Avrupa Bilim ve Teknoloji Dergisi
Özel Sayı, S. 513-529, Ekim 2019
© Telif hakkı EJOSAT'a aittir
Araştırma Makalesi

www.ejosat.com ISSN:2148-2683

European Journal of Science and Technology
Special Issue, pp. 513-529, October 2019
Copyright © 2019 EJOSAT
Research Article

Simulation of Lidar-Based Robot Detection Task using ROS and Gazebo

Zahir Yılmaz[1*], Levent Bayındır[2]

[1] Ataturk University, Department of Computer Engineering, Erzurum, Turkey (ORCID: 0000-0002-5009-6763)
[2] Ataturk University, Department of Computer Engineering, Erzurum, Turkey (ORCID: 0000-0001-7318-5884)

(This publication has been presented orally at HORA congress.)

(First received 1 August 2019 and in final form 25 October 2019)

(**DOI:** 10.31590/ejosat.642840)

ATIF/REFERENCE: Yılmaz, Z., & Bayındır, L. (2019). Simulation of Lidar-Based Robot Detection Task using ROS and Gazebo. *European Journal of Science and Technology*, (17), 513-529.

Abstract

In the last few decades, the robotics world has seen great progress at all levels, from personal assistant robots to multi-robotic and intelligent swarm systems. Simulation platforms play a critical role in this improvement due to efficiency, flexibility, and fault tolerance they provide during the cycles of developing and testing new strategies and algorithms. In this paper, we model a new mobile robot equipped with a 2D Lidar using Robot Operating System (ROS) and use this robot model to develop a robot detection method in Gazebo simulation environment. Detecting surrounding objects and distinguishing robots from these objects (kin detection) are essential in multi-robot and swarm robotic applications. In this paper, we use Lidar to handle this task by applying the following steps: (1) acquisition of laser data and pre-processing, (2) segmentation of data using the point-distance-based segmentation method, (3) classification of segments by applying two levels of filtering: filtering by segment diameter which aims to eliminate segments that don't fit a certain size (Lidar size) using features for each segment, filtering by segment shape to check remaining segments to test if they fit the Lidar's shape (which is a circle with known radius) or not by using the circle fitting method and (4) identify the position of kin relative to the observer robot. Two different scenarios are discussed in the experiments section. In the first scenario, many cylindrical objects with radius different from the robot's Lidar were used in addition to a robot, and thus objects are distinguished from the robot in the first level of filtering without using the second one which may be a complex operation. In the second scenario, various objects with a similar radius were used, and due to the similarity in the radius between the Lidar and the objects, it was necessary to apply all the method's steps to detect the kin robots. It was noticed from experiments that the accuracy of the results depends on two main factors: the distance between the observer robot and other objects or robots and the amount of noise in the Lidar measurements.

Keywords: Robot detection, Kin detection, Lidar, ROS, Gazebo, Robotics.

* Corresponding Author: Ataturk University, Department of Computer Engineering, Erzurum, Turkey, ORCID: 0000-0002-5009-6763, yilmaz92.tr@gmail.com

