DEPTH-BASED SCENE MAPPING THROUGH SPATIO-TEMPORAL KNOWLEDGE INTEGRATION

by

Meriç Durukan

B.S., Mechatronics Engineering, Marmara University, 2016

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 Systems and Control Engineering

Boğaziçi University

2021

DEPTH-BASED SCENE MAPPING THROUGH SPATIO-TEMPORAL KNOWLEDGE INTEGRATION

DATE OF APPROVAL: 22.02.2021

ACKNOWLEDGEMENTS

First, I would like to express my gratitude to my supervisor, Prof. H.Işıl Bozma for all of her support, assistance, and encouragement during the thesis.

I would like to thank Prof. Yağmur Denizhan and Prof. Hakan Temeltaş for being a member of my thesis committee. Moreover, I would like to thank Prof. Ferit Öztürk for his contributions on the temporal knowledge accumulation part of my thesis.

This study has been supported in part by TUBITAK EEEAG-118E857.

I would like to express my thanks to Kemal Bektaş, Serhat İşcan, and Doğan Patar for their valuable support and assistance. I also would like to thank all previous members of ISL that I worked with before.

I also want to express my special thanks to Mehmet Salih Sakoğlu and Emre Hoş for their friendship and support.

Finally, I would like to thank my parents Hikmet Durukan and Ahmet Durukan, my brother Bekir Mert Durukan for their endless patience and support.

ABSTRACT

DEPTH-BASED SCENE MAPPING THROUGH SPATIO-TEMPORAL KNOWLEDGE INTEGRATION

This thesis is concerned with scene mapping by a mobile robot using point cloud data. It is a complex process that requires the robot to segment the incoming data, represent it compactly and efficiently, and then use the resulting knowledge in its learning and decision-making. Segmentation enables the robot to determine the point cloud object candidates. The robot bases its learning and reasoning on the detected segments. Range sensors, such as LIDAR, are essential for a robot to extract environmental information. However, they generally create sparse data. For this reason, the sparse data should be considered specially. A novel approach to segmentation is proposed based on an extension of density-based clustering in the spherical coordinate system. We present the deformable sphere approximation (DSA) descriptor as a novel 3D descriptor that encodes point cloud objects. Experimental results show that our representation method is capable of classifying the objects. Finally, we consider how the robot can use all knowledge available to it. We propose an approach in which the robot also considers the knowledge accumulated through tracking the objects' temporal continuity. For this, we propose the temporal deformable sphere approximation (T-DSA) descriptor. Its construction requires the robot to track object candidates. For this, we propose a novel multi-tracking approach based on combining Kalman Filtering and multi-object matching considering position and shape similarity. We then compare the various schemes the robot can use in order to utilize the resulting knowledge. Our experimental results show that the T-DSA descriptor improves the classification performance compared to only the instantaneous DSA descriptors. As such, the robot is able to build and evolve a scene map as it is navigating in it.

ÖZET

UZAMSAL-ZAMANSAL BİLGİ YARDIMIYLA DERINLİK TEMELLİ SAHNE HARİTALANDIRILMASI

Bu tezde, robotların nokta bulutu verilerini kullanarak haritalandırma yapması amaçlanmıştır. Bu zorlu işlem için gelen verilerin bölütlenmesi, kapsayıcı şekilde tanımlanması ve üretilen bilginin öğrenmede ve karar vermede kullanılması gerekmektedir. Bölütleme robota aday nesnelerin tanımlanmasını sağlar. Robot bu bilgileri öğrenme ve karar verme aşamalarında kullanır. LİDAR gibi derinlik algılayıcı sensörler robotların çevresel bilgi edinmeleri için önemlidir. Fakat, genellikle ayrık verili ortam taramaları üretirler. Bu yüzden, bu sensörlerin verilerinin işlenmesi özel olarak ele alınmalıdır. Bu çalışmada bölütleme işlemi için küresel koordinat düzleminde çalışacak yoğunluk esaslı bir yöntem önerilmiştir. Devamında, oluşan bölütleri betimleme amaçlı yamulmuş küre yaklaşıklık betimleyicisi önerilmiştir. Elde edilen deneysel sonuçlar tanımlayıcının nesneleri kategorilere ayırmada başarılı çalıştığı görülmüştür. Robot hareket ederken oluşan veri akışının anlık olarak değerlendirilmesi sahne anlamlandırmada çok önemli olsa da genellikle bu veriler üzerinden zamansal muhakeme yapılmaz. Fakat, robot hareketiyle oluşan bilgi akışında nesneler üzerinden bir devamlı olarak bir bilgi akışı gerçekleşmektedir. Bu bilgi akışını kullanmak adına zamansal yamulmuş küre yaklaşıklık betimleyicisi önerilmektedir. Nesnelerin takibi için Kalman filtreleme ve konum ve şekil benzerliğinin aynı anda kullanıldığı çoklu nesne eşleştirme yöntemi önerilmiştir. Böylece, robot hem anlık, hem de zamansal verileri kullanarak etrafındaki nesneleri tanıyabilmekte ve ortama ait anlambilimsel harita oluşturabilmektedir. Nesne sınıflandırmasına yönelik deneylerde, robotlarda zamansal yamulmuş küre yaklaşıklık betimleyicisinin anlık oluşturulmuş betimleyicilere göre performans artışı sağladığı gözlemlenmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

A_x	State transition matrix for Kalman filter
A_y	Observation transition matrix for Kalman filter
D	Point cloud set for segments
$\mathcal{D}'_o$	Centered point cloud
$\mathcal{D}_o$	Point cloud for an object
E	Total error for segmentation
$e_{h_1 h_2}(f)$	The vector of an orthonormal set of trigonometric functions
F_k	Kalman gain matrix
G	Similarity matrix
H_1	The number of first order harmonics
H_2	The number of second order harmonics
I_{co}	DSA descriptor of object o at position c
I^A_{co}	T-DSA descriptor of object o
k	Discrete time index
k_1	First discrete time index of set $\mathcal{K}_o$
k_N	Last discrete time index of set $\mathcal{K}_o$
$\mathcal{K}$	Set of discrete time indexes
$\mathcal{K}_o$	Index sequence of object o
l	Index for a segment
L	Index set of labels
M_1	Minimum neighbors for dense points
M_2	The number of the closest set for DSA computation resolution
$\mathcal{N}()$	Neighboring points
$\mathcal{O}$	Set of object candidates
O	Oversegmentation error
o	An object candidate
$\mathcal{P}$	Non ground point cloud data
p	A point in non ground point cloud data

Q_p	Maximum connectivity set
Q	Non-ground point cloud in spherical coordinates
Q_o	Point cloud of object o spherical coordinates
R	The set of real numbers
r_{ok}	Planar distance of robot to object o at time k
S^1	$[0, 2\pi]$
S^2	Two-dimensional spherical coordinate system
T_o	Track of object o
U	Undersegmentation error
$\mathcal{U}$	Classification model for DSA descriptor
$\mathcal{U}_A$	Classification model for T-DSA descriptor
$w_{omh_1h_2}$	mh_1h_2 th temporal vector of object o
x_{ok}	Kalman state of object o at discrete time k
$\hat{x}_{ok}$	Estimated Kalman state of object o at discrete time k
$\hat{y}_{ok}$	Observation of object o at discrete time k
$z_{h_1h_2}$	h_1, h_2 th coefficient of DSA descriptor
α_k	Heading at discrete time k
β_1	Position similarity measure
β_2	Shape similarity measure
$\Delta\mu_{ok}$	Mean change of object o at discrete time k
γ_1	Coefficient for pan angle threshold
γ_2	Coefficient for tilt angle threshold
δ_ψ	Tilt scan resolution
δ_ϕ	Pan scan resolution
μ_o	the mean points of a point cloud
ν_1	Weight of position similarity
ν_2	Weight of shape similarity
ψ	Tilt angle
ϕ	Pan angle
ρ	Depth value
ρ_{co}	Deformed sphere map

Σ_o	Covariance matrix of Q_o
τ_u	Threshold for undersegmentation error
τ_o	Threshold for oversegmentation error
τ_ρ	Threshold for depth value
τ_ψ	Threshold for tilt angle
τ_ϕ	Threshold for pan angle

LIST OF ACRONYMS/ABBREVIATIONS

3D	Three Dimensional
2D	Two Dimensional
CAD	Computer-Aided Design
DSA	Deformed Sphere Approximation
ISL	Intelligent Systems Laboratory
MLP	Multi Layer Perceptron
LIDAR	Light Detection And Ranging
RADAR	Radio Detection And Ranging
T-DSA	Temporal Deformed Sphere Approximation

1. INTRODUCTION

Mobile robots must be capable of understanding their surroundings in many tasks involving human interaction, navigation and mapping. Semantic parsing of objects constitutes a core part of this. For this reason, they need to identify the objects in their surroundings correctly. The environmental information is extracted from the scenes taken by sensors, such as camera, LIDAR, and RADAR. Range sensors such as Kinect-based sensors and LIDAR sensors provide three-dimensional (3D) point cloud data. As they enable mobile robots to sense the 3D world around them with high accuracy and range, most mobile robots are equipped with one or more of these sensors nowadays. However, semantic parsing of the collected point cloud data is a complex process. It requires solving three different problems: i) finding object candidates, ii) representing 3D objects, iii) interpreting the data collected over time. This thesis focuses on these three problems.

First, the problem of finding object candidates is considered. This requires the segmentation of the incoming 3D data. This is an integral part of the 3D scene understanding because it enables the robot to determine the objects of interest. As such, throughout the thesis, the words object, object candidate and (point cloud) segments are used interchangeably. As such, the reliability of segmentation highly affects the performance of the subsequent processes. Kinect-like sensors have a practical ranging limit of five meters and are used indoors. On the other hand, LIDAR sensors have maximum ranges of thirty to a hundred meters. As such, they are mostly used outdoors, although they are also suitable for indoor usage. Furthermore, the two sensors also vary concerning their data type - namely, Kinect generates spatially dense data while LIDAR generates sparse data. Thus, the segmentation algorithm should be capable of reliably determining the object candidates regardless of point cloud data density.

The next step is the representation of point cloud objects. The representation enables the robot to build, use and evolve its knowledge of objects internally. Thus,

it is critical to the successful completion of many robotic scene tasks involving human interaction. The point cloud objects can be dense or sparse depending on the sensor used. Hence, the representation should be usable with all point cloud objects regardless of its density.

The final step is the usage of the collected data. In most work, the descriptors are constructed from instantaneous data. However, the robot collects data continuously as it moving and thus, there is a temporal accumulation of data. While the temporal accumulation of data can also provide important information, this is not used in related work with few exceptions. In this work, both instantaneous and temporally accumulated data are considered. For this, the robot tracks all the detected object candidates. Then, it accumulates the information from each object and uses its object recognition-related reasoning.

1.1. Problem Statement

The thesis handles the problem in three stages:

(i) First, the object proposals should be found properly even with sparsely scanned 3D scenes. Moreover, most proposed works segment the sparse scenes by using user-defined parameters that depend on the scene's complexity.

(ii) Secondly, the robot needs to represent point cloud objects regardless of their density. As the robot can see the objects from different viewpoints and angles, if possible, the representation should be invariant to the robot's heading.

(iii) As a mobile robot moves, the incoming data has both an instantaneous and temporal aspect. The robot should be capable of using both in its reasoning.

1.2. Contribution

The major contributions of the thesis as follows:

(i) A novel approach is proposed to segment sparsely scanned 3D scenes. Unlike most works, our approach finds the segments in the spherical coordinate system. The parameters of the segmentation are selected considering the sensor's scan resolution.

(ii) A novel representation method is proposed for instantaneous object candidates' data. The descriptor is referred to as the Deformed Sphere Approximation (DSA) descriptor. The representation method works with various types of data-namely LIDAR, Kinect, and CAD data.

(iii) A novel representation method is proposed for temporally accumulated object candidates' data. This is achieved through the temporal extension of the DSA descriptor - namely temporal deformed sphere approximation (T-DSA) descriptor. As this requires the tracking of object candidates, a novel tracking algorithm is also presented.

1.3. Organization of Thesis

The organization of the thesis as follows:

- The proposed segmentation method is presented in Chapter 2. First, related literature is discussed. Following, the segmentation approach is presented. Lastly, extensive experimental results are discussed.
- The DSA descriptor is presented in Chapter 3. Again, the first related literature is explained. Next, the formulation of the descriptor is given. Finally, it is evaluated experimentally, including a comparative study with previous descriptors and various sensor data.
- Temporal accumulation of data regarding the object candidates is explained in Chapter 4. First, we explain the tracking of object candidates. Next, the formu-

lation of the T-DSA is explained. Finally, experimental results are discussed.

- In Chapter 5, the thesis concludes with a summary and future work.

2. SEGMENTATION FOR SPARSELY SCANNED 3D SCENES

This chapter focuses on the generation of 3D segments by a robot endowed with a LIDAR sensor. The segmentation is an essential first step in the standard perception pipeline associated with semantic parsing. This is because the robot can then use the resulting three-dimensional (3D) point cloud objects for categorizing and/or recognizing the 3D objects in its surroundings. Thus, the segmentation needs to be done without any prior information relating to their specific shapes or categories. This is a non-trivial problem due to the vast shape, size, and viewpoint variations amongst the objects - as is the case in typical work or home environments.

The sensors vary in regards to the density of point cloud data they provide. A point cloud data is considered to be dense if most scanned surfaces' connectivity can be captured with the connectivity of non-empty cells - considering a discretization of the world with a constant resolution [1]. Thus, range sensing based on 360° rotating scanners with 16, 32, or 64 beams such as LIDAR sensors provides sparse point cloud data. The focus in this chapter is to find possible object candidates even with sparse point cloud data without using class or category information.

The outline of the chapter is as follows: First, a summary of previous methods is presented in Section 2.1. The proposed algorithm is detailed in Section 2.2. Finally, experimental results are given in Section 2.3.

2.1. Related Literature

The proposed approaches can be categorized into two groups depending on whether they use dense or sparse data.

In the former case, the parameters of the dense-data algorithms are adjusted [2, 3]. These algorithms involve constant resolution grid-based models, and they yield poor results when used with sparse 3D LIDAR data. This is because the number of empty cells increases with sparser data, leading to objects being over-segmented. Consequently, distant objects are inclined to be over-segmented, while nearer objects can be wrongly merged. In fact, the sparsity and scattering of associated point cloud data are immensely affected by the distance. As an object's distance gets further, the associated point cloud data tends to be more sparse and scattered away from each other. Otherwise, objects get closer, and their point cloud data tends to be denser and closer to each other.

In the latter case, this problem is addressed by designing algorithms that specifically consider the sparse nature of the scan data. Commonly, the ground plane from point cloud data is removed either by assuming flat ground surfaces or non-flat ground surfaces [1,4–6]. The ground plane extraction is followed by segmenting the remaining data while using various interpolation schemes to fill the holes due to its sparsity.

The most common approach is to project 3D data onto a 2D plane - using either occupancy grids [7, 8] or LIDAR's scan line information [6]. The fact that determining the optimal grid dimensions has an important role in the segmentation performance. However, finding the optimal grid dimension is difficult as it depends on the scene. Furthermore, these approaches tend to miss the information on the vertical plane. Secondly, the segmentation is carried out by using clustering methods based on surface geometry [1, 9–14]. Here, the descriptive geometrical features are clustered by using variants of the nearest neighbor search. Moreover, Held et al. propose a learning-based method using spatial, semantic, and temporal knowledge to enhance the segmentation performance [15]. A third approach has been to consider the geometry of sense explicitly. Indeed, most works consider the cylindrical range image obtained by projecting the 3D data onto a cylinder whose axis is the rotational axis of the scanner so that the pixel values correspond to the distance measurements. This type of 3D data structure is called a range image. The advantage of using range images is that operating range

images is substantially faster than reasoning on the 3D point cloud. For instance, the connected components algorithm is applied on the cylindrical range image, and only components with angular extend higher than a given threshold are deemed to segment [16,17]. Finally, many network-based 3D object detection methods have been developed to find the object with their semantic information [18, 19]. They generate object proposal volumes before the calculation of semantic features. Although the methods are very efficient algorithms in detection, their success is highly dependent on the labeled data.

2.2. Density Based Spherical Segmentation

The proposed segmentation algorithm is based on a density-based clustering method. The segments are found by clustering dense adjacent points using non-ground scan points. The first step is to remove the ground data. To extract the ground information from the scene, we use the method of [6] due to its real-time applicability and simplicity. The implementation details and experimental results of the ground extraction are given in Appendix 5.

Let $\mathcal{P}$ refers to the non-ground point cloud data. The segmentation aims to associate an object candidate label with each point $p \in \mathcal{P}$. Let L denote the index set of labels. As such, we can determine the set of object candidates as indexed by the set $\mathcal{O}$ where each object candidate $o \in \mathcal{O}$ is determined by the point cloud data $\mathcal{D} \subset \mathcal{P}$ having the same label $l \in L$.

Once the ground data is removed, the proposed algorithm takes advantage of the fact that the LIDAR sensor's scanning geometry is defined in the spherical coordinate system with the origin at LIDAR optical center [20]. Interestingly, this representation corresponds to the raw data provided by the sensor and thus is available directly. The proposed method is motivated by the observation that dense regions in the spherical coordinate system correspond to 3D segments. Hence, the segments are determined by searching adjacent points with a density-based clustering method. To do this, we

propose a novel metric to find the adjacent points. Our method's main advantage is that the search parameters are determined by the scan parameters of the scan rather than the complexity of the scanned scene. Our extensive experimental results show that our approach finds the segments in the scene successfully.

