Interaction System: Dataset, Approach and Application", *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 370–377, 2016.

34. Wu, W., C. Li, Z. Cheng, X. Zhang and L. Jin, "YOLSE: Egocentric Fingertip Detection from Single RGB Images", *2017 IEEE International Conference on Computer Vision Workshops*, pp. 623–630, 2017.

35. Jain, V., G. Garg, R. Perla and R. Hebbalaguppe, "GestARLite: An On-Device Pointing Finger Based Gestural Interface for Smartphones and Video See-Through Head-Mounts", *ArXiv*, Vol. abs/1904.09843, 2019.

36. Hu, K., S. J. Canavan and L. Yin, "Hand Pointing Estimation for Human Computer Interaction Based on Two Orthogonal-Views", *International Conference on Pattern Recognition*, pp. 3760–3763, 2010.

37. Das, S. S., "Precise Pointing Direction Estimation using Depth Data", *2018 27th IEEE International Symposium on Robot and Human Interactive Communication*, pp. 202–207, 2018.

38. Droeschel, D., J. Stückler and S. Behnke, "Learning to interpret pointing gestures with a time-of-flight camera", *2011 6th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 481–488, 2011.

39. Van den Bergh, M., 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", *2011 International Workshop on Robot and Human Communication*, pp. 357–362, 2011.

40. Abidi, S., M. Williams and B. Johnston, "Human Pointing as a Robot Directive", *2013 8th ACM/IEEE International Conference on Human-Robot Interaction*, pp. 67–68, 2013.

41. Fernandez, A., L. Bergesio, A. Bernardos, J. Besada and J. Casar, "A Kinect-Based System to Enable Interaction by Pointing in Smart Spaces", *2015 IEEE Sensors Applications Symposium*, pp. 1–6, 2015.

42. Wong, N. and C. Gutwin, "Where are you pointing?: the accuracy of deictic pointing

in CVEs", *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2010.

43. Rahman, A., J. A. Mahmud and M. Hasanuzzaman, "Pointing and Commanding Gesture Recognition in 3D for Human-Robot Interaction", *International Conference on Innovation in Engineering and Technology*, pp. 1–10, 2018.

44. Redmon, J., "Darknet: Open Source Neural Networks in C", `http://pjreddie.com/darknet/`, 2013–2016, last accessed on 15/6/21.

45. "COCO Data Set", `https://cocodataset.org/#home`, 2020, last accessed on 05/3/21.

46. Alani, A. A., G. Cosma, A. Taherkhani and T. McGinnity, "Hand Gesture Recognition Using an Adapted Convolutional Neural Network with Data Augmentation", *2018 4th International Conference on Information Management*, pp. 5–12, 2018.

47. Arun, K. S., T. Huang and S. D. Blostein, "Least-Squares Fitting of Two 3-D Point Sets", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-9, pp. 698–700, 1987.

48. Leavens, D. A. and W. Hopkins, "The Whole-Hand Point: the Structure and Function of Pointing from a Comparative Perspective", *Journal of Comparative Psychology*, Vol. 113, No. 4, pp. 417–425, 1999.

49. Azagra, P., F. Golemo, Y. Mollard, M. Lopes, J. Civera and A. C. Murillo, "A Multimodal Dataset For Object Model Learning From Natural Human-Robot Interaction", *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 6134–6141, 2017.

50. Azagra, P., J. Civera and A. C. Murillo, "Incremental Learning of Object Models From Natural Human–Robot Interactions", *IEEE Transactions on Automation Science and Engineering*, Vol. 17, pp. 1883–1900, 2020.

51. Hosoya, E., H. Sato, M. Kitabata, I. Harada, H. Nojima and A. Onozawa, “Arm-Pointer: 3D Pointing Interface for Real-World Interaction”, *ECCV Workshop on HCI*, 2004.

52. Zermas, D., I. Izzat and N. Papanikolopoulos, “Fast Segmentation of 3D Point Clouds: A Paradigm on LIDAR Data for Autonomous Vehicle Applications”, *2017 IEEE International Conference on Robotics and Automation*, pp. 5067–5073, 2017.

53. Kuznetsova, A., S. J. Hwang, B. Rosenhahn and L. Sigal, “Expanding Object Detector’s Horizon: Incremental Learning Framework for Object Detection in Videos”, *2015 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 28–36, 2015.

54. Yao, A., J. Gall, C. Leistner and L. Gool, “Interactive Object Detection”, *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3242–3249, 2012.

55. Triki, A., R. Aljundi, M. B. Blaschko and T. Tuytelaars, “Encoder Based Lifelong Learning”, *2017 IEEE International Conference on Computer Vision*, pp. 1329–1337, 2017.