ROS ve Gazebo Kullanarak Lidar Tabanlı Robot Tespit Görevinin Simülasyonu

Öz

Son birkaç on yılda, robotik dünyası, kişisel asistan robotlardan çoklu robotik ve akıllı sürü sistemlerine kadar her seviyede büyük ilerleme gördü. Simülasyon platformları, yeni stratejiler ve algoritmalar geliştirme ve test etme döngüleri sırasında sağladıkları verimlilik, esneklik ve hata toleransı nedeniyle bu gelişmede kritik bir rol oynamaktadır. Bu makalede, Robot İşletim Sistemi'ni (ROS) kullanarak 2D lazerli telemetre ile donatılmış yeni bir mobil robot modelliyoruz ve bu robot modelini Gazebo simülasyon ortamında bir robot algılama yöntemi geliştirmek için kullanıyoruz. Çevredeki nesneleri algılamak ve robotları bu nesnelerden ayırt etmek (kin algılama), çoklu robot ve sürü robotik uygulamalarda çok önemlidir. Bu makalede, robot algılama görevini yerine getirmek için Lidar kullanılarak aşağıdaki adımları uygulamaktadır: (1) lazer verilerinin elde edilmesi ve ön işleme, (2) nokta-mesafeye dayalı segmentasyon yöntemini kullanarak verinin segmentlere ayrılması, (3) iki filtreleme seviyesi uygulayarak segmentlerin sınıflandırılması: her bölüm için özellikler kullanarak belirli bir boyuta (Lidar boyutu) uymayan bölümleri ortadan kaldırmayı amaçlayan bölüm çapına göre filtreleme yapmak, kalan parçaları Lidar'ın şekline uyup uymadığını (bilinen yarıçapı olan bir daire) test etmek için segment şekline göre filtreleme ve (4) gözlemci robota göre komşu robotun pozisyonunu tanımlar. Deneyler bölümünde iki farklı senaryo ele alınmıştır. İlk senaryoda, bir robotun yanı sıra, robotun Lidar'ından farklı yarıçapa sahip birçok silindirik nesne kullanılmış ve bu nedenle nesneler, ikinci bir karmaşık işlem olabilecek ikinci filtreyi kullanmadan, ilk filtreleme düzeyinde robottan ayrılmıştır. İkinci senaryoda, benzer yarıçapa sahip çeşitli nesneler kullanıldı ve Lidar ile nesneler arasındaki yarıçaptaki benzerlik nedeniyle, kin robotları tespit etmek için tüm yöntemin adımlarını uygulamak gerekliydi. Deneylerden, sonuçların doğruluğunun iki ana faktöre bağlı olduğu gözlenmiştir: gözlemci robotu ile diğer nesneler veya robotlar arasındaki mesafe ve Lidar ölçümlerindeki gürültü miktarı.

Anahtar Kelimeler: Robot tespiti, Soydaş tespiti, Lidar, ROS, Gazebo, Robotik.

1. Introduction

Robotics may be one of the most common words we frequently hear in the world of technology. Robotics is the science and study of robots, which aims to make machines that are able to perform tasks on human commands or by themselves to make work easier or more productive.

A robot is an integrated system that consists of different types of actuators, sensors, control systems, and software to perform various tasks, whether in our daily life or for the industry. Robotics is an indisciplinary research of physics, mechanical and electrical engineering, and computing science. In general, there are two main types of robots based on the degree of mobility: fixed robots which cannot move with respect to certain components of their environment; mobile robots which can travel in the environment by using various means of locomotion. Nevertheless, nowadays mobile robots are more desirable due to their navigation and sensing capabilities that are useful for many different tasks. Moreover, mobile robots can now work in cooperation to accomplish some critical tasks that are beyond the capabilities of a single robot as the case of multi-robot and swarm robotic systems.

Swarm robotics is an innovative approach inspired from studying the collective behavior of social animals which have the ability to cooperate and coordinate among themselves for solving common problems such as foraging and flocking [1]. Several mobile robot platforms have been developed to study swarm robotic systems, some of which are listed in Table 1. But as is clear from Table1, the price of these robots is relatively high, and the cost will increase dramatically when the number of required robots is high as in the case of swarm robotics. In addition to that, there will also be an operation cost when more than one or two experiments is required to create or evaluate some algorithms. Therefore, it was necessary to look for an alternative solution to reduce the total cost and facilitate the algorithms development process. One possible solution is developing and using robot simulators.

Table 1. Comparison of some swarm robotic platforms

Robot Name	Sensors	Cost (USD)	Diameter	Autonomy
AMiR [2]	Distance, light, bearing	84	6.5 cm	2 h
R-one [3]	Light, IR, gyro, bump, Wi-Fi module	285	10 cm	6 h
Autobot [4]	IR, encoder, radio communication	112	12 cm	2 h
Colias [5]	Distance, light, bump, bearing, range	32	4 cm	1-3 h
Jasmine [6]	Distance, light, bearing	103	3 cm	1-2 h
Kobot [7]	Distance, bearing, vision, compass	1034	12 cm	10 h
E-puck [8]	Distance, camera, bearing, accele, mic	750	7.5 cm	1-10 h
Pi swarm [9]	IR, accelerometer, magnetometer, gyroscope, temperature, light, RF transceiver, LED’s	213	9 cm	2 h