Each point $p \in \mathcal{P}$ is associated with a pan $\phi \in [0, 2\pi)$ and tilt $\psi \in [0, \pi)$ and the associated depth reading $\rho(\phi, \psi)$. Let Q be the corresponding non-ground point cloud data expressed in the spherical coordinates. The fact that LIDAR scans the environment with constant resolutions. For example, with a Velodyne-VLP16 LIDAR, pan angle resolution is $\delta_\phi \in \{0.1^\circ, 0.2^\circ, 0.3^\circ, 0.4^\circ\}$ and tilt angle resolution is $\delta_\psi = 2^\circ$. Hence, the segments cannot be determined by searching adjacent points with a state-of-the-art density-based clustering method. This is because these approaches use a L_2-norm like metric to detect the neighboring points in Euclidean space. For this reason, we consider spherical coordinate system and introduce the ellipsoid distance metric that normalizes the pan, tilt and depth data considering different thresholds for each. Let these be denoted by $\tau_\phi, \tau_\psi, \tau_\rho > 0$, respectively. Then, for a point $q \in Q$ where $q = \begin{bmatrix} \phi & \psi & \rho(\phi, \psi) \end{bmatrix}^T$, its neighboring points $\mathcal{N}(q)$ is determined as in Equation 2.1:

$$\forall q' \in \mathcal{N}(q) \quad \frac{(\phi - \phi')^2}{\tau_\phi^2} + \frac{(\psi_a - \psi')^2}{\tau_\psi^2} + \frac{(\rho_a - \rho')^2}{\tau_\rho^2} \leq 1 \tag{2.1}$$

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

$$\tau_\phi = \gamma_1 \delta\phi \quad \tau_\psi = \gamma_2 \delta\psi \quad \tau_\rho = \delta\rho$$

where γ_i, $i = 1, 2$ are determined empirically.

The neighboring points $\mathcal{N}(q)$ of a given point need to satisfy a given density criterion - namely the cardinality of the neighboring points set $\mathcal{N}(q)$ should satisfy a minimum neighbor criterion as $|\mathcal{N}(q)| > M_1$. Here, $M_1 \in Z^{>0}$ corresponds to the

minimum number of neighbors. If the $\mathcal{N}(q)$ satisfies the minimum neighbor criterion, q is interpreted as a dense point. The dense point q is a core point for a new segment in S. After that, the segment is expanded by implementing this procedure for all found neighboring sets. Note that the parameter M_1 is a user-defined parameter that depends on selecting the threshold values. Namely, the value γ_1, γ_2 is an upper bound on the cardinality of $\|\mathcal{N}(q)\|$. Hence, M_1 is selected to be some percentage of this value. It is important to state that the minimum density (M_1) criterion is questioned for all queries, and it deals with under-segmentation and the noise in the LIDAR scan.

Moreover, the time complexity another essential issue for searching the points in the boundary ellipsoid volume because it can be very high if all non-ground scan points questioned for a query point. To cope with this issue, we used the indexed structure of the spherical point cloud. Indeed, the constant resolution parameters of the LIDAR scan enable us to index the point cloud according to its pan and tilt resolution. By doing this, we can eliminate irrelevant points. The points to consider are selected from the maximum connectivity set, Q_q. The maximum connectivity set includes possible connectivity distances of the neighboring points, and it is determined at the beginning of the segmentation. Note that if an element in Q_q does not satisfy Equation 2.2, it will not satisfy Equation 2.1. For this reason, the set Q_q is determined by using Equation 2.2. Pseudo-code for our segmentation approach is given in Figure 2.1.

$$\frac{(\phi - \phi')^2}{\tau_\phi^2} + \frac{(\psi - \psi')^2}{\tau_\psi^2} \leq 1 \tag{2.2}$$

Input: Non-ground point cloud in spherical coordinates(Q)

Output: Point wise cluster labels(L)

Parameters: $\tau_\psi, \tau_\phi, \tau_\rho$: threshold values

M_1: minimum number of points for a segment

$\delta\psi, \delta\phi$: scan resolutions

Initialization: $label \Leftarrow 0$: label for cluster

for $i = 1 : |Q|$ **do**

  Find Q_q using Equation 2.2: possible connectivity set

  if $label(q_i) \neq 0$ **then**

    continue

  end if

  $neighs \Leftarrow$ Search with Q_q and find neighs for q using Equation 2.1

  if $|neighs| > M_1$ **then**

    $label \Leftarrow label + 1$

    $label(q_i) \Leftarrow label$

    $it \Leftarrow 0$

    while $|neighs| = it$ **do**

      Find $Q_q{}'$ using Equation 2.2: possible connectivity set

      $expand \Leftarrow$ Search with $Q_q{}'$ and find neighs for q' using Equation 2.1

      if $|expand| > M_1$ **then**

        $label(q') \Leftarrow label$

        $neighs \Leftarrow$ Merge unique $expand$ with $neighs$

      end if

      $it \Leftarrow it + 1$

    end while

  else

    $label(q_i) \Leftarrow -1$: noise

  end if

end for

Figure 2.1. Pseudo-code for Density Based Spherical Segmentation

2.3. Experimental Results

Our proposed approach has been tested in three different scenerios :

(i) ISL scenes: We take two indoor LIDAR scans from our laboratory(ISL). In fact, the LIDAR is Velodyne VLP16, and it is integrated with a mobile robot. The segmentation labels are not available, so the evaluation of the scenes is carried out visually.

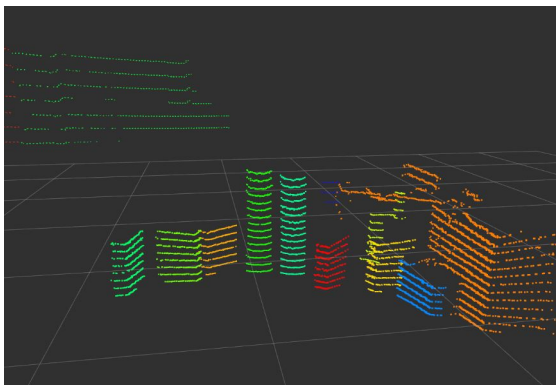
(ii) Velodyne Simulator: We test our method in Gazebo environment integrated with ROS. The fact that Velodyne Simulator enables us to simulate both Velodyne VLP-16 and Velodyne HDL-32E. As in ISL scenes, there is no segment information in the scenes, so we evaluate the results visually. Results quantitatively and test whether it creates under-segmentation or over-segmentation errors.

(iii) KITTI tracking dataset: Our segmentation approach is evaluated on the KITTI tracking dataset [21]. The tracking KITTI dataset was collected from a moving car on city streets to evaluate the perception system of autonomous driving. In the dataset, the point cloud data are taken from Velodyne HDL-64E LIDAR. Although point-wise labeling is not available, bounding boxes are adequate to test our approach [15]. The evaluation of the dataset is vital for our method. This is because the dataset paves the way for evaluating our approach.

2.3.1. ISL scenes

We generate two scenes from our lab. The objects in Figure 2.2 are selected from among the objects with simple geometry and positioned at nearly without any contact. However, in Figure 2.3, the scene is more difficult to segment because the scene contains an object such as a radiator, where proper depth perception can be difficult, and most objects are in contact with each other. Before the implementation of density-based segmentation, the ground is extracted from both scenes. The parameters for both scenes are given in Table 2.1.

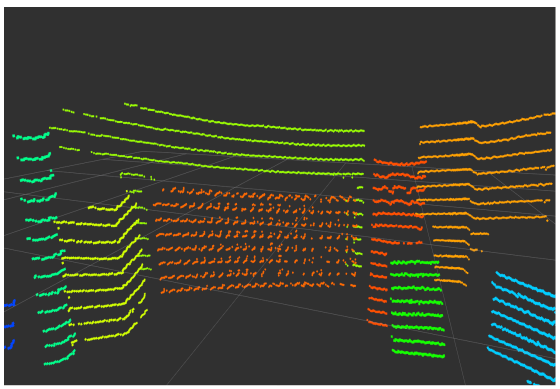
(a) Simple scene



(b) The segmentation of the simple scene

Figure 2.2. The segmentation of simple ISL scene

(a) Hard scene



(b) The segmentation of the hard scene

Figure 2.3. The segmentation of hard ISL scene

Table 2.1. The parameters for the segmentation of ISL scenes

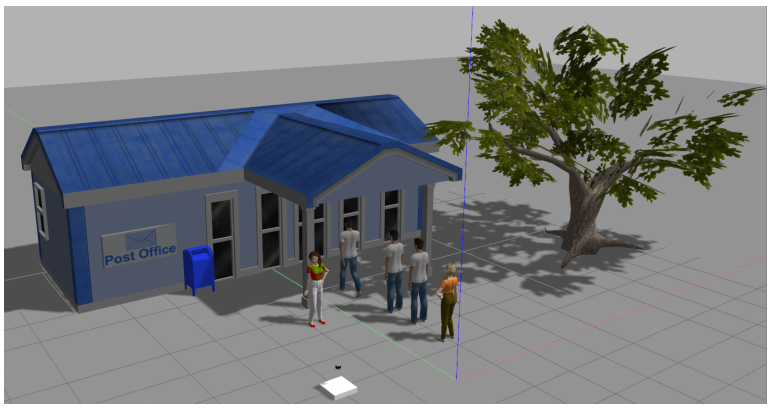
Segmentation parameters	ISL Scenes	
	Simple (Figure 2.2)	Hard (Figure 2.3)
τ_ϕ	4°	4°
τ_ψ	2°	2°
τ_ρ	$0.1m$	$0.1m$
M_1	15	15
Scan parameters		
δ_ϕ	2°	2°
δ_ψ	0.2°	0.2°

2.3.2. Velodyne Simulator

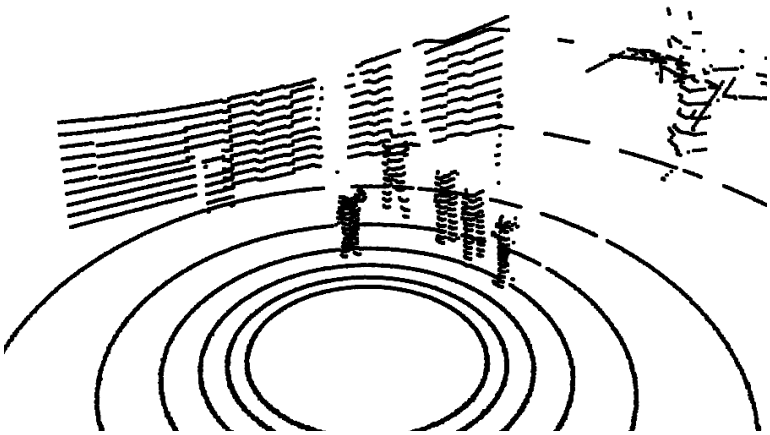
Velodyne Simulator includes two different LIDAR sensors, Velodyne VLP-16 and Velodyne HDL-32E. The most important differentiating feature of the two sensors is scanning resolution in tilt direction. Velodyne HDL-32E scan the environment with 1.33° resolution, but it is 2° for Velodyne VLP-16. We test our segmentation method in the simulator for both the Velodyne VLP-16 scan and the Velodyne HDL-32E scan. Since we test our segmentation method in an indoor environment (ISL), we construct two different outdoor environments to be segmented. For two scenes, the segmentation parameters are given in Table 2.2.

The first scene is taken with Velodyne VLP-16. The simulation view, the raw point cloud, and the segmented point cloud are given in Figure 2.4. In this scene, the proposed algorithm works well and segments all fully visible objects. However, some outdoor objects are not fully detectable for sparse LIDAR scenes. For example, the tree in our post office scene is sensed with a highly sparse point group even for the spherical coordinate system. Since our method expects dense points in the spherical coordinates system, the tree could not be segmented properly.

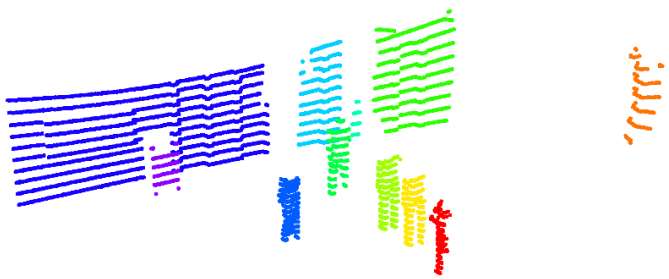
The second scene is taken with Velodyne HDL-32E. We experiment to gain an insight to work with the KITTI dataset. For this reason, we design a scene that contains possible street objects, such as pickup trucks, pedestrians, and waste containers. The simulation view, the raw point cloud, and segmented point cloud are given in Figure 2.5. It can be seen that the proposed method successfully segments the fully visible objects in the scene, i.e., between the pickup truck and the pedestrian on the left side in Figure 2.5b. It is an important state that in LIDAR scenes, some objects can obstruct other objects' view. Because the proposed approach needs to find a consistent geometrical relationship between points, this problematic issue can cause an over-segmentation error for non-fully visible objects.



(a) Simulation view

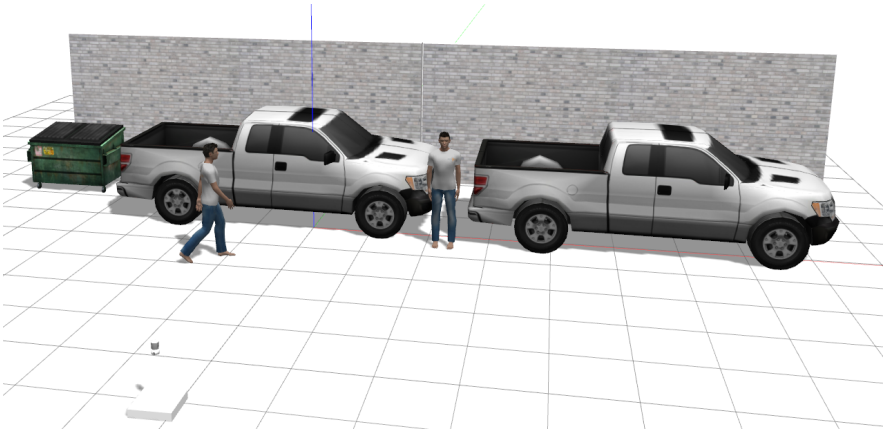


(b) Raw point cloud

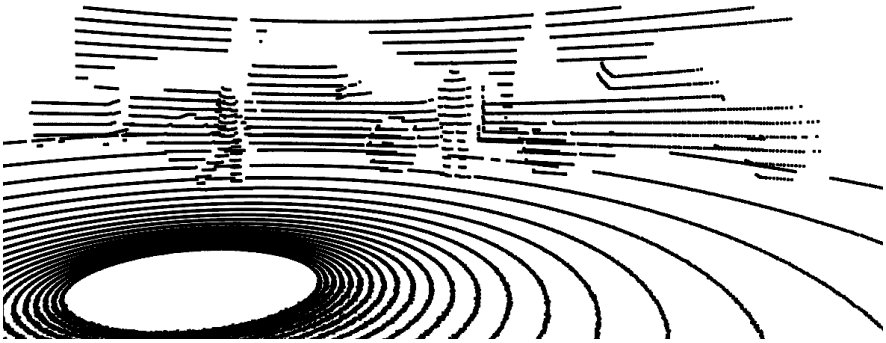


(c) Segmented point cloud

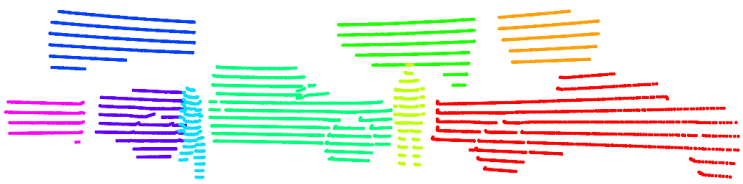
Figure 2.4. Post office scenario to test proposed approach



(a) Simulation view



(b) Raw point cloud



(c) Segmented point cloud

Figure 2.5. Street scenario to test proposed approach

Table 2.2. The parameters for the segmentation of Velodyne Simulator scenes

Segmentation parameters	Velodyne Simulator	
	Post Office (Figure 2.4)	Street (Figure 2.4)
τ_ϕ	6°	4°
τ_ψ	1°	1°
τ_ρ	$0.5m$	$0.5m$
M_1	20	20
Scan parameters		
δ_ϕ	2°	1.33°
δ_ψ	0.2°	0.2°

2.3.3. KITTI tracking dataset

We evaluate our approach using KITTI tracking dataset. The dataset consists of 21 sequential frames. For each scene, there are bounding boxes in the camera frame. We use nearly the same strategies of [15] to detect under-segmentation and over-segmentation errors. Let $\mathcal{O}^*$ denote the set of ground truth segments. For each detected object $o \in \mathcal{O}$, finding a segment that best matches with the ground truth are defined as:

$$o^* \in \underset{o' \in \mathcal{O}^*}{\operatorname{argmax}} |\mathcal{D}_o \cap \mathcal{D}^*_{o'}| \tag{2.3}$$

Let $\mathcal{D}_o$ denote the point cloud data of object o and $\mathcal{D}^*_{o'}$ denote ground truth object point cloud data. After finding the best matches, under-segmentation and over-segmentation errors can be calculated. Under-segmentation errors can be seen when the object is segmented with a close object. Over-segmentation errors can be defined as finding multiple segments instead of one segment. Defined metrics are defined as:

$$U = \frac{1}{|\mathcal{O}|} \sum_{(o,o^*)} \mathbb{1}\left(\frac{|\mathcal{D}_o \cap \mathcal{D}^*_{o^*}|}{\mathcal{D}_o} < \tau_u\right) \tag{2.4}$$

$$O = \frac{1}{|\mathcal{O}|} \sum_{(o,o^*)} \mathbb{1}\left(\frac{|\mathcal{D}_o \cap \mathcal{D}^*_{o^*}|}{\mathcal{D}^*_{o^*}} < \tau_o\right) \tag{2.5}$$

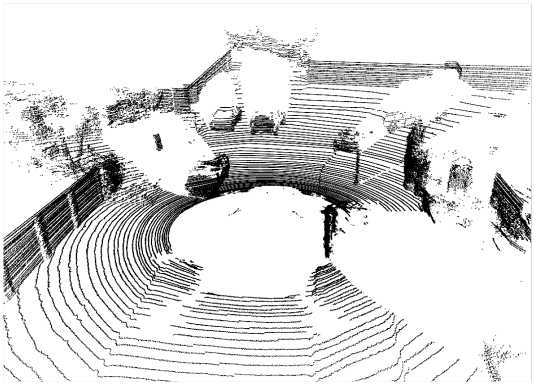
Where, $\mathbb{1}$ is defined as an indicator function that is equal to 1 when the given input is true, otherwise 0. τ_u and τ_o are defined under-segmentation and over-segmentation errors, respectively. As stated in [15], over-segmentation is a more problematic issue than under-segmentation because over-segmentation errors cause the creation of false objects. However, from our observation, the bounding boxes do not fully cover the whole segments in the 3D scene because of the bounding boxes' frame. For this reason, we set τ_o to 0.9 instead of 1.0, and we set τ_u to 0.5. Moreover, the summation of the two errors can be defined as the overall error rate (E) metric for the segmentation.