56. Li, Z. and D. Hoiem, “Learning without Forgetting”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 40, pp. 2935–2947, 2018.

57. Camoriano, R., G. Pasquale, C. Ciliberto, L. Natale, L. Rosasco and G. Metta, “Incremental Robot Learning of New Objects with Fixed Update Time”, *2017 IEEE International Conference on Robotics and Automation*, pp. 3207–3214, 2017.

58. Lyubova, N., S. Ivaldi and D. Filliat, “From Passive to Interactive Object Learning and Recognition Through Self-Identification on a Humanoid Robot”, *Autonomous Robots*, Vol. 40, pp. 33–57, 2016.

59. Kasaei, S. H., M. Oliveira, G. H. Lim, L. Lopes and A. Tomé, “Interactive Open-Ended Learning for 3D Object Recognition: An Approach and Experiments”, *Journal of Intelligent & Robotic Systems*, Vol. 80, pp. 537–553, 2015.

60. Erkent, O. and H. I. Bozma, “Bubble Space and Place Representation in Topological

Maps", *The International Journal of Robotics Research*, Vol. 32, No. 6, pp. 671–688, 2013.

61. Sibson, R., "SLINK: An Optimally Efficient Algorithm for the Single-Link Cluster Method", *The Computer Journal*, Vol. 16, pp. 30–34, 1973.

62. Ward, J. H., "Hierarchical Grouping to Optimize an Objective Function", *Journal of the American Statistical Association*, Vol. 58, pp. 236–244, 1963.

63. Quigley, M., "ROS: an Open-Source Robot Operating System", *ICRA 2009*, 2009.

64. Bradski, G., "The OpenCV Library", *Dr. Dobb's Journal of Software Tools*, 2000.

65. Rusu, R. and S. Cousins, "3D is here: Point Cloud Library (PCL)", *2011 IEEE International Conference on Robotics and Automation*, pp. 1–4, 2011.

66. Shearer, J. M. and M. A. Wolfe, "ALGLIB, a Simple Symbol-Manipulation Package", *Communications of the ACM*, Vol. 28, pp. 820–825, 1985.

APPENDIX A: User's Guide

This chapter presents details of the developed robot system.

A.1. Hardware

The software is run on a Kobuki Turtlebot robot as shown in Figure A.1. Its sensory input is from a RGB-D Kinect sensor that is mounted on top of the robot. All the processing is done on the Nvidia Jetson Tx2 that the robot is equipped with.



Figure A.1. Turtlebot Robot Used in the Experiments.

A.2. Required Software

All the developed software is adapted to work in ROS framework [63]. The following software libraries are required:

- Ubuntu 18.04
- OpenCV [64] 4.2 or higher

- PCL library [65]
- cv_bridge with OpenCV 4
- ALGLIB [66]

A.3. Project Software

The header (*.h) files associated with this project are as follows:

- cloudmerging.h
- clusteringClasses.h
- detection.h
- ground_segmentation.h
- lidar_seg.h
- object_finder.h
- RGB_HSV.h
- util.h

The code (*.cpp) files are as follows:

- cloudmerging.cpp
- clusteringClasses.cpp
- detection.cpp
- ground_segmentation.cpp
- lidar_seg.cpp
- main.cpp
- object_finder.cpp
- RGB_HSV.cpp
- util.cpp

A.4. Usage

The associated processing can be invoked by executing the following commands on different terminals in order:

- $roslaunch\ freenect_launch\ freenect.launch$
- $rosrun\ object_finder\ find_object$

APPENDIX B: Permissions

Permission to use the images from MHRI data set [49] is given below in Figure B.1.

Permission

August 3, 2021

Multimodal Human-Robot Interaction (MHRI) [1] data set is a publicly available data set and researchers are permitted to use the images from it in their papers or theses if the reference is given.

Pablo Azagra Millan

AZAGRA MILLAN, PABLO (FIRMA)

Firmado digitalmente por AZAGRA MILLAN, PABLO (FIRMA)
Nombre de reconocimiento (DN): c=ES, serialNumber=/09222/BE, sn=AZAGRA, givenName=PABLO, cn=AZAGRA MILLAN, PABLO (FIRMA)
Fecha: 2021.08.05 09:25:01 +02'00'

References

[1] P. Azagra, F. Golemo, Y. Mollard, M. Lopes, J. Civera, and A. C. Murillo. A multimodal dataset for object model learning from natural human-robot interaction. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 6134–6141, 2017.

Figure B.1. MHRI data set permission