Simulation and virtual reality, became essential tools in many fields of science in recent years, particularly when the process of testing some method or trying to innovate a novel algorithm needs more than one prototype and evaluating these models is costly due to possible failures as in robotics research. Robot simulators have seen considerable development in the last decade, and the list of simulation software and tools can be quiet long, as an instance, Player-Stage-Gazebo (PSG) simulator [31] considered one of the most popular simulators in swarm robotics applications. In fact, the first obstacle that can be faced when we want to deal with like these tools is to decide which the best simulator for our own application. Actually, there is no ideal simulation tool for all scenarios because each of them has advantages and drawbacks, and this issue depends primarily on the kind of application and the required features. But in general context, simulators can be classified under two main categories based on robot prototyping method: (1) Using the own simulator as a creation tool like Marilou [34] and 4DV-Sim [35], (2) Using external tools like Webots [33], V-REP [32], and Gazebo [13]. From another side, simulators differ in degree of complexity, while some simulation software has a user-friendly and relatively simple tools but restricted only to 2D environments, other simulation tools focus more on the accuracy of physical simulation engines and allow users to integrate their own robot models and virtual environments but still a bit complex to deal with [10].

In addition to the emergence of simulators, there were also many efforts to develop some common systems that provide a wide collaboration for developing various robotic applications, and that is what was later referred to as robot software platforms. These platforms can be used alongside simulators, which can form a powerful utility for communicating with them, sending commands to engines, reading sensor data, and responding in accordance with the task assigned to the robot. There is a wide diversity in the robotics industry, which is associated with different hardware platforms. Therefore, having some kind of robot software platform which allows even non-robot field software engineers to develop many robot applications without having expertise in hardware is considered very valuable work as it reduces development time dramatically. So, platforms within the robotics field, have been gaining great attention recently, and today, there are many different kinds of them [11].

Whereas simulators can be used to develop various individual robotics platforms, using them in the field of multi-robot systems remains more common. Many collective behaviors tasks such as aggregation [27], pattern formation [28] and flocking [29] have been developed using simulators but regardless of the type of task required, detecting surrounding objects in the environment and distinguishing robots from these objects are essential for robot’s operations. Several studies have proposed solutions to handle this issue which is often called “kin detection task” using different types of sensor like, infrared [20], ultrasound [21], vision [22], and Lidar [23, 24, 25] sensors. Lidar-based studies use a common approach which depends on using circular robots or any circular object attached to the robot (As in our case where we use a circular-shaped Lidar) and then attempting to detect these robots by taking advantage of circularity and known diameter of these robots or objects attached to them. The main idea of these studies is to find segments from Lidar measurements corresponding objects in the environment and apply one of the circle fitting methods to find robots. In this study, we present kin robot detection task as an example of using the pre-designed mobile robot in multi-robot tasks. All tests have been performed in a simulation environment using ROS and Gazebo with using circular objects which have different diameters from the Lidar and non-circular objects having the same width as the Lidar diameter.

This paper is organized as follows. Section 2 presents the architecture of ROS as a robotics middleware to model robot and to handle sensor data and Gazebo as a simulator framework. In section 3, the process of creating a model and simulate it in Gazebo will be described, followed by implementation of kin detection task in section 4. We end this paper with conclusions and describe future works being planned in the last section.

2. ROS and Gazebo

2.1. ROS Architecture

ROS stands for Robot Operating System, which is an open-source robot software platform that provides services expected from any operating system like hardware abstraction, device control, communication between processes, and file management. It also provides various tools and libraries which give the ability to build, write, debug, and run code across different workstations [12]. ROS is not a real operating system in the conventional sense such as Windows, Linux, and Mac, but it is a meta-operating system that runs over the installed operating system, and it has the ability to perform processes such as scheduling, data transmission/reception, loading, monitoring, and error handling by utilizing virtualization layer between applications and distributed computing resources [11]. In order to achieve that ROS has some distinctive characteristics as follows:

- ***Distributed Structure:*** ROS was built from small software modules called nodes, where each node has the ability to run and exchange data independently with the help of a master node called "roscore" that acts as a lookup table and provide the communication between these nodes.
- ***Package Management:*** In ROS, codes that relate to each other and achieve the same purpose are grouped in the form of packages so they can be easily managed.
- ***Public Repository:*** ROS is an open-source package-based framework, so packages are available for any developer on a public level (e.g., GitHub).