We test our segmentation approach using different parameter values for $\tau_\phi, \tau_\psi, \tau_\rho$, M_1. Results and used parameters are given in Table 2.3. The best segmentation results are obtained for $\tau_\phi = 1.44°, \tau_\psi = 2.4°, \tau_\rho = 1.0m, M_1 = 10$. It is essential to state that the scans in the KITTI dataset are taken with Velodyne HDL-64E, and its resolution parameters are $\delta_\phi = 0.08°, \delta_\psi = 0.4°$. We select τ_ϕ, τ_ψ values according to scan resolutions. We select τ_ρ and M_1 values based on our visual experiments. Table 2.3 shows how the change in parameters affects the segmentation performance. As seen in the table, the low values for τ_ϕ , τ_ψ , τ_ρ create low under-segmentation errors; however, these values increase in over-segmentation errors. Hence, the optimal parameters for the segmentation should create low total error and low over-segmentation error from our perspective.

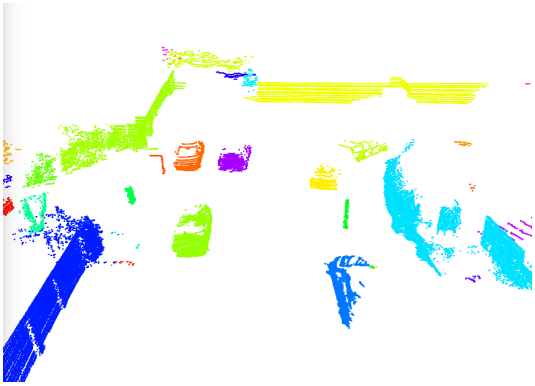
Table 2.3. KITTI tracking results

Parameters				Results			
$\tau_\phi(^\circ)$	$\tau_\psi(^\circ)$	$\tau_\rho(m)$	M_1	U	O	E	Average Time(ms)
0.96	1.6	1.0	10	0.2116	0.4296	0.6212	18.58
0.96	1.6	1.0	15	0.1780	0.7403	0.9183	19.47
0.96	1.6	1.5	10	0.2667	0.3694	0.6163	18.35
0.96	1.6	1.5	15	0.2284	0.6836	0.9120	19.00
0.96	2.4	1.0	10	0.2402	0.2178	0.4580	33.91
0.96	2.4	1.0	15	0.2316	0.2853	0.5169	34.39
0.96	2.4	1.5	10	0.3056	0.1671	0.4727	33.80
0.96	2.4	1.5	15	0.2942	0.2264	0.5206	34.00
1.44	1.6	1.0	10	0.2694	0.1859	0.4553	34.96
1.44	1.6	1.0	15	0.2504	0.2656	0.5160	34.86
1.44	1.6	1.5	10	0.3375	0.1407	0.4782	34.08
1.44	1.6	1.5	15	0.3167	0.2075	0.5242	34.21
1.44	2.4	1.0	10	0.2805	0.1475	0.4280	39.20
1.44	2.4	1.0	15	0.2607	0.2024	0.4631	39.38
1.44	2.4	1.5	10	0.3473	0.1065	0.4538	38.52
1.44	2.4	1.5	15	0.3333	0.1507	0.4840	38.82

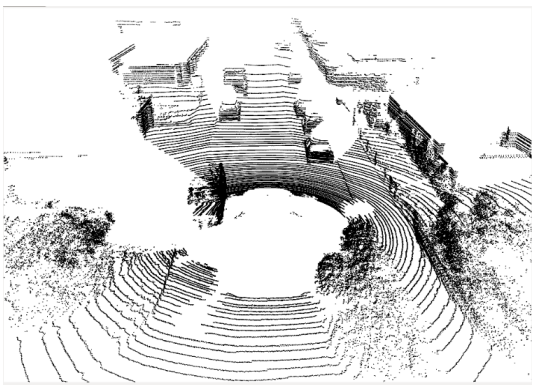
Two examples from the KITTI tracking data set and their segmentation results are given in Figure 2.6. It can be deduced from the results that our algorithm works sufficiently to segment the objects in sparsely 3D scenes.



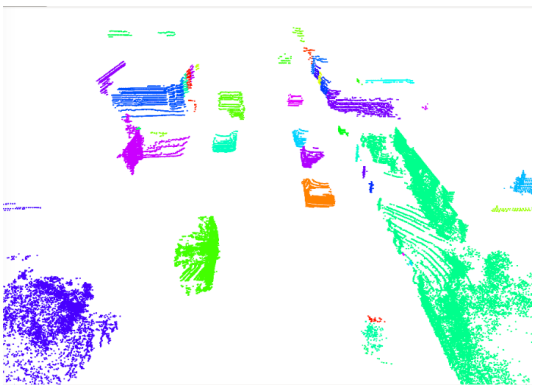
(a) Raw point cloud



(b) Segmented point cloud



(c) Raw point cloud



(d) Segmented point cloud

Figure 2.6. Two examples of segmentation results for KITTI tracking

3. REPRESENTATION OF POINT CLOUD OBJECTS

This chapter is on the representation of point cloud objects. As explained previously, this is integral to many robotic tasks such as those invoving human-robot interaction and manipulation. It enables the robot to store its sensor-based knowledge internally. The sensor-based knowledge is typically obtained either using Kinect-like sensors or LIDAR sensors. Kinect-like sensors have a practical ranging limit of five meters and are generally used for indoor scenes. On the contrary, LIDAR sensors have maximum ranges of thirty to hundred meters. As such, they are mostly suitable for both indoor and outdoor usage. Moreover, the two types of sensors also vary in their data type-namely sparse or dense. In this chapter, our focus is to find an appropriate representative descriptor that models the 3D objects, whether their data are sparse or dense. For this, we propose the deformable sphere approximation (DSA) descriptor. The DSA descriptor is rotationally invariant concerning the robot's heading and is also lightweight. Furthermore, it can be used with both dense and sparse data.

The outline of the chapter as follows: Firstly, related literature will be summarized in Section 3.1. Following, the computation of the DSA descriptor is explained in Section 3.2.2. The chapter will conclude with experimental results using benchmark 3D point cloud object data sets are presented in Section 3.3 - involving a comparative study with the state-of-the-art hand-crafted descriptors and learning network-based representations.

3.1. Related Literature

A 3D point cloud descriptor maps the sensory information from a 3D space to a feature space. Since its acts as a shape signature, it should keep as much information on the 3D shape possible. Shape descriptors are primarily used for class-level recognition because instance-level recognition is not generally possible using only point cloud data, even for human beings. A plethora of 3D descriptors has been proposed - either through

being hand-crafted [22] or more recently as learned by deep networks [23–25].

Deep learning networks simultaneously offer the multilayered representation and classification of the point cloud data. Network-based representations are mainly not class-agnostic. This is because obtaining features and learning are done together. Thus, learning requires class-labeled data in general. Moreover, if the robot encounters a new type of class, the network should be retrained. However, to cope with these issues, class-agnostic network-based representations are also being developed [26–28]. Classification networks operate on the level of whole object [29–33]. As such, they do not require detailed reasoning about the 3D structure of the object, as is the case with those aimed at reconstruction [34–36].

Another property pertains to whether input conversion is necessary or not. In most robotic applications, descriptors that are directly derived from point cloud data are preferred - as input conversion tends to increase the computational cost [37–39]. The raw point cloud data may be directly processed [40] or in most cases converted to other representations such as voxels [31, 41, 42], octrees [43], meshes [44], graphs [32], spherical functions [45,46] or multi-views of image data [47,48] before being input to the network. Unfortunately, some of the associated processing further increases the already high computational and memory resource requirements. Furthermore, even if the input conversion does not require for some methods, they still cannot be directly used - since they cannot be input arbitrarily sized input point cloud data [26–28]. Either the data needs to be pre-processed to convert it into a standard size, or the network structure needs to be changed. The former results in loss of information, while the latter requires learning to be redone.

The representation methods are also considered with respect to rotational invariance - particularly rotation around a vertical axis - as this results in recognition robustness [49]. For example, re-expressing the data as a spherical function yields rotationally invariant representations [45, 46]. The main advantage of spherical function representation is that it enables independence from the point cloud size.

The proposed descriptors also vary with respect to their flexibility of usage with different types (i.e., dense vs. sparse) data. With dense data such as Kinect, two of the best hand-crafted descriptors are View Point Feature Histogram (VFH) [50] and Ensemble of Shape Functions (ESF) [51] [52]. However, these cannot be directly used with sparse data such as LIDAR data. In such a case, interpolation techniques are required [53]. Alternatively, descriptors such as Global Fourier Histogram Descriptor (GFH) are designed specifically for sparse data such as LIDAR [54].

Interestingly, 3D representation has been originally considered within 3D solid modeling [55, 56] in the computer vision community. Here, one popular approach has been to use spherical harmonics [57]. For example, the descriptor is constructed from the coefficients of three spherical functions used to define the given 3D surface [58, 59]; however, for a reliable description, the surface type (open, closed, tori or tube) needs to be known. In another work, the descriptor is obtained through combining the different coefficients obtained by decomposing the three-dimensional data into a collection of functions defined on concentric spheres over different radii [60]. As such, the input needs to be transformed into a voxel representation.

3.2. Deformable Sphere Approximation Descriptor

Consider a segment o and recall $\mathcal{D}_o \subset R^3$ denote the respective point cloud. Let its mean be denoted by $\mu_o \in R^3$. The goal is to derive a d-dimensional vector I such that it encodes this data. We propose the deformable sphere approximation (DSA) descriptor for this. It is motivated by previous work on scene representation [63]. Similarly, its representation is derived from encoding the point cloud data distribution in the spherical coordinate system. However, differing from previous work, it is defined in an object-centric coordinate system as derived from the point cloud object data.

Table 3.1. The literature search for 3D descriptors and classification methods

Representation + Learning	Properties					
	Class Agnostic?	# Params.	Input Conversion	Direct Usage?	Vertical Rot. Invariance?	Varied Data Type?
SPH [60] + SVM	✓	544	64^2 Voxels	-	✓	✗
VFH [50] + MLP	✓	308	-	✓	✓	✗
ESF [51] + MLP	✓	640	-	✓	✗	✗
GFH + SVM [54]	✓	864	-	✓	✓	✗
3DShapeNets [29]	✗	38M	30^3 Voxels	-	✗	✓
ORION [42]	✗	4M	32^3,28^3 Voxels	-	✗	✓
VoxNet [41]	✗	890K	32^3 Voxels	-	✗	✓
ECC [32]	✗	-	32^3 Voxels	-	✗	✓
PointNet [30]	✗	3.5M	-	✗	✗	✓
PointNet++ [40]	✗	1.7M	-	✗	✗	✓
LightNet [61]	✗	30K	32^3 Voxels	-	✗	✓
FoldingNet [27]	✓	1M	-	✗	✗	✓
Latent-GAN [28]	✓	-	-	✗	✗	✓
GeoCNN [62]	✗	557K	-	✗	✗	✓
Spherical-CNN [45]	✗	500K	2×64^2	-	✓	✓
SF-CNN [46]	✗	-	-	✓	✓	✓
LP-3DCNN [33]	✗	2M	32^3 Voxels	-	✗	✓
ClusterNet [26]	✓	-	-	✓	✓	✗
DSA (proposed) + MLP	✓	400	-	✓	✓	✓

The deformed sphere approximation (DSA) descriptor is computed in three stages:

(i) First the point cloud object data $\mathcal{D}_o$ is mapped to a deformed sphere;

(ii) The deformed sphere is approximated by double trigonometric Fourier series;

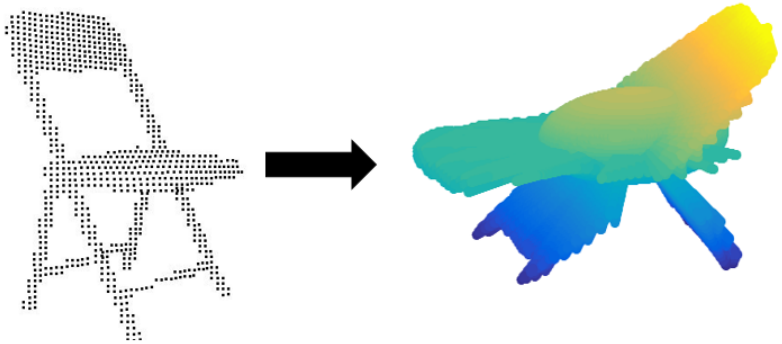
(iii) Rotational invariants are derived.

3.2.1. Deformed Sphere Mapping

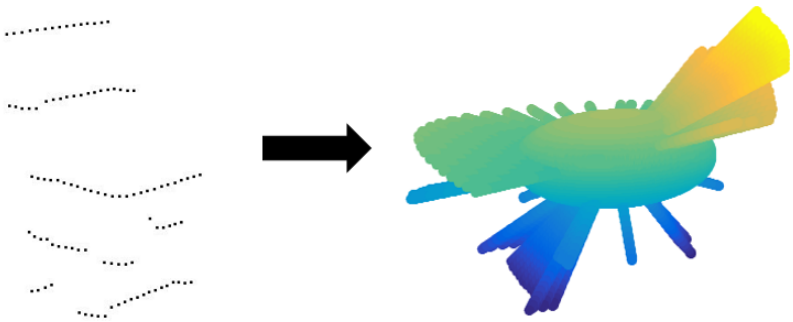
For each object o, the deformed sphere map $\rho_{co} : S^2 \to R^{\geq 0}$ is defined based on its associated point cloud data $\mathcal{D}_o$. Here, the first subindex c indicates robot's position dependency and second subindex o indicates object dependency:

$$\rho_{co}(f) = \begin{cases} \rho_0 + r(p) & \exists p \in \mathcal{D}'_o \text{ s.t. } f(p) = f \\ \rho_0 & \text{otherwise} \end{cases}$$

where $\mathcal{D}'_o$ is the transformed point cloud with origin at μ_o. The map $f : \mathcal{D}'_o \to S^2$ is defined as $f(p) = \begin{bmatrix} f_1(p) & f_2(p) \end{bmatrix}^T$ with $f_1(p) \in [-\pi, \pi]$ and $f_2(p) \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ as being the pan and tilt angles respectively and $r(p)$ is the respective radial distance in the local coordinate system. Note that the map ρ_{co} can be visualized as a deformed S^2-sphere with radius ρ_0 based on the respective data set $\mathcal{D}'_o$. The deformed sphere examples for Kinect and LIDAR point clouds for a chair object are shown in Figure 3.1.



(a) Deformed sphere for kinect point cloud data



(b) Deformed sphere for LIDAR point cloud data

Figure 3.1. Deformed spheres for a chair object with various sensor types

3.2.2. Deformed Sphere Approximation

Next, the deformed sphere map ρ_{co} is approximated using double trigonometric Fourier series (DTFS) [64]:

$$\rho_{co}(f) \cong \sum_{h_1=0}^{H_1-1} \sum_{h_2=0}^{H_2-1} \lambda_{h_1 h_2 h} z^T_{oh_1h_2}(c) e_{h_1h_2}(f). \tag{3.1}$$

In this equation, the parameters $\lambda_{h_1 h_2 h_t}$ are defined as:

$$\lambda_{h_1 h_2} = \begin{cases} 0.25 & \text{if } h_1 = 0, h_2 = 0 \\ 0.5 & \text{if } h_1 > 0, h_2 = 0 \text{ or } h_1 = 0, h_2 > 0 \\ 1 & \text{if } h_1 > 0, h_2 > 0. \end{cases} \tag{3.2}$$

The parameters H_1 and H_2 are positive-valued integers that correspond to the number of harmonics. For each pair (h_1, h_2) of harmonics, the vector $e_{h_1 h_2}(f) \in R^4$ is a vector of an orthonormal set of trigonometric basis functions:

$$e_{h_1 h_2}(f) = \begin{bmatrix} \cos(h_1 f_1)\cos(2h_2 f_2) \\ \sin(h_1 f_1)\cos(2h_2 f_2) \\ \cos(h_1 f_1)\sin(2h_2 f_2) \\ \sin(h_1 f_1)\sin(2h_2 f_2) \end{bmatrix}, \tag{3.3}$$

These functions have periods $-\pi \leq f_1 \leq \pi$ and $-\frac{\pi}{2} \leq f_2 \leq \frac{\pi}{2}$ and are orthogonal on the corresponding rectangle.

The set of vectors $z_{coh_1 h_2} \in R^4$, $h_1 = 0, \ldots, H_1 - 1$, $h_2 = 0, \ldots, H_2 - 1$ is comprised of double trigonometric Fourier series coefficients defined as:

$$z_{oh_1 h_2}(c) = \tfrac{2}{\pi^2} \begin{bmatrix} \int_{-\pi}^{\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \rho_{co}(f)\cos(h_1 f_1)\cos(2h_2 f_2) df_1 df_2 \\ \int_{-\pi}^{\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \rho_{co}(f)\sin(h_1 f_1)\cos(2h_2 f_2)) df_1 df_2 \\ \int_{-\pi}^{\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \rho_{co}(f)\cos(h_1 f_1)\sin(2h_2 f_2)) df_1 df_2 \\ \int_{-\pi}^{\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \rho_{co}(f)\sin(h_1 f_1)\sin(2h_2 f_2) df_1 df_2 \end{bmatrix} \tag{3.4}$$

Finally, the DSA descriptor $I_{co} \in R^d$ is obtained by considering the rotationally invariant $H_1 H_2$-dimensional vector:

$$I_{co} = \left[I_{co00}, \ldots, I_{co(H_1-1)(H_2-1)} \right]^T \tag{3.5}$$

Note that the DSA descriptor encodes single segment o as observed from location c with:

$$I_{coh_1h_2} = z^T_{coh_1h_2} z_{coh_1h_2}. \tag{3.6}$$

Two remarks are noteworthy: First, while DTFS representation is analogous to the spherical harmonics representation since they both use orthogonal basis functions defined over the sphere. However, they differ in terms of used basis functions. The former uses basic trigonometric functions, while the latter uses the standard spherical representation and spherical harmonic basis functions. Hence, while the coefficients of DTFS are computed directly as given by Equation 3.4, those of spherical harmonics need the computation of Legendre polynomials and 3-dimensional vector integration [65].

In practice, DTFS coefficients are numerically computed. This requires the discretization of the continuous integral of Equation 3.4. To do this, we use the sphere surface points as deformed by the centered point cloud object data o. Let $\delta f_1, \delta f_2 > 0$ denote the corresponding discrete differentials in the pan and tilt directions respectively. Then, the DTFS coefficients are numerically computed as:

$$z_{h_1h_2} \cong \frac{2}{\pi^2} \begin{bmatrix} \sum_{p\in\mathcal{D}_o} \rho(f(p)) \cos(h_1 f_1(p)) \cos(2h_2 f_2(p)) \delta f_1 \delta f_2 \\ \sum_{p\in\mathcal{D}_o} \rho(f(p)) \sin(h_1 f_1(p)) \cos(2h_2 f_2(p)) \delta f_1 \delta f_2 \\ \sum_{p\in\mathcal{D}_o} \rho(f(p)) \cos(h_1 f_1(p)) \sin(2h_2 f_2(p)) \delta f_1 \delta f_2 \\ \sum_{p\in\mathcal{D}_o} \rho(f(p)) \sin(h_1 f_1(p)) \sin(2h_2 f_2(p)) \delta f_1 \delta f_2 \end{bmatrix} \tag{3.7}$$

In general, for two different points $p, p' \in \mathcal{D}_o$ where $p \neq p'$, the corresponding discrete differentials will be different. This is because the density of point cloud object data will vary depending on the sensor type, 3D object shape and viewing geometry. In order to get a good estimate of the differentials, we use $\mathcal{N}_{M_2}(p)-$neighborhood of each point

$p \in \mathcal{D}_o$ - namely the closest M_2 point cloud data and take the average of associated differentials over the whole set $\mathcal{O}$:

$$\begin{bmatrix} \delta f_1 \\ \delta f_2 \end{bmatrix} = \frac{1}{M_2 \left|\mathcal{D}_o\right|} \sum_{p \in \mathcal{D}_o} \sum_{p' \in N_K(p)} \begin{bmatrix} \left\| f_1(p') - f_1(p) \right\|_{2\pi} \\ \left\| f_2(p') - f_2(p) \right\|_{\pi} \end{bmatrix} \tag{3.8}$$

Due to the fact that DTFS contains periodical functions, it is important to take into account Nyquist frequency during sampling. Especially for sparse data, the representation can be disrupted by false sampling. For this reason, we calculate maximum sampling rates for f_1 and f_2, and compare the rates with δf_1 and δf_2. The calculation of the maximum sampling rates is given in Equation 3.9, and the comparisons of calculated sampling rates with maximum sampling rates are given in Equation 3.10.

$$\begin{bmatrix} \delta f_{1max} \\ \delta f_{2max} \end{bmatrix} = \begin{bmatrix} 2\pi / H_1 \\ \pi / H_2 \end{bmatrix} \tag{3.9}$$

$$\begin{bmatrix} \delta f_1 \\ \delta f_2 \end{bmatrix} = \begin{bmatrix} min(\delta f_{1t}, \delta f_{1max}) \\ min(\delta f_{2t}, \delta f_{2max}) \end{bmatrix} \tag{3.10}$$

The selection of M_2 is essential, and it directly affects the sampling rate. Too high M_2 values result in the coarse sampling of the sphere surface, which negatively affects representativeness. On the other hand, too low M_2 values can adversely influence the descriptor's generalization with various sensors. Finally, it should be underlined that two or more points correspond to the same discrete differential surface area, then the deformation is done, taking into account their average depth values.

The DSA descriptor's computation has two stages: First, DTFS coefficients of the deformed sphere are computed. Let $N = |\mathcal{D}_o|$. This is of order $O(N \log N + NH_1H_2)$. The first term is due to δf_1 and δf_2 computation. The coefficients can then be incrementally computed depending on the sphere's deformation based on the depth data and the associated viewing geometry in spherical coordinates. Next, the DSA descriptor is derived. Here, the associated computation is of order $O(H_1H_2)$.

3.3. Experimental Results

The proposed descriptor has been evaluated with three well-known data sets having different data types:

(i) Kinect: The Washington RGB-D objects [66] data set contains 202,549 partial views obtained from Kinect sensor corresponding to household objects from 51 classes. To consider reliable data, only objects with at least 100 points are used.

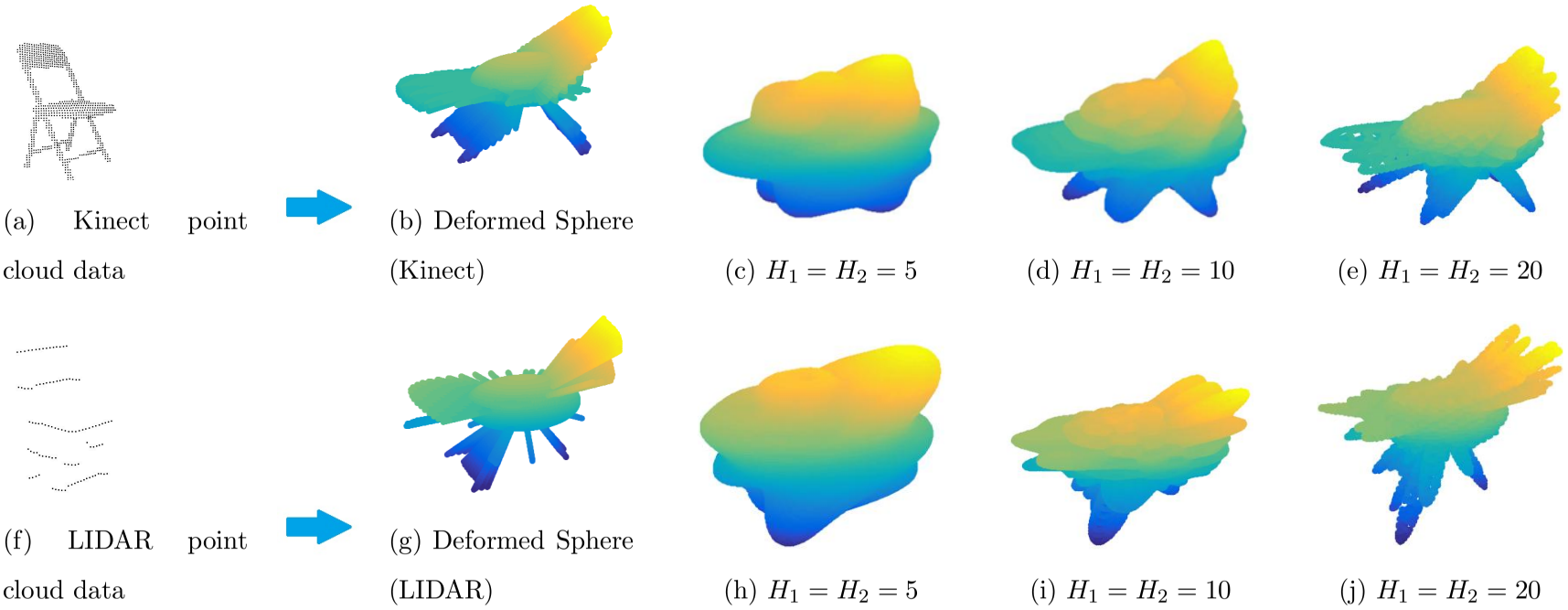
(ii) LIDAR: Sydney Urban Object data set contains with Velodyne HDL-64E LIDAR data of 588 labeled partial views from 14 various classes [53]. As recommended by the data set authors, the original data set is augmented with 18 rotations.

(iii) CAD: ModelNet10 and ModelNet40 involve mesh data of 10 and 40 object classes, respectively [29]. There are 4499 mesh objects divided into 3991 objects for training and 908 objects for the test in ModelNet10. There are 12,311 mesh objects that are split into 9843 objects for training and 2468 for the test in ModelNet40. The data uniformly sampled with 1024 points, then the sampled data is normalized into a unit sphere. [67]

3.3.1. Approximated Spheres by DSA

At first, we want to visually show how DSA descriptors represent the deformed sphere into an approximated sphere. The fact that no data set contains both Kinect and LIDAR data in the literature. For this reason, we select a chair object from ModelNet10, generate Kinect and LIDAR point cloud data of the object. The chair object is

first viewed with a Kinect sensor, and the associated point cloud object data is shown in Figure 3.2a. It is observed to be quite dense. Next, the same object is now viewed with a LIDAR with the same robot pose. The resulting point cloud data is quite sparse, as seen in Figure 3.2f. To assess whether the data type affects the descriptor or not, we compare the approximations of the deformed spheres generated for each data type as shown in Figure 3.2b and Figure 3.2g respectively. Two observations are notable: First, despite the big difference in data density, the deformed sphere representations are quite similar. Second, as the number of harmonics increases, approximation error decreases as expected. On the other hand, it also results in minute shape details being unnecessarily encoded within the descriptor.



(a) Kinect point cloud data

(b) Deformed Sphere (Kinect)

(c) $H_1 = H_2 = 5$

(d) $H_1 = H_2 = 10$

(e) $H_1 = H_2 = 20$

(f) LIDAR point cloud data

(g) Deformed Sphere (LIDAR)

(h) $H_1 = H_2 = 5$

(i) $H_1 = H_2 = 10$

(j) $H_1 = H_2 = 20$

Figure 3.2. Deformed spheres of objects and their approximations.

3.3.2. Classification Performance Across Data Types

We consider the classification performance of the proposed approach across three data types. DSA descriptor constructed with $\rho_0 = 1$, and three different number of harmonics $H_1, H_2 \in \{5, 10, 20\}$. We set M_2 to 2 for LIDAR and CAD objects, and we set M_2 to 5 for Kinect objects. Therefore, the sizes of obtained DSA descriptors are 25, 100, 400. For learning, Multi-Layer Perceptron (MLP) classifiers are used. MLPs are trained with different training/test split strategies. For the Kinect data set, we divide the data into 80% training set and 20% test set. For the LIDAR data set, we use the strategy of the authors [53]. Accordingly, the augmented LIDAR data is split into four-folds. The average of the performances classifiers' performances for the folds are taken as the result. Lastly, using original training and test splits for ModelNet10 and ModelNet40, we evaluate the proposed descriptor for point cloud data from CAD objects. It is important to state that differing from most methods; we do not implement any augmentation on the ModelNet objects. In the tests, the point cloud object's class is assigned to the class with the maximum probability score found by the MLPs. For the Kinect data set, accuracy and F1-score are used as the performance measure. For the LIDAR dataset, F1-score is used as the performance measure. Finally, we use accuracy performance for CAD objects. The results are as shown in Table 3.2. The best performance is obtained with with $H_1 = H_2 = 10$. Using a larger number such as $H_1 = H_2 = 20$ can lead to overfitting - as it encodes the object surface with unnecessary minute detail.

Table 3.2. DSA performance across data types with different harmonics.

H_1, H_2	Size	Kinect		LIDAR	CAD	
		Accuracy	F1 Score	F1 Score	Accuracy (ModelNet10)	Accuracy(ModelNet40)
5	25	91.71	0.919	0.790	85.16	70.22
10	100	92.73	0.928	0.803	85.93	68.44
20	400	90.18	0.908	0.793	81.98	62.52

3.3.3. Comparative Classification Performance

We compare the classification performance of the DSA descriptor with state-of-the-art baseline approaches. There are both hand-crafted and network-based descriptors. For the comparison, the DSA descriptor constructed with $H_1 = H_2 = 10$ is used since it shows the best performance. The comperative classication performance is given in Table 3.3. In the this table $\rightsquigarrow$ means the method needs input conversion, and - means the algorithm does not work without additional processing. The results as presented in Table 3.3 can be detailed as follows:

(i) Kinect data set: We consider ESF and VHF that are known to be two of the best hand-crafted and inherently Kinect-style descriptors [52]. In all experiments, the same MLP structure is used. To enforce fair comparison, the descriptors are directly formed from the input point cloud data without any pre-processing to all. All descriptors are standardized before MLP training. It is observed that the classification performance of all the descriptors is roughly the same.

(ii) LIDAR data set: We consider both the hand-crafted GFH and network-based descriptors obtained with ORION, VoxNet, ECC, and LightNet. It is observed that the DSA descriptor has the best average F1-Score of 0.803.

(iii) CAD (ModelNet10 and ModelNet40) data sets: In general, network-based representations are used to evaluate the data sets. It is observed that accuracy varies between 79.79%-95.3% for ModelNet10 and between 68.23%-93.4% for ModelNet40. It is observed that all descriptors have a lower accuracy with ModelNet40. While the proposed DSA descriptor's performance is not the highest, it is still relatively high - 85.9% and %68.4 for ModelNet10 and ModelNet40, respectively. It is partially attributed to the fact that since all deformed spheres are constructed with 1024 points sampled uniformly. Moreover, ModelNet40 objects are not aligned according to a specific axis differing from ModelNet10. These situations can cause a decrease in the performance of our proposed method. In a nutshell, the DSA descriptor can represent point cloud objects consistently across different data types without any pre-processing required.

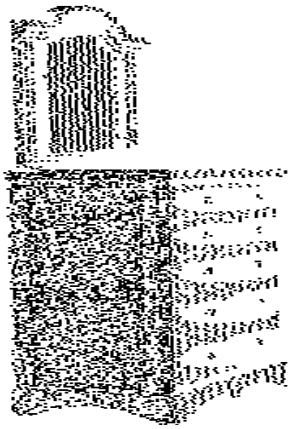
Table 3.3. Comparative classification performance.

Representation + Learning	Data Types				
	Kinect (Dense)		LIDAR (Sparse)	CAD	
				ModelNet10	ModelNet40
	Accuracy(%)	F1 Score	F1 Score	Accuracy(%)	Accuracy(%)
SPH [60] + SVM ⇝	-	-	-	79.79	68.23
VFH [50] + MLP	92.29	0.9240	-	-	-
ESF [51] + MLP	93.93	0.9352	-	-	-
GFH + SVM [54]	-	-	0.710	-	-
3DShapeNets [29] ⇝	-	-	-	83.5	77.3
ORION [42] ⇝	-	-	0.778	93.9	89.7
VoxNet [41] ⇝	-	-	0.730	92.0	83.0
ECC [32] ⇝	-	-	0.784	89.3	82.4
PointNet [30]	-	-	-	-	89.2
PointNet++ [40]	-	-	-	-	90.7
LightNet [61] ⇝	-	-	0.796	93.4	88.9
FoldingNet [27]	-	-	-	94.4	88.4
Latent-GAN [28]	-	-	-	95.3	85.7
GeoCNN [62]	-	-	-	-	93.4
Spherical-CNN [45] ⇝	-	-	-	-	88.9
SF-CNN [46]	-	-	-	-	91.4
LP-3DCNN [33] ⇝	-	-	-	94.4	92.1
ClusterNet [26]	-	-	-	93.8	86.8
DSA (proposed) + MLP	92.73	0.9285	0.803	85.9	68.4

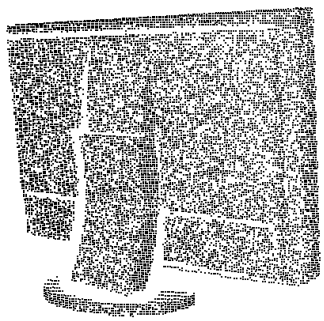
3.3.4. Transfer Learning Across Data Types

Classification experiments show that the DSA descriptor can be used to represent both Kinect and LIDAR point cloud objects reliably. It suggests that it may be possible to transfer the learning of objects with one type of sensor to classify those obtained with other types of sensors. Next, we have investigated how much is possible. To do this, we use the Kinect and LIDAR point cloud object data obtained from ModelNet10 dataset. The ModelNet10 dataset has ten different classes with 100 objects selected randomly

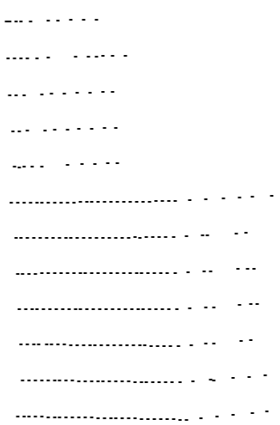
from each class. In order to be able to create point cloud data, we randomly sample the mesh of each object with 100,000 points. As the objects have varying scales, these points are brought to a standard scale by fitting the sampled data in a unit sphere. We simulated a mobile robot with sensors positioned at 50cm height from the ground is made to view these objects from varying poses around the objects. Its distance to the objects varies 1 and 6 meters with increments of 0.2m while its angular position varies between $[-180°,\ 180°]$ with increments $10°$. Hence, there are 900 various alternatives.



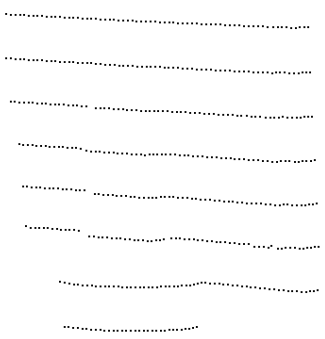
(a) Kinect point cloud dresser data



(b) Kinect point cloud monitor data



(c) LIDAR point cloud dresser data



(d) LIDAR point cloud monitor data

Figure 3.3. Sample Kinect and LIDAR objects data from ModelNet10.

We choose 100 different positions randomly for each object. Both Kinect and LIDAR point cloud object data are obtained. For Kinect, pan and tilt resolutions for obtaining the point cloud data are set as 0.0895° and 0.095°, respectively. For LIDAR, they are set as 0.2° and 2° respectively. Finally, hidden points are eliminated using the Hidden Point Removal algorithm [68] along with depth buffering. Two samples of the objects from Kinect and LIDAR can be seen in Figure 3.3.