- ***Multi-Lingual:*** ROS platform provides client libraries that support different programming languages such as Python, C++, and Lisp. So, ROS software modules can be written in any language for which a client library has been written. ROS client libraries communicate with one another by following a convention that describes how messages are “flattened” or “serialized” before being transmitted over the network.

2.2. ROS Terminology

ROS has certain concepts that should be familiar to everyone who would like to deal with ROS. These concepts are nodes, messages, topics, services, and actions.

- ***Nodes:*** Node is the smallest software module in ROS; it can be described as one executable program that can run independently and communicate with other nodes to send and receive data by establishing peer-to-peer links. This operation is provided by a core node that gives naming and registration services to allow nodes to exchange data between each other, and this node is called the master node (roscore).
- ***Messages:*** Nodes exchange data among themselves by passing messages which could be either primitive data type such as integer, floating-point, boolean, etc. or composed of other messages such as messages come from sensors like Lidar.

In order to ensure the maximum reusability of codes, ROS is designed in the form of nodes. Therefore, there must be a mechanism for communication between these nodes provided by passing messages. ROS provides three different methods for exchanging data:

- ***Topics:*** This method is considered the base and the most used transmission method. It is an asynchronous unidirectional message transmission/reception method which is used when exchanging data continuously. Messages are sent in turn via topics which represent a channel for a specific kind of data. Communication over topics uses the concept of the publisher (sender node) and subscriber (receiver node), and both of them use the same type of messages by using the same topic name registered in the master node. As topics are unidirectional and remain connected to send or receive messages continuously, it is suitable for sensor data that requires publishing messages periodically.
- ***Services:*** The second type of communication is through services. It is a bidirectional synchronous communication method that depends on request/reply messages where the service client requests some service and the service server responds to the incoming requests. Unlike the topic, the service does not maintain the connection. Therefore, when the request and response of the service are completed, the connection between the two nodes is disconnected. So, it is often used to command a robot to perform a specific action or nodes to perform certain events with a specific condition.
- ***Actions:*** The last type of communication method is to connect using action, which is very similar to the service method, but it is characterized by containing feedback messages that report task state to the client periodically. Therefore, communication over action is used when a requested goal takes long time to complete, and for this reason, progress feedback is necessary.

2.3. Gazebo Simulator

Gazebo is a 3D simulator that provides robots, sensors, environment models for 3D simulation required for robot development, and offers realistic simulation with its physics engine. Gazebo is one of the most popular simulators in recent years and has been selected as the official simulator of the DARPA Robotics Challenge in the US. It is one of the most famous simulators in the field of robotics because of its high performance and various plugins. Moreover, Gazebo is developed and distributed by Open Robotics, which is in charge of ROS and its community, so it is compatible with ROS [13].

Gazebo can work as a stand-alone program, but there is also availability to establish a connection with Gazebo using a different type of Application Programmer Interfaces (APIs) and libraries. ROS is considered one of the most popular API that can be used to connect with Gazebo and which forms a powerful tool in the robotic field. In order to do that, ROS uses a set of packages named gazebo_ros_pkgs that provides wrappers around the stand-alone Gazebo and which can achieve integration with different ROS components. These packages provide the necessary interfaces to simulate a robot in Gazebo using ROS messages, topics, and services and give the ability to communicate with available sensor models included in Gazebo such as sonar, scanning laser range-finders and GPS.

Figure 1 shows an overview of the gazebo_ros_pkgs interface, which is a set of ROS packages that provide the necessary interfaces to simulate a robot in the Gazebo. Some of the packages that exist in this interface are explained below.

- **gazebo_ros:** this package wraps Gazebo server and Gazebo client by using two Gazebo plugins that provide the necessary ROS interface for messages, services, and dynamic reconfiguration.
- **gazebo_plugins:** a ROS package that provides the interface for different type of sensors, motors, and other plugins that are robot-independent Gazebo plugins.

- **gazebo_msgs:** this package is used for interacting with Gazebo from ROS using one of the three message exchanging methods.
- **gazebo_worlds:** this package has been merged to gazebo_ros, it provides a variety of predesigned worlds and model files that can be used in any launch file.
- **gazebo_tests:** this package has been merged to gazebo plugins, it contains a variety of unit tests for gazebo, tools, and plugins.
- **gazebo_api_plugin:** it is a gazebo plugin that provides various tools to deal with gazebo and spawn robot models to Gazebo.

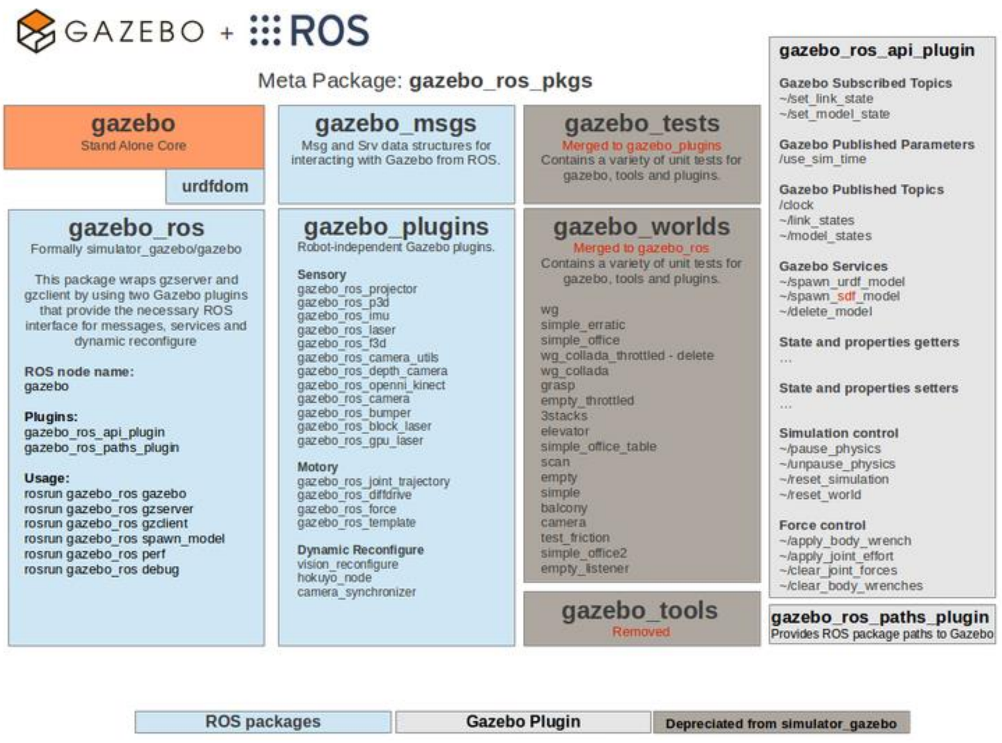


Figure 1. *An overview of the gazebo_ros_pkgs interface. The interface contains several packages such as gazebo_msgs, gazebo_ros, and gazebo_plugins [36].*