We consider three different level of harmonics $H_1, H_2 \in \{5, 10, 20\}$. Using a random split strategy, 80% of the samples are used for learning, while the remaining 20% are used in the classification. Four alternative schemes are investigated. In two, we use the same data types for both learnings - namely either only Kinect or LIDAR data. This is done to determine the best possible performance levels. Next, we consider learning transfer across data types: learning based on Kinect data followed by testing with LIDAR data and learning based on LIDAR data followed by testing with Kinect data. The results are as shown in Table 3.4.

First, we observe that without any transfer learning, the best performance is obtained with $H_1 = H_2 = 10$. Second, learning transfer across data types yields around 30-46% accuracy. We find this perfect considering Kinect and LIDAR objects are very different from each other. Kinect data is rather dense, whereas LIDAR data is rather sparse. Interestingly, LIDAR to Kinect learning transfer has better accuracy. It is attributed to the fact that LIDAR objects have a lesser amount of detail on the respective object surfaces. As such, learning with LIDAR means learning general shape characteristics. Hence, learning is more easily transferred from LIDAR to Kinect point cloud object data.

Table 3.4. Classification performance with learning transfer across data types.

H_1, H_2	Training Set	Test Set	Accuracy (%)	F1 Score
5	Kinect	Kinect	$86,03$	0.8624
	Kinect	LIDAR	$32,89$	0.3016
	LIDAR	LIDAR	$72,90$	0.7105
	LIDAR	Kinect	$44,46$	0.4357
10	Kinect	Kinect	$88,61$	0.8897
	Kinect	LIDAR	$30,56$	0.2710
	LIDAR	LIDAR	$75,52$	0.7518
	LIDAR	Kinect	$46,09$	0.4504
20	Kinect	Kinect	$87,46$	0.8795
	Kinect	LIDAR	$20,34$	0.1829
	LIDAR	LIDAR	$73,95$	0.7347
	LIDAR	Kinect	$43,83$	0.4319

4. OBJECT LEARNING FOR SCENE MAPPING

The chapter studies the problem of improving object learning through considering all the knowledge available to the robot. In the previous chapter, the robot considers the instantaneous sensory data, encodes this data using the proposed DSA descriptor, and then learns objects or recognizes objects based on these descriptors. However, if the robot is moving, the incoming data has a temporal aspect. Using this data will have two advantages. First, the knowledge accumulated through the objects' temporal continuity can also help the robot better recognize the objects around it. By the spatio-temporal accumulation of data, a more comprehensive knowledge regarding these objects can be built. This can be especially beneficial with sparse data such as 3D LIDAR scans. Second, the robot will not need to reason about objects that are already classified, and hence it can reduce the burden of computational processing. In this chapter, we propose an approach that considers using temporal data so that the robot uses both instantaneous and accumulated knowledge. For this, we propose the temporal deformable sphere approximation (T-DSA) descriptor. The T-DSA descriptor is designed to encode a stream of point cloud object data about each distinct object in the scene. As such, it requires the robot to track the detected objects. The robot then uses the respective track data in order to construct the T-DSA descriptor.

The outline of the chapter is as follows: First, we review the related literature in Section 4.1. Then the general approach for the proposed method is mentioned briefly in Section 4.2. After that, our novel multi-object tracking method is given in Section 4.3. Then, we introduced the strategy for merging instant features in Section 4.4. Finally, the experimental results are given in Section 4.7.

4.1. Related Literature

As the robot is navigating around in a scene, it has a continuous stream of incoming point cloud data. Typically most work base their reasoning on the instantaneous

and do not take advantage of its sequential nature. However, using the temporal nature presents two main advantages for point cloud processing. It requires the robot to establish the relationship between consecutive frames.

The typical approach is to use ICP-based (or its variants) algorithms in order to merge the LIDAR frames [69, 70]. Moreover, the ICP-based methods can suffer from large pose displacements. Statistical methods can be used for alleviating these problems [71, 72]. They can deal with some drawbacks of ICP-based algorithms, but they are still computationally expensive methods.

The second approach is to track objects in sequential frames. This problem has also been studied extensively and many 3D object tracking algorithms have been proposed through extending the existing two-dimensional (2D) tracking methods [73–75]. Thus, most of them are based on bounding boxes with the help of Kalman Filter [76,77]. Some work uses the features or points in the LIDAR scan [78–83] to take advantage of LIDAR systems' reliability.

The information from the tracked object should be used for the improvement in efficiency and reliability. [15, 84] use sequential features in a probabilistic manner to benefit from the tracked objects. With recent developments in deep learning, some advanced networks, such as RNN, can keep the frames' information in a certain time. In fact, RNN (or its special type LSTMs) can merge the segments with the measured robot states [85, 86]. Choy et al. [87] propose a modern CNN approach to evaluate objects temporally. On the other hand, another method is to complete the shape using either using a trained neural network or auto-encoder [28, 88, 89].

4.2. Object Learning For Scene Mapping: General Approach

Consider the robot to be navigating through a sequence of locations $c_k = [c_{k1}\, c_{k2}]^T \in R^2$ with headings $\alpha_k \in S^1$. Here, $k \in \mathcal{K}$ where $\mathcal{K} = \{0, 1, \ldots, \}$ is the ordered set of discrete time index. We propose an approach that enables the problem to consider

both instantaneous and temporally accumulated knowledge of objects. Our proposed approach consists of the following stages: The robot acquires a new point cloud data frame and determines the object candidates. It then tracks these candidates with a novel tracking algorithm. It then encodes the tracked objects' data using a novel descriptor T-DSA (Temporal Deformable Sphere Approximation) descriptor. Finally, it combines knowledge from both the DSA and T-DSA descriptors in its learning and reasoning.

4.3. Multi Object Tracking

Suppose that the robot has determined a set of point cloud objects $\mathcal{O}_k$ at time k. For each object $o \in \mathcal{O}_k$, a track T_o is formed. The track is defined by a ordered set of states $T_o = \{x_k \mid k \in \mathcal{K}_o\}$. The states are derived from the respective point cloud data $\mathcal{D}_o \subset R^3$ with $o \in \mathcal{O}_k$. Let the point cloud data mean be defined by $\mu_{ok} = \begin{bmatrix} \mu_{ok_1} & \mu_{ok_2} & \mu_{ok_3} \end{bmatrix}^T \in R^3$. Furthermore, let $\Delta\mu_{ok} = \begin{bmatrix} \Delta\mu_{ok_1} & \Delta\mu_{ok_2} & \Delta\mu_{ok_3} \end{bmatrix}^T \in R^3$ represent the change in mean from previous scan to the current scan. The mean and its change $\Delta\mu_{ok}$ are used to define the object candidates' states as:

$$x_{ok} = \left[\mu_{ok_1}, \mu_{ok_2}, \mu_{ok_3}, \Delta_{\mu_{ok_1}}, \Delta_{\mu_{ok_2}}, \Delta_{\mu_{ok_3}}\right]^T \tag{4.1}$$

The ordered set K_o corresponds to the index sequence of the segments in the track. Furthermore, if $K_o = \{k_1, \ldots, k_N\}$, then k_1 indicates when the the track starts and k_N indicates the last index in the track. Suppose track ends when $k_N = k_E$. As long as the robot continues with its tracking of segment o, the set $\mathcal{K}_o$ expands accordingly.

The robot uses Kalman Filtering and matching object candidates from adjacent frames to update the states. For Kalman filtering, we use constant velocity model [90] - similar to [76,77]. However, our algorithm is different from these works as it does not use bounding box properties to track the segments. This is because using a bounding box

can hinder the detection of under-segmentation or over-segmentation errors. Rather, we use a novel approach based on matching segments from consecutive frames. As such, the tracking approach consists of three different stages:

(i) Object candidates' state prediction
(ii) Matching segments from consecutive point cloud data frames to determine corresponding segments
(iii) Updating the object candidates' states

4.3.1. Model States and State Prediction

The dynamics of the states are defined as:

$$\hat{x}_{ok} = A_x x_{ok-1} + u_{ok-1} \tag{4.2}$$

$$\hat{y}_{ok} = A_y \hat{x}_{ok} + v_{ok-1} \tag{4.3}$$

where, $\hat{y}_{ok}$ indicates the observation, A indicates state transition matrix, and H indicates observation matrix. We assume constant linear velocity for the center coordinates, so there is no change for $\Delta_{o_{k_1}}, \Delta_{o_{k_2}}, \Delta_{o_{k_3}}$ from $k-1$ to k. A_x and A_y are defined as follows:

$$A_x = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad A_y = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Here, $\hat{x}_{ok}$ is calculated upon the previous state x_{ok-1} and process noise u_{ok-1}, and $\hat{y}_{ok}$ is calculated upon the current state $\hat{x}_{ok}$ and observation noise v_{ok-1}. u_{ok} and v_{ok} follow Gaussian distributions with zero means and covariances Σ_1 and Σ_2. These noise

parameters are experimentally determined and influence the prediction performance.

4.3.2. Segment Matching

The core part of multi-object tracking is to match the state predictions with the detected segments in the current scene. The matching based on a similarity matrix G that is constructed using the set of objects $\mathcal{O}_{k-1}$ and $\mathcal{O}_k$ as determined in two consecutive instances. For each tuple $(o, o') \in \mathcal{O}_k \times \mathcal{O}_{k-1}$, the entry $G(o, o')$ of this matrix measures the similarity of the objects o and o'. The similarity measures encodes two different metrics:

(i) Similarity of positions $\beta_1(o, o')$
(ii) Similarity of overall shapes $\beta_2(o, o')$

$$G(o, o') = \nu_1 \, \beta_1(o, o') + \nu_2 \, \beta_2(o, o') \tag{4.4}$$

Here, ν_1 and ν_2 are the weights for similarity matrix. We set these two parameters to 0.5 to use the average of $\beta_1(o, o')$ and $\beta_2(o, o')$.

The similarity of positions is computed based on the average distance between their respective point cloud data. Recall that for each $o \in \mathcal{O}_k$ represents the set of N_{k-1} point clouds are associated with object o at time k. It is important to state that the point clouds of the segments at time $k-1$ are shifted according to $\hat{x}_{ok}$. For each $o' \in \mathcal{O}_{k-1}$ we construct nearest neighbors models using KD-tree [91]. Finally, position similarity measure $\beta_1(o, o')$ of object $o \in \mathcal{O}_k$ and $o' \in \mathcal{O}_{k-1}$ feature $(\beta_1(o, o'))$ is defined as follows:

$$\beta_1(o, o') = \frac{1}{||\mathcal{D}_o||} \sum_{p \in \mathcal{D}_o} \zeta(p, \mathcal{D}_{o'}) \tag{4.5}$$

where $\zeta(p, \mathcal{D}_{o'})$ refers to the closest point cloud point in the set$\mathcal{D}_{o'}$. In some cases, searching for all points in $\mathcal{D}_{o'}$ can be computationally expensive. To speed up the calculation, we use a smaller set of randomly selected points from $\mathcal{D}_{o'}$.

The shape similarity is computed based on comparing their covariance matrices using Förstner and Moonen's covariance distance metric [92]. For each $o \in \mathcal{O}_k$, let Q_o denote the respective point cloud data $\mathcal{D}_o$ expressed in spherical coordinates. Shape similarity $\beta_2(o, o')$ of two objects $o \in \mathcal{O}_k$ and $o' \in \mathcal{O}_{k-1}$ is defined as follows:

$$\beta_2(o, o') = \sqrt{trace(\ln^2(\sqrt{\Sigma_o^{-1}}\Sigma_{o'}\sqrt{\Sigma_o^{-1}}))} \tag{4.6}$$

where Σ_o denotes the covariance matrix of Q_o.

Using the similarity matrix, we use a greedy algorithm to find matching pairs [77]. The pseudo-code for the greedy algorithm is given in Figure 4.1.

4.3.3. State Updates

The states are updated using matching information as follows:

$$x_{ok} = A_x\hat{x}_{ok} + F_k(y_{ok} - \hat{y}_{ok}) \tag{4.7}$$

Here, F_k represents Kalman gain matrix, which is recursively calculated, and y_{ok} represents the center coordinates of the matched segment at time k as determined from the respective point cloud data - namely $y_{ok} = \mu_{ok}$.

Input:

G : Similarity matrix

τ_m: Threshold for matching

Output:

List of matched pairs

Initialization:

$MP \Leftarrow \emptyset$: *matched pairs*

$M_{k-1} \Leftarrow \emptyset$ *matched object from* $\mathcal{O}_{k-1}$

$M_k \Leftarrow \emptyset$: *matched object from* $\mathcal{O}_k$

sortedPairs $\Leftarrow$ *IndexPairsSortbyDistance(G)*

for $n = 1 : |sortedPairs|$ **do**

  $(l, d) = sortedPairs(n)$

  if $l \notin M_{k-1}$ and $d \notin M_k$ **then**

    if $G(l, d) < \tau_m$ **then**

      $MP.append((l, d))$

      $M_{k-1}.append(l)$

      $M_k.append(d)$

    else

      break

    end if

  end if

end for

return MP

Figure 4.1. Greedy algorithm for multi object matching

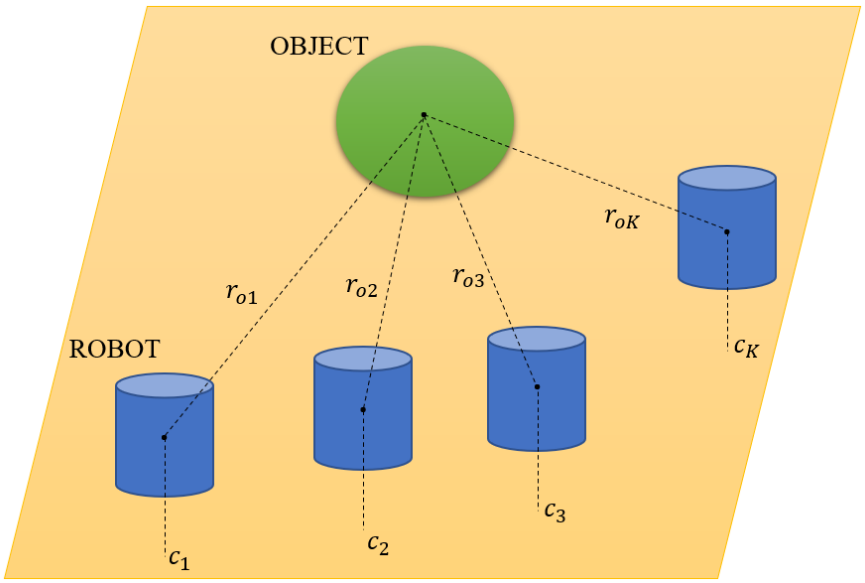


Figure 4.2. T-DSA is incrementally formed as a track evolves

4.4. Temporal DSA

The Temporal DSA (T-DSA) descriptor is the Spatio-temporal extension of the DSA descriptor to accumulate tracked objects' instantaneous DSA features. The accumulation is done over the track path of each object with respect to the robot. This is preferred to encode the relative position change of the object concerning the robot. For each tracked object o, consider the path R_o in the $x_1 - x_2$ plane described by $\{c_k\}_{\mathcal{K}_o}$ as showed in Figure 4.2. Let the length of the path be defined by $2\Delta_o$. Let r_{ok} denote the planar distance of the robot to the object at time k as defined by:

$$r_{ok} = \sqrt{\mu_{ok1}^2 + \mu_{ok2}^2} \tag{4.8}$$

Now consider the $\varphi_o : R_o \times \mathcal{F} \to R$ over this path with values as follows:

$$\varphi_o(r, f) = \begin{cases} \rho_{c_o}(f) & \text{if } \exists k \in \mathcal{K}_o \text{ s.t. } r = r_{ok} \\ \rho_0 & \text{otherwise} \end{cases} \tag{4.9}$$

As such, spatial samples of φ_{ok} correspond to observations from r_{ok}. Hence, for $k \in \mathcal{K}_o$, the function φ_{ok} encodes the knowledge accumulated across the track of object o upto index k - namely from the start of change in r_{ok} track at location r_{ok_1} to location r_{ok}.

The map φ_o can then be approximated as:

$$\varphi_o(r, f) \cong \sum_{m=0}^{H_3-1} \lambda_m y_m^T(f) e'_m(r) \tag{4.10}$$

where H_3 is the number of spatial harmonics, λ_m is defined as in Eq. 3.2 and $e'_m(r)$ is defined as:

$$e'_m(r) = \begin{bmatrix} \cos(\frac{mr\pi}{\Delta_o}) \\ \sin(\frac{mr\pi}{\Delta_o}) \end{bmatrix},$$

The vectors $y_m(f)$ are defined as:

$$y_m(f) = \tfrac{1}{\Delta_o} \begin{bmatrix} \int_{-\Delta_o}^{\Delta_o} \varphi_o(r, f) \cos(\frac{mr\pi}{\Delta_o}) \delta r \\ \int_{-\Delta_o}^{\Delta_o} \varphi_o(r, f) \sin(\frac{mr\pi}{\Delta_o})) \delta r \end{bmatrix} \tag{4.11}$$

Now, also using the approximation of ρ_o on the rhs of Eq. 4.12,

$$\varphi_o(r, f) \cong \sum_{m=0}^{H_3-1} \sum_{h_1=0}^{H_1-1} \sum_{h_2=0}^{H_2-1} \lambda_m \lambda_{h_2 h} w_{omh_1h_2}^T (e'_m(r) \otimes e_{h_1h_2}(f)) \tag{4.12}$$

Here, $\otimes$ denotes the Kronecker product. Namely if $A \in R^{n \times m}, B \in R^{p \times q}$, then $A \otimes B \in R^{np \times mq}$ with an ij^{th} block of size $p \times q$ specified by $a_{ij}B$. The vector $w_{omh_1h_2} \in R^8$ is defined as:

$$w_{omh_1h_2} = \tfrac{1}{\Delta_o} \begin{bmatrix} \int_{-\Delta_o}^{\Delta_o} z_{oh_1h_2}(r) \cos(\frac{mr\pi}{\Delta_o}) \delta r \\ \int_{-\Delta_o}^{\Delta_o} z_{oh_1h_2}(r) \sin(\frac{mr\pi}{\Delta_o}) \delta r \end{bmatrix} \tag{4.13}$$

Finally, the T-DSA descriptor is obtained by only considering $H_3 = 0$ and then observing that only the first four terms of $w_{o0h_1h_2}$ are non-zero. The descriptor I_o^A is formed based on these terms by considering the rotationally invariant H_1H_2-dimensional vector:

$$I_o^A = \left[I_{o0}^A, \ldots, I_{o(H_1-1)(H_2-1)}^A\right]^T \tag{4.14}$$

where

$$I_{oh_1h_2}^A = w_{o0h_1h_2}^T w_{o0h_1h_2}. \tag{4.15}$$

In practice, the vectors $w_{omh_1h_2}$ are numerically computed via the discretization of the continuous integral of Equation 4.13.

4.5. Classification Decisions

The robot has both instantaneous and accumulated knowledge. The formed is encoded through the DSA descriptors and the latter is encoded by the T-DSA descriptors.

It can use any learning scheme to learn each separately. In this work, we use MLP classification models. These models are then used to compute the probability of a descriptor being a given class c. Let $\mathcal{U}_c^i(I)$ represent the probability of c class for the DSA descriptor I and $\mathcal{U}_c^A(I)$ represent the probability of of c class for the descriptor I^A using the accumulated T-DSA descriptors.

We can then use a variety of different performance measures for classification. Recall for a tracked object o, $I_{okh_1h_2}$ represents the instantaneous DSA descriptor taken at discrete time index k and $I_{oh_1h_2}^A$ represents T-DSA descriptor. These measures are defined based on whether they use only instantaneous data, accumulated data or both.

The following measures consider only instantaneous data:

(i) Measure-1: $\max_c \mathcal{U}(I_{oh_1h_2})$
This measure considers instantaneous DSA descriptors only and chooses the class with the highest probability.

(ii) Measure-2: $\max_c \frac{1}{\mathcal{K}} \sum_{k \in \mathcal{K}} \mathcal{U}(I_{okh_1h_2})$
This measure computes the average until time k and chooses the class with the highest average probability.

Similarly, the robot can consider only accumulated data:

(i) Measure-3: $\max_c \mathcal{U}^A(I^A_{oh_1h_2})$
This measure considers the T-DSA descriptors and chooses the class with the highest average probability.

(ii) Measure-4: $\max_c \frac{1}{\mathcal{K}} \sum_{k \in \mathcal{K}} \mathcal{U}^A(I^A_{okh_1h_2})$
This measure considers the average T-DSA descriptors based and chooses the class with the highest average probability

Finally, the robot can also use both and integrate their results:

(i) Measure-5: $\max_c \frac{1}{2\mathcal{K}} \sum_{k \in \mathcal{K}} \mathcal{U}(I_{okh_1h_2}) + \frac{1}{2\mathcal{K}} \sum_{k \in \mathcal{K}} \mathcal{U}^A(I^A_{okh_1h_2})$
This measure adds probabilities from DSA and T-DSA descriptors and chooses the class with the highest weighted probability.

4.6. Scene Mapping

In the scene mapping procedure, each object is assigned with its predicted labels in a unique position according to their center of points. The map frame is originated where the robot starts moving. The fact that the classification models can change their decision while the robot is moving. We update the object's map decision with the latest decision from the classification model in these circumstances. Scene mapping

procedure can work with various decision models of classification models.

4.7. Experimental Results

In this section, we discuss experimental results regarding scene mapping.

4.7.1. Simulation Results

First, we consider indoor settings. To the best of our knowledge, there is no sequential LIDAR dataset for indoor environments. For this reason, we conduct our experiments in Gazebo environment using the Velodyne simulator. We consider 15 objects- considering five Gazebo objects with three different categories [93]. The used objects are given in Figure 4.3. A sample from each class is placed in an environment. The robot is also randomly placed in this environment with a random heading. It then starts moving at a random speed as long as it does not reach workspace boundaries. Throughout its movement, it acquires point cloud data along. This is repeated 15 times for each object sample. Hence, 45 paths are obtained.

Figure 4.3. Learning objects from Gazebo

The parameters for learning and scene experiments are given in Table 4.1.

Table 4.1. Parameters for learning and experiments

δ_ϕ	δ_ψ	τ_ϕ	τ_ψ	τ_ρ	M_1	H_1	H_2	M_2	τ_m	δ_r
0.2°	2°	1.5°	6°	$0.8m$	5	10	10	2	1	$0.1m$

The robot uses the incoming data to construct DSA and T-DSA descriptors which are then used to learn two separate MLP classification models. The learned models are then used in object classification tests by having the robot move through 3 paths for the 15 Gazebo objects in a similar (random) manner. In many paths, the results from the two models turn out to be similar. A sample case is shown in Figure 4.4. Here, both the instantaneous and accumulated knowledge correctly classify the tracked object as class 2. This does not turn out to be always the case, as shown in Figure 4.5. Here, it is observed that the decisions based on the instantaneous DSA descriptor are not correct in certain instances. However, the accumulated knowledge can continue with correct decisions regardless. Such an improvement is even more evident in the next case, as shown in Figure 4.6. Here while both models are observed to yield a greater number of wrong decisions nevertheless, the performance with accumulated knowledge based on the T-DSA descriptors is much better. Of course, there are also cases where both the instantaneous and accumulated knowledge cannot recover from wrong decisions, as is the case in Figure 4.7. Here, while the ground truth is class 4, the robot's classification decision is false.

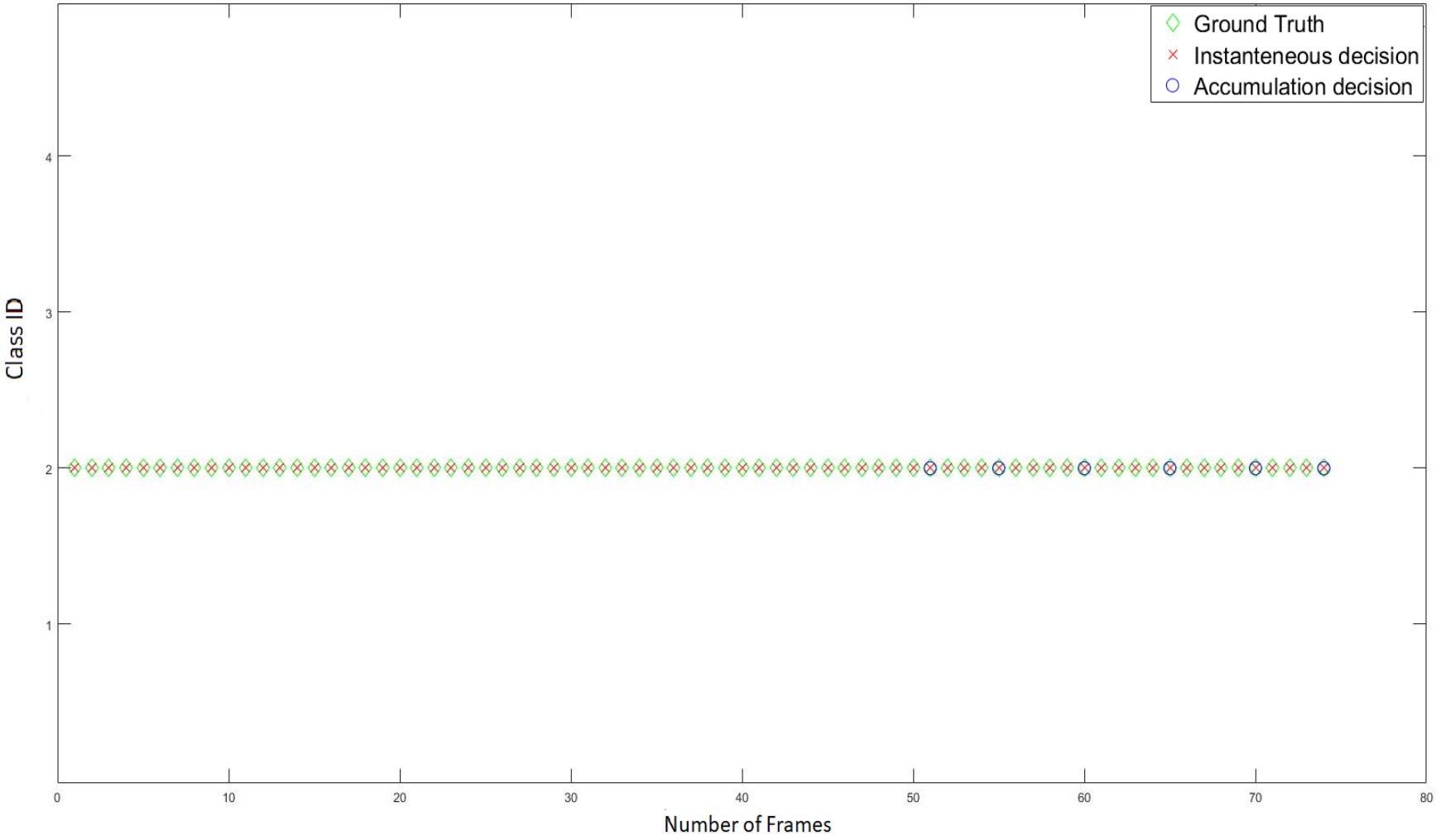


Figure 4.4. Both instantaneous and accumulated decisions agree.

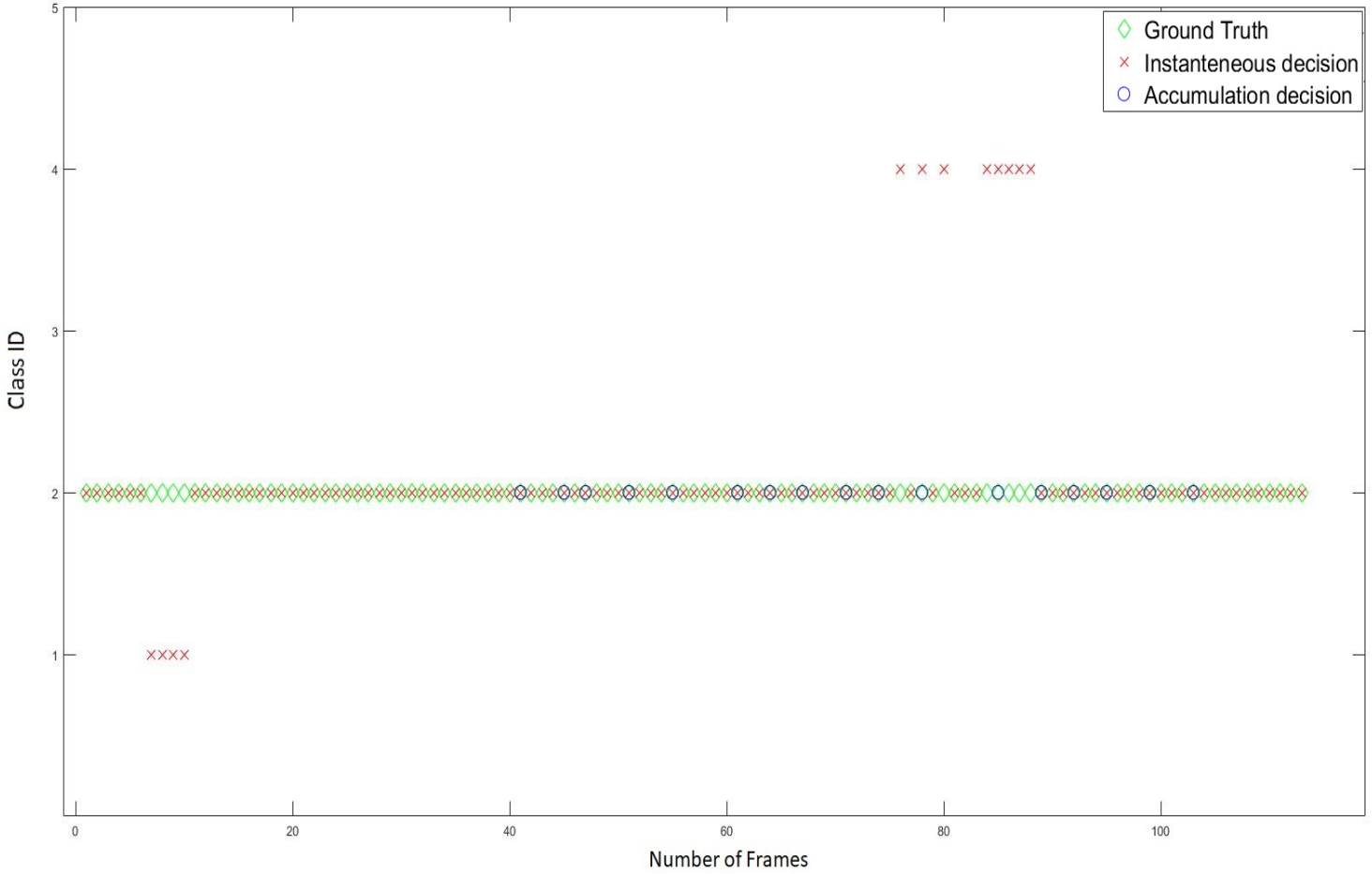


Figure 4.5. Accumulated knowledge corrects instantaneous decisions.

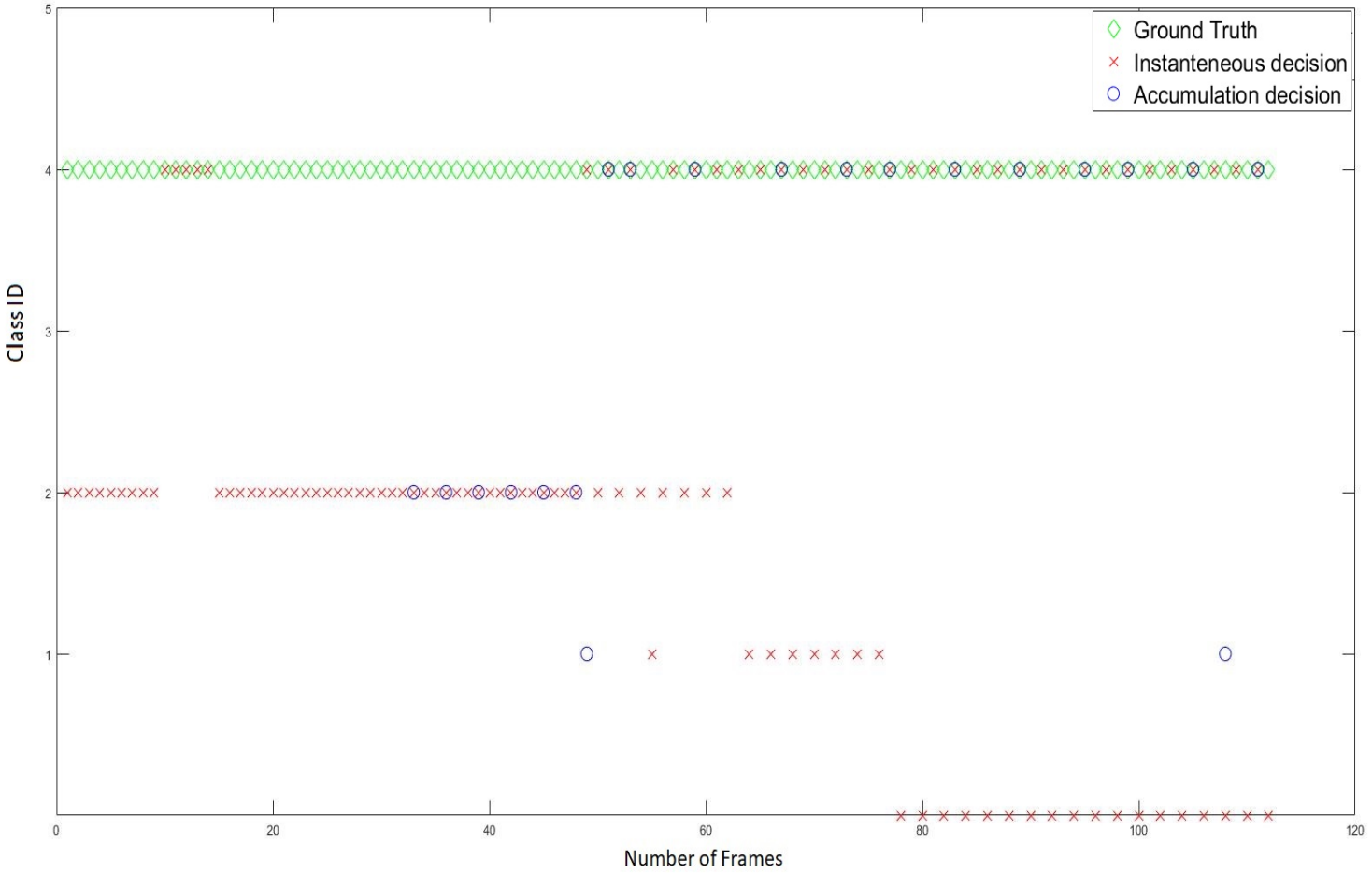


Figure 4.6. Accumulated knowledge improves instantaneous decisions.

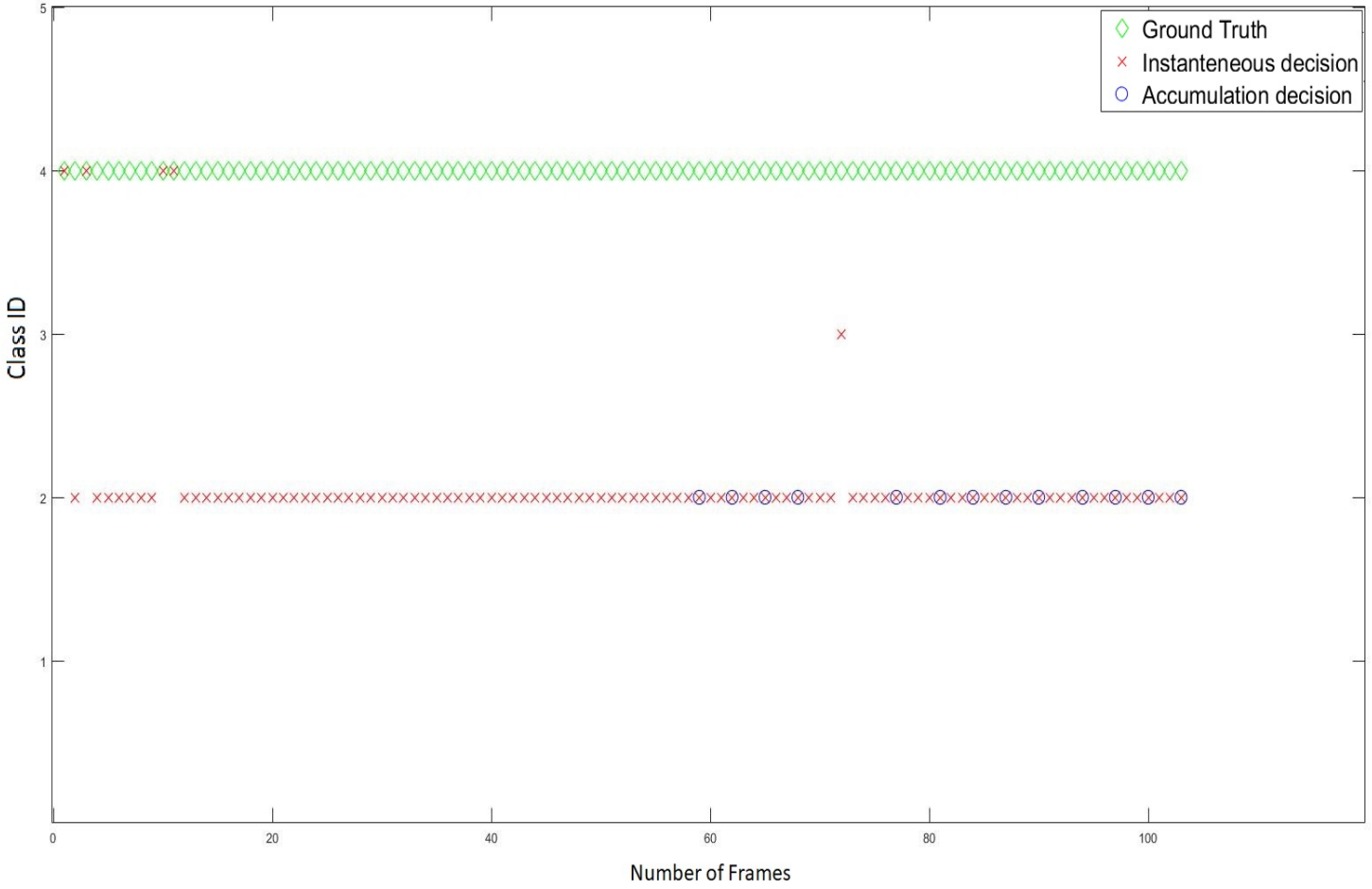
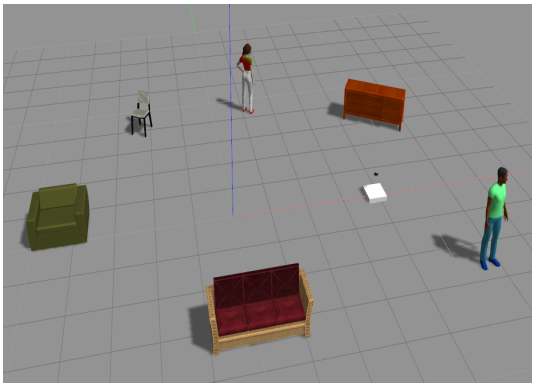


Figure 4.7. Both decisions are false throughout the track.

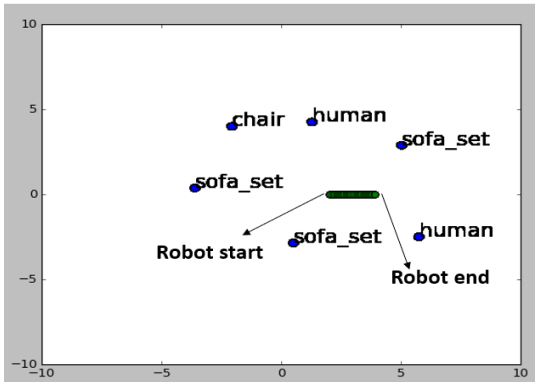
Table 4.2. Precision Recall F-1 scores for different decision models

Measure	Precision	Recall	F_1
Measure-1	0.8269	0.7917	0.7818
Measure-2	0.9036	0.8258	0.8125
Measure-3	0.8906	0.8113	0.7965
Measure-4	0.9214	0.8376	0.8172
Measure-5	0.9169	0.8415	0.8203

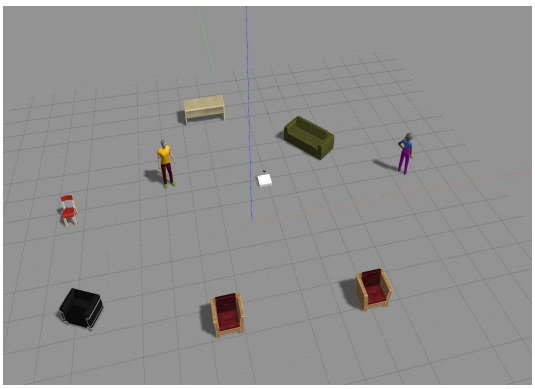
Precision, recall, and F_1 rates are computed for each measure. The results are presented in Table 4.2. It is observed that with the T-DSA descriptor, performance improves considerably. The decision reaches the highest performance with the Measure-5 decisions that are constructed with all accumulation information and all instantaneous information in a sequence.



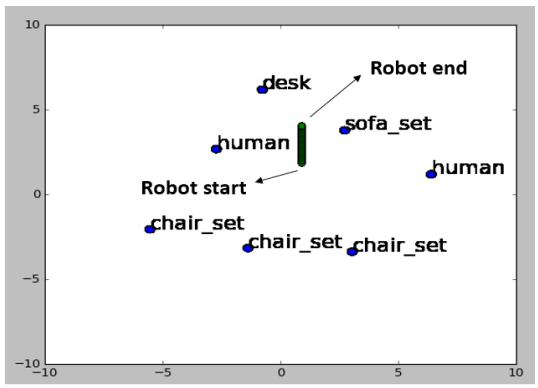
(a) Simulation scene-3



(b) Map for simulation scene-3



(c) Simulation scene-8



(d) Map for simulation scene-8

Figure 4.8. Two examples from simulation scenes

After that, we test our approach for various Gazebo scenes by using Measure-5 decision for object label. Two map examples for simulation scenes are given in Figure 4.8. It can be easily seen that combining instantaneous DSA with T-DSA works well for mapping. To map each object in the scene, the T-DSA should be calculated for the object, and sometimes segmentation errors can cause a lag for T-DSA calculation. This circumstance can be seen in Figure 4.8d. The overall performance in eight simulation scene is presented in Table 4.3. There are some similar objects in terms of depth manner, such as desk and sofa. For this reason, the proposed approach often confuses desk with sofa set. This problem should be solved adding more different objects in the data set.

Table 4.3. Simulation mapping results

Scene Number	Chair		Chair set		Sofa set		Human		Desk	
	Expected	Found	Expected	Found	Expected	Found	Expected	Found	Expected	Found
1	2	2	1	1	0	0	1	1	0	0
2	0	1	1	0	3	3	2	2	0	0
3	1	1	0	0	1	3	2	2	1	0
4	3	3	1	0	0	0	4	4	0	0
5	1	1	1	1	1	1	2	2	1	1
6	1	1	0	0	1	3	2	2	3	1
7	3	3	1	1	2	2	3	3	0	0
8	1	0	3	3	1	1	2	2	1	1

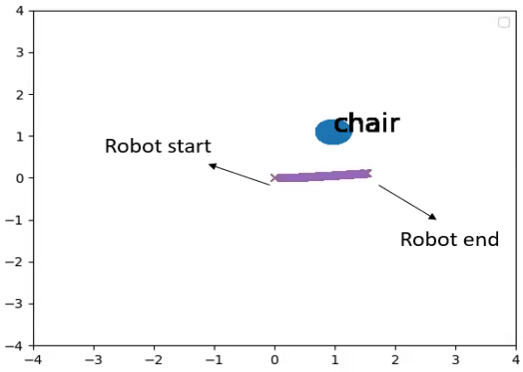
4.7.2. Real Life Results

The proposed approach has also been tested with a real mobile robot. For this, a Kobuki-Turtlebot endowed with a Velodyne VLP-16 sensor has been used. In these experiments, the robot uses directly uses the object models as learned in the simulation experiments. The robot is made to navigate in four different scenes with varying number of objects. Two sample scenes are shown in Figure 4.9. The task is simplified so that the robot classifies only nearby objects. Hence, the robot considers point cloud data only within 2 meters of distance. The results are given in Table 4.4. It is observed that both recall and precision decreases compared to the simulation results. This is partly attributed to the fact that the robot uses models trained in simulation.

Table 4.4. Real life mapping results

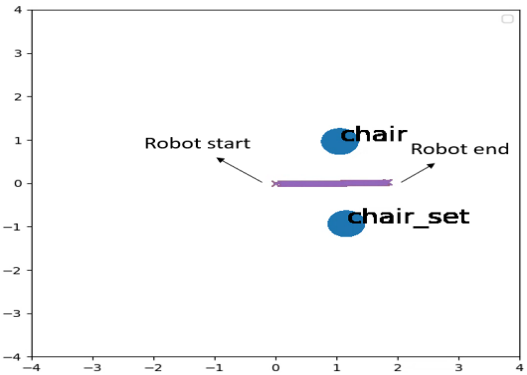
Scene Number	Chair		Chair set		Sofa set		Human		Desk		Robot Movement
	Expected	Found	Expected	Found	Expected	Found	Expected	Found	Expected	Found	
1	1	1	0	0	0	0	0	0	0	0	1.72m
2	2	1	0	1	0	0	0	0	0	0	1.87m
3	2	2	0	0	0	0	1	0	0	0	1.41m
4	0	0	0	0	1	1	1	1	0	0	1.64m

(a) Real scene-1



(b) Map for real scene-1

(c) Real scene-2



(d) Map for real scene-2

Figure 4.9. Two examples from Gazebo scenes

5. CONCLUSION AND FUTURE WORK

The main focus of the thesis is to improve the depth data interpretation capability of a mobile robot. This is a complex task as it requires the robot to address three important problems - namely segmentation, point cloud object representation, and using all the information available. Thanks to the progress in 3D sensing technologies, 3D sensors are being widely used for this purpose. Some of these sensors, such as Kinect, generates dense point cloud data while others, such as LIDAR, generate sparse point cloud data. Each has its own advantages concerning factors such as field of view, available data, and cost. Hence, the proposed approaches must be capable of working with both types of data.

The first contribution of the thesis pertains to finding object candidates. It has been shown that extracting meaningful information from sparse data tends to be difficult. Hence, the proposed approach must be applicable with sparse point cloud data. For this, a novel approach is presented. Differing from the previous works, the proposed segmentation method is carried out in spherical coordinates. This enables the robot to set the segmentation parameters based on the scan parameters of the sensor. We evaluate this method with LIDAR data from indoor and outdoor settings and show that the proposed approach can be used to segment sparse point cloud data.

Next, we consider the representation of point cloud objects. The representation is critical to both object learning and recognition. It must be invariant to pose changes as much as possible and have low complexity. For this, we propose the deformed sphere approximation (DSA) descriptor. The DSA descriptor satisfies these important properties. Furthermore, it can represent both dense and sparse point cloud data. Our experimental results show that our descriptor reaches the best classification performance for LIDAR data, and for Kinect and CAD data, its performance is comparable.

Finally, we consider the extension of the DSA descriptor to encode temporal data. This is because the robot's analysis of its surroundings cannot be based purely on instantaneous data. Rather the incoming data continual and temporal coherency of the data should also be considered in its reasoning. In particular, if the robot is navigating in its environment, it can accumulate the objects' sensory data. Hence, the representation must be capable of encoding accumulated data. For this, we propose the temporal DSA (T-DSA) descriptor. The T-DSA descriptor encodes the accumulated knowledge of objects as the robot is moving around. It is based on tracking the detected object candidates and accumulate information based on these tracks. For tracking, a novel method that combines Kalman filtering with point cloud segment matching is developed. The sensory data coming from each track is then accumulated with the T-DSA descriptor. Rather than evaluating objects only using instant views, our experimental results show that robot creates more reliable object classification results by using sequential object information.

There can be two extensions as future work. The first is to learn the class information considering contextual information among objects. The second one is to use the enhanced class information in place recognition and localization.

REFERENCES

1. Douillard, B., J. Underwood, N. Kuntz, V. Vlaskine, A. Quadros, P. Morton and A. Frenkel, "On the segmentation of 3D LIDAR point clouds", *IEEE International Conference on Robotics and Automation*, pp. 2798–2805, IEEE, 2011.

2. Rabbani, T., F. Van Den Heuvel and G. Vosselmann, "Segmentation of point clouds using smoothness constraint", *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. 36, No. 5, pp. 248–253, 2006.

3. Rusu, R. B., *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*, Ph.D. Thesis, Computer Science department, Technische Universitaet Muenchen, Germany, October 2009.

4. Himmelsbach, M., F. V. Hundelshausen and H.-J. Wuensche, "Fast segmentation of 3D point clouds for ground vehicles", *IEEE Intelligent Vehicles Symposium*, pp. 560–565, IEEE, 2010.

5. Luo, Z., M. V. Mohrenschildt and S. Habibi, "A Probability Occupancy Grid Based Approach for Real-Time LiDAR Ground Segmentation", *IEEE Transactions on Intelligent Transportation Systems*, Vol. 21, No. 3, pp. 998–1010, 2020.

6. Zermas, D., I. Izzat and N. Papanikolopoulos, "Fast segmentation of 3d point clouds: A paradigm on lidar data for autonomous vehicle applications", *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5067–5073, IEEE, 2017.

7. Korchev, D., S. Cheng, Y. Owechko *et al.*, "On real-time lidar data segmentation and classification", *Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV)*, p. 1, The Steering Committee of The World Congress in Computer Science, 2013.

8. Behley, J., V. Steinhage and A. B. Cremers, "Laser-based segment classification using a mixture of bag-of-words", *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4195–4200, IEEE, 2013.

9. Pauling, F., M. Bosse and R. Zlot, "Automatic segmentation of 3d laser point clouds by ellipsoidal region growing", *Australasian Conference on Robotics and Automation*, Vol. 10, 2009.

10. Moosmann, F., O. Pink and C. Stiller, "Segmentation of 3D lidar data in non-flat urban environments using a local convexity criterion", *IEEE Intelligent Vehicles Symposium*, pp. 215–220, IEEE, 2009.

11. Choe, Y., S. Ahn and M. J. Chung, "Fast point cloud segmentation for an intelligent vehicle using sweeping 2D laser scanners", *9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pp. 38–43, IEEE, 2012.

12. Choe, Y., S. Ahn and M. J. Chung, "Online urban object recognition in point clouds using consecutive point information for urban robotic missions", *Robotics and Autonomous Systems*, Vol. 62, No. 8, pp. 1130–1152, 2014.

13. Klasing, K., D. Wollherr and M. Buss, "A clustering method for efficient segmentation of 3D laser data", *IEEE international conference on robotics and automation*, pp. 4043–4048, IEEE, 2008.

14. Wang, D. Z., I. Posner and P. Newman, "What could move? finding cars, pedestrians and bicyclists in 3d laser data", *IEEE International Conference on Robotics and Automation*, pp. 4038–4044, IEEE, 2012.

15. Held, D., D. Guillory, B. Rebsamen, S. Thrun, S. Savarese, R. Holladay, S. Javdani, A. Dragan, S. Srinivasa, O. Salzman *et al.*, "A Probabilistic Framework for Real-time 3D Segmentation using Spatial, Temporal, and Semantic Cues", *Journal Article*, Vol. 32, No. 3, pp. 473–483, 2016.

16. Bogoslavskyi, I. and C. Stachniss, "Fast range image-based segmentation of sparse 3D laser scans for online operation", *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 163–169, IEEE, 2016.

17. Bogoslavskyi, I. and C. Stachniss, "Efficient online segmentation for sparse 3d laser scans", *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, Vol. 85, No. 1, pp. 41–52, 2017.

18. Shi, S., X. Wang and H. Li, "Pointrcnn: 3d object proposal generation and detection from point cloud", *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 770–779, 2019.

19. Yan, Y., Y. Mao and B. Li, "Second: Sparsely embedded convolutional detection", *Sensors*, Vol. 18, No. 10, p. 3337, 2018.

20. Velodyne, *Velodyne Lidar - VLP 16 User Manual, 63-9243 Rev.D*, `https://greenvalleyintl.com/wp-content/uploads/2019/02/Velodyne-LiDAR-VLP-16-User-Manual.pdf`, accessed in January 2021.

21. Geiger, A., "Are we ready for autonomous driving? The KITTI vision benchmark suite", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3354–3361, 2012.

22. Han, X., J. S. Jin, J. Xie, M. Wang and W. Jiang, "A comprehensive review of 3D point cloud descriptors", *CoRR*, Vol. abs/1802.02297, 2018, `http://arxiv.org/abs/1802.02297`.

23. Rostami, R., F. S. Bashiri, B. Rostami and Z. Yu, "A Survey on Data-Driven 3D Shape Descriptors", *Computer Graphics Forum*, Vol. 38, No. 1, pp. 356–393, 2019.

24. Su, J.-C., M. Gadelha, R. Wang and S. Maji, "A Deeper Look at 3D Shape Classifiers", *European Conference on Computer Vision*, pp. 645–661, Springer, 2018.

25. Georgiou, T., Y. Liu, W. Chen and M. Lew, "A survey of traditional and deep learning-based feature descriptors for high dimensional data in computer vision", *Int'l J. of Multimedia Information Retrieval*, Vol. 9, pp. 135–170, 2020.

26. Chen, C., G. Li, R. Xu, T. Chen, M. Wang and L. Lin, "Clusternet: Deep hierarchical cluster network with rigorously rotation-invariant representation for point cloud analysis", *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 4994–5002, 2019.

27. Yang, Y., C. Feng, Y. Shen and D. Tian, "Foldingnet: Point cloud auto-encoder via deep grid deformation", *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 206–215, 2018.

28. Achlioptas, P., O. Diamanti, I. Mitliagkas and L. Guibas, "Learning representations and generative models for 3d point clouds", *International Conference on Machine Learning*, pp. 40–49, PMLR, 2018.

29. Wu, Z., S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang and J. Xiao, "3d shapenets: A deep representation for volumetric shapes", *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 1912–1920, 2015.

30. Qi, C. R., H. Su, K. Mo and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation", *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 652–660, 2017.

31. Zhi, S., Y. Liu, X. Li and Y. Guo, "LightNet: a lightweight 3D convolutional neural network for real-time 3D object recognition", *Proceedings of the Workshop on 3D Object Retrieval*, pp. 9–16, 2017.

32. Simonovsky, M. and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs", *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 3693–3702, 2017.

33. Kumawat, S. and S. Raman, "LP-3DCNN: Unveiling Local Phase in 3D Convolutional Neural Networks", *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 4903–4912, 2019.

34. Mescheder, L., M. Oechsle, M. Niemeyer, S. Nowozin and A. Geiger, "Occupancy Networks: Learning 3D Reconstruction in Function Space", *IEEE Conf. on Computer Vision and Pattern Recognition*, 2019.

35. Park, J. J., P. Florence, J. Straub, R. A. Newcombe and S. Lovegrove, "DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation", *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 165–174, Computer Vision Foundation, 2019.

36. Tatarchenko, M., S. R. Richter, R. Ranftl, Z. Li, V. Koltun and T. Brox, "What Do Single-view 3D Reconstruction Networks Learn?", *CoRR*, Vol. abs/1905.03678, 2019.

37. Johnson, A. and M. Hebert, "Surface matching for object recognition in complex three-dimensional scenes", *Image and Vision Computing*, Vol. 16, No. 9, pp. 635 – 651, 1998.

38. Guo, Y., F. Sohel, M. Bennamoun, M. Lu and J. Wan, "Rotational Projection Statistics for 3D Local Surface Description and Object Recognition", *International Journal of Computer Vision*, Vol. 105, No. 1, p. 63, 2013.

39. Rusu, R. B., A. Holzbach, M. Beetz and G. Bradski, "Detecting and segmenting objects for mobile manipulation", *IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pp. 47–54, IEEE Computer Society, 2009.

40. Qi, C. R., L. Yi, H. Su and L. J. Guibas, "PointNet++: Deep Hierarchical Feature

Learning on Point Sets in a Metric Space", *arXiv preprint arXiv:1706.02413*, 2017.

41. Maturana, D. and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition", *IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, pp. 922–928, IEEE, 2015.

42. Sedaghat, N., M. Zolfaghari, E. Amiri and T. Brox, "Orientation-boosted voxel nets for 3d object recognition", *arXiv preprint arXiv:1604.03351*, 2016.

43. Wang, P., Y. Liu, Y. Guo, C. Sun and X. Tong, "O-CNN: Octree-based Convolutional Neural Networks for 3D Shape Analysis", *CoRR*, Vol. abs/1712.01537, 2017, `http://arxiv.org/abs/1712.01537`.

44. Feng, Y., Y. Feng, H. You, X. Zhao and Y. Gao, "MeshNet: Mesh Neural Network for 3D Shape Representation", *CoRR*, Vol. abs/1811.11424, 2018.

45. Esteves, C., C. Allen-Blanchette, A. Makadia and K. Daniilidis, "Learning SO(3) Equivariant Representations with Spherical CNNs", *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

46. Rao, Y., J. Lu and J. Zhou, "Spherical Fractal Convolutional Neural Networks for Point Cloud Recognition", *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pp. 452–460, 2019.

47. Su, H., S. Maji, E. Kalogerakis and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition", *IEEE Int'l Conf. on Computer Vision*, pp. 945–953, 2015.

48. Mitchell, E., K. S. Engin, V. Isler and D. D. Lee, "Higher-Order Function Networks for Learning Composable 3D Object Representations", *CoRR*, Vol. abs/1907.10388, 2019.

49. Gezawa, A. S., Y. Zhang, Q. Wang and L. Yunqi, "A Review on Deep Learning

Approaches for 3D Data Representations in Retrieval and Classifications", *IEEE Access*, Vol. 8, pp. 57566–57593, 2020.

50. Rusu, R. B., G. Bradski, R. Thibaux and J. Hsu, "Fast 3d recognition and pose using the viewpoint feature histogram", *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2155–2162, IEEE, 2010.

51. Wohlkinger, W. and M. Vincze, "Ensemble of shape functions for 3d object classification", *IEEE Int'l Conference on Robotics and Biomimetics*, pp. 2987–2992, IEEE, 2011.

52. Hana, X.-F., J. S. Jin, J. Xie, M.-J. Wang and W. Jiang, "A comprehensive review of 3d point cloud descriptors", *arXiv preprint arXiv:1802.02297*, 2018.

53. De Deuge, M., A. Quadros, C. Hung and B. Douillard, "Unsupervised feature learning for classification of outdoor 3d scans", *Australasian Conference on Robitics and Automation*, Vol. 2, p. 1, 2013.

54. Chen, T., B. Dai, D. Liu and J. Song, "Performance of global descriptors for velodyne-based urban object recognition", *IEEE Intelligent Vehicles Symposium Proceedings*, pp. 667–673, IEEE, 2014.

55. Shilane, P., P. Min, M. Kazhdan and T. Funkhouser, "The Princeton Shape Benchmark", *Proceedings of the Shape Modeling International 2004*, p. 167–178, 2004.

56. Tangelder, H. W. H. and R. C. Veltkamp, "A survey of content based 3D shape retrieval methods", *Multimed Tools Appl*, Vol. 39, pp. 441–471, 2008.

57. Driscoll, J. R. and D. M. Healy, "Computing Fourier Transforms and Convolutions on the 2-Sphere", *Adv. Appl. Math.*, Vol. 15, No. 2, p. 202–250, 1994.

58. Burel, G. and H. Henocq, "Three-dimensional invariants and their application to object recognition", *Signal processing*, Vol. 45, No. 1, pp. 1–22, 1995.

59. Staib, L. H. and J. S. Duncan, "Model-based deformable surface finding for medical images", *IEEE Transactions on Medical Imaging*, Vol. 15, No. 5, pp. 720–731, 1996.

60. Kazhdan, M., T. Funkhouser and S. Rusinkiewicz, "Rotation Invariant Spherical Harmonic Representation of 3D Shape Descriptors", L. Kobbelt, P. Schroeder and H. Hoppe (Editors), *Eurographics Symposium on Geometry Processing*, The Eurographics Association, 2003.

61. Zhi, S., Y. Liu, X. Li and Y. Guo, "Toward real-time 3D object recognition: A lightweight volumetric CNN framework using multitask learning", *Computers & Graphics*, Vol. 71, pp. 199–207, 2018.

62. Lan, S., R. Yu, G. Yu and L. S. Davis, "Modeling local geometric structure of 3d point clouds using geo-cnn", *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 998–1008, 2019.

63. Erkent, Ö. and H. I. Bozma, "Bubble space and place representation in topological maps", *The Int'l J. of Robotics Research*, Vol. 32, No. 6, pp. 672–689, 2013.

64. Tolstov, G. P., *Fourier Series*, Prentice-Hall, 1962.

65. Shen, L., H. Farid and M. McPeek, "Modeling three-dimensional morphological structures using spherical harmonics", *Evolution; International Journal of Organic Evolution*, Vol. 63, No. 4, pp. 1003–1016, 4 2009.

66. Lai, K., L. Bo, X. Ren and D. Fox, "A large-scale hierarchical multi-view rgb-d object dataset", *IEEE international conference on robotics and automation*, pp. 1817–1824, IEEE, 2011.

67. Qi, C. R., L. Yi, H. Su and L. J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space", *CoRR*, Vol. abs/1706.02413, 2017, `http://arxiv.org/abs/1706.02413`.

68. Katz, S., A. Tal and R. Basri, "Direct Visibility of Point Sets", *ACM Trans. Graph.*, Vol. 26, No. 3, p. 24–1:10, Jul. 2007, https://doi.org/10.1145/1276377.1276407.

69. Zheng, Z., Y. Li and W. Jun, "LiDAR point cloud registration based on improved ICP method and SIFT feature", *2015 IEEE International Conference on Progress in Informatics and Computing (PIC)*, pp. 588–592, IEEE, 2015.

70. Cheng, L., S. Chen, X. Liu, H. Xu, Y. Wu, M. Li and Y. Chen, "Registration of laser scanning point clouds: A review", *Sensors*, Vol. 18, No. 5, p. 1641, 2018.

71. Myronenko, A. and X. Song, "Point set registration: Coherent point drift", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 32, No. 12, pp. 2262–2275, 2010.

72. Tamaki, T., M. Abe, B. Raytchev and K. Kaneda, "Softassign and em-icp on gpu", *First International Conference on Networking and Computing*, pp. 179–183, IEEE, 2010.

73. Karunasekera, H., H. Wang and H. Zhang, "Multiple object tracking with attention to appearance, structure, motion and size", *IEEE Access*, Vol. 7, pp. 104423–104434, 2019.

74. Osep, A., W. Mehner, M. Mathias and B. Leibe, "Combined image-and world-space tracking in traffic scenes", *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1988–1995, IEEE, 2017.

75. Wojke, N., A. Bewley and D. Paulus, "Simple online and realtime tracking with a deep association metric", *IEEE international conference on image processing (ICIP)*, pp. 3645–3649, IEEE, 2017.

76. Weng, X., J. Wang, D. Held and K. Kitani, "3d multi-object tracking: A baseline and new evaluation metrics", *arXiv preprint arXiv:1907.03961*, 2020.

77. Chiu, H.-k., A. Prioletti, J. Li and J. Bohg, "Probabilistic 3d multi-object tracking for autonomous driving", *arXiv preprint arXiv:2001.05673*, 2020.

78. Moosmann, F. and C. Stiller, "Joint self-localization and tracking of generic objects in 3D range data", *IEEE International Conference on Robotics and Automation*, pp. 1146–1152, IEEE, 2013.

79. Kaestner, R., J. Maye, Y. Pilat and R. Siegwart, "Generative object detection and tracking in 3d range data", *IEEE International Conference on Robotics and Automation*, pp. 3075–3081, IEEE, 2012.

80. Dewan, A., T. Caselitz, G. D. Tipaldi and W. Burgard, "Motion-based detection and tracking in 3d lidar scans", *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4508–4513, IEEE, 2016.

81. Weng, X., Y. Wang, Y. Man and K. Kitani, "GNN3DMOT: Graph Neural Network for 3D Multi-Object Tracking with Multi-Feature Learning", *arXiv preprint arXiv:2006.07327*, 2020.

82. Zhang, W., H. Zhou, S. Sun, Z. Wang, J. Shi and C. C. Loy, "Robust multi-modality multi-object tracking", *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2365–2374, 2019.

83. Luo, W., B. Yang and R. Urtasun, "Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3569–3577, 2018.

84. Li, X. and J. Guivant, "Efficient and accurate object detection with simultaneous classification and tracking", *arXiv preprint arXiv:2007.02065*, 2020.

85. Fan, H. and Y. Yang, "PointRNN: Point recurrent neural network for moving point cloud processing", *arXiv preprint arXiv:1910.08287*, 2019.

86. Huang, R., W. Zhang, A. Kundu, C. Pantofaru, D. A. Ross, T. Funkhouser and A. Fathi, "An LSTM Approach to Temporal 3D Object Detection in LiDAR Point Clouds", *arXiv preprint arXiv:2007.12392*, 2020.

87. Choy, C., J. Gwak and S. Savarese, "4d spatio-temporal convnets: Minkowski convolutional neural networks", *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3075–3084, 2019.

88. Stutz, D. and A. Geiger, "Learning 3d shape completion under weak supervision", *International Journal of Computer Vision*, Vol. 128, No. 5, pp. 1162–1181, 2020.

89. Giancola, S., J. Zarzar and B. Ghanem, "Leveraging shape completion for 3d siamese tracking", *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1359–1368, 2019.

90. Kalman, R. E., "A new approach to linear filtering and prediction problems", *Transactions of the ASME–Journal of Basic Engineering*, Vol. 82, pp. 35–45, 1960.

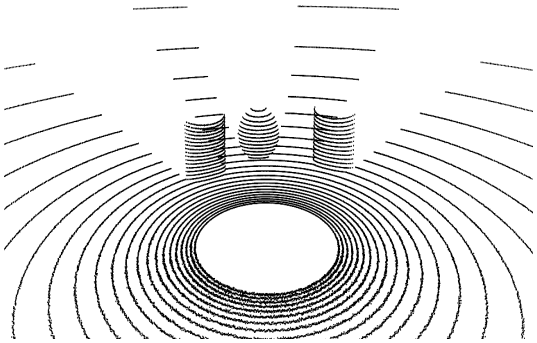
91. Bentley, J. L., "Multidimensional binary search trees used for associative searching", *Communications of the ACM*, Vol. 18, No. 9, p. 509–517, 1975.

92. Förstner, W. and B. Moonen, "A metric for covariance matrices", *Geodesy-the Challenge of the 3rd Millennium*, pp. 299–309, Springer, 2003.

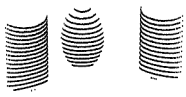
93. Rasouli, A. and J. K. Tsotsos, "The effect of color space selection on detectability and discriminability of colored objects", *arXiv preprint arXiv:1702.05421*, 2017.

APPENDIX A: GROUND SEGMENTATION

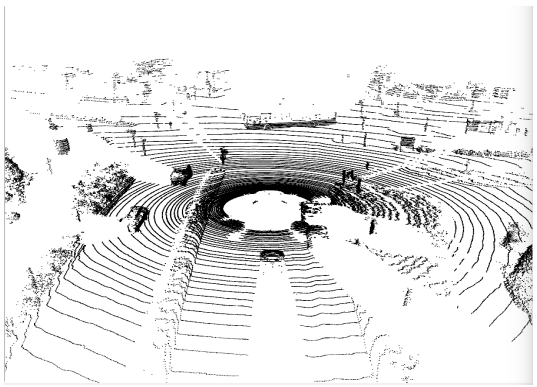
This section explains the ground segmentation method [6]. In this approach, the points in the point cloud are sorted in ascending order according to their height. Then, a certain number of seed points in the sorted point cloud are selected. As LIDAR data tends to be noisy, the nosily sensed points needs to be eliminated. This is done by taking account the height of the sensor. Each points in the point cloud is tested using the height of the seed points and a given threshold for the seed height. If a point is below the calculated height, it will be taken as a primitive ground plane. Using these points a ground plane is estimated, and the estimation of the ground plane is enhanced iteratively by using detected ground points. Two ground extraction examples from velodyne simulator and KITTI tracking dataset are shown in Figure A.1



(a) Raw simulator data



(b) Ground segmented simulator data



(c) Raw KITTI data

(d) Ground segmented KITTI data

Figure A.1. Ground segmentation examples

APPENDIX B: USER GUIDE

This section explains the hardware and software developed.

B.1. Hardware

During the real life experiments Kobuki Turtlebot is used as shown in the FigureB.1. The robot uses a Velodyne VLP16 and a NVIDIA Jetson Xavier.

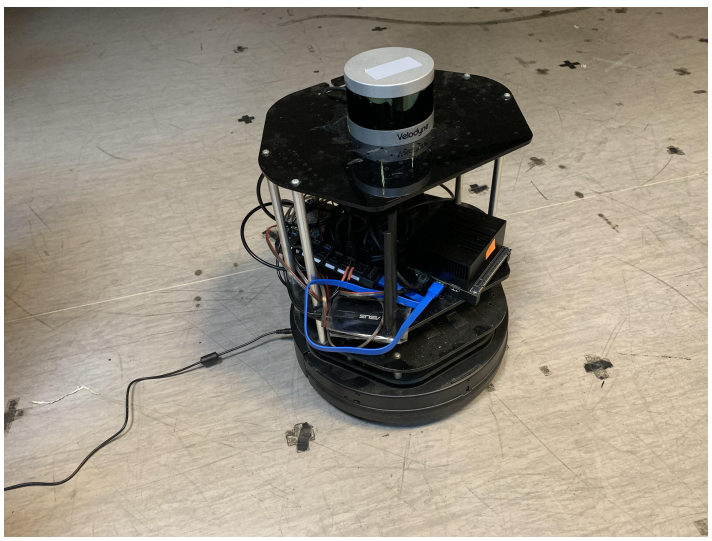


Figure B.1. Kobuki Turtlebot

B.2. Software

B.2.1. Software Requirements

The requirements to run the all algorithms are as follows:

- Ubuntu 16.04 or Ubuntu 18.04
- ROS Kinetic or Melodic
- Frugally-deep

B.2.2. Running software on the robot

NVIDIA Jetson Xavier and Velodyne VLP16 should be connected to the power using external power outputs of Kobuki Turtlebot. Then, in order to use the developed software, the following steps need to be followed:

- Open a new terminal and execute the command:
  roslaunch velodyne_pointcloud VLP16_points.launch

- To filter the noise in point cloud points, execute the command:
  rosrun lidar_process statistical_outliers_removal

- To segment the ground plane from the velodyne scan, execute the command:
  rosrun lidar_process ground_segmentation

- To run DBS segmentation, execute the command:
  rosrun lidar_process segmentation_image

- To publish robot odometry, execute the command:
  rosrun lidar_process pose_taker

- Tracking, class prediction and decision processes can be run via this command:
  rosrun tracker tracker

- To visualize the constructed map, execute this command:
  rosrun tracker map.py

- To move the robot, follow this command:
  rosrun tracker move_velodyne.py