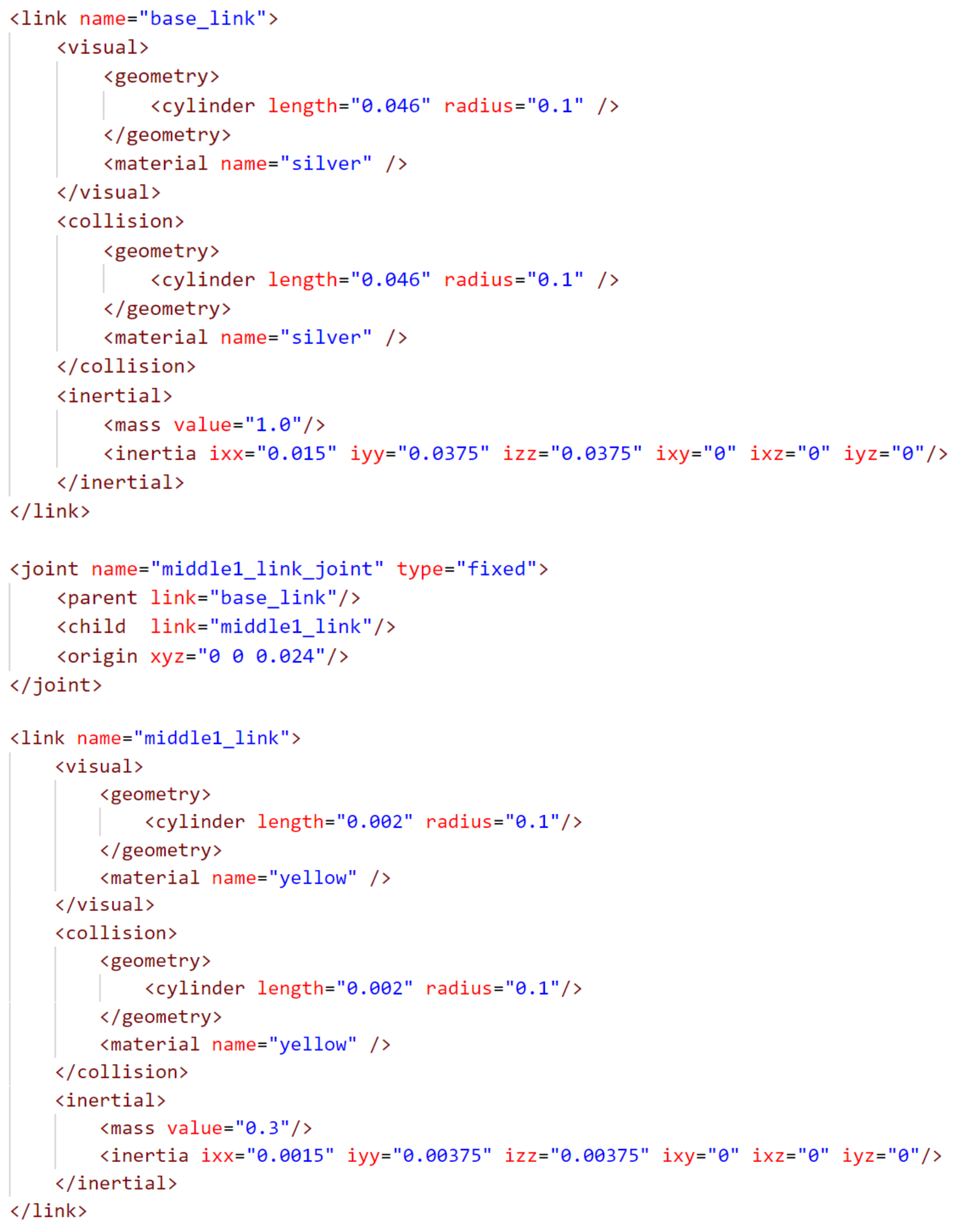
3. Robot Modeling

In this study, we used simulation of an early version of a new robot being designed and developed at Ataturk University for robotic research. We have performed all experiments using ROS and Gazebo simulator. This robot has a main cylindrical body with a height of 8cm and a radius of 9cm; it also has three wheels attached to the main body (right, left and front wheels). In order to apply our research, the robot was equipped with a 2-D lidar attached to the top of the robot.

In the world of robot simulation, it is so common to have the ability to model your own robot, handle various sensor data received from this robot, control the robot by actuators and test or evaluate algorithms. Therefore, there should be a mechanism that allows translating the kinematics model of the robot to a convenient format to deal with this robot's model using many ROS standard tools. ROS provides this feature by XML language where robot models could be described in an XML format called Unified Robot Description Format (URDF), which is expressed as a 3D model for which each model can be moved or operated according to their corresponding degree of freedom, so they can be used for simulation or control.

URDF files describe the physical configuration of the robot, such as how many wheels it has, where they are placed, and which directions they turn in. This information will be used by Rviz (which is the 3D visualization tool of ROS) to visualize the state of the robot [15], by Gazebo to simulate it, and by systems like the navigation stack to make it drive around the world in a purposeful manner [14]. This format is designed to represent a wide variety of robots, from a two-wheeled toy to a walking humanoid but regardless of the complexity of the robot, there are two basic elements to model any robot which are links and joints. Links are the rigid parts of the robot, such as a chassis or a wheel, while joints work to connect these links, defining how they can move with respect to each other. In URDF file, links are represented through <link> and </link> tags which represent one specified part of the robot's model (one link), while <joint> and </joint> tags are used to describe the type of joint between two specified links (parent and child links) [16]. In addition to that, each link or joint has some subtags that can be used to define the characteristics of this link or joint. For instance, the link mass and the amount of inertia over the different axes could be defined inside the link tags. Moreover, the visual appearance of robot models can be greatly improved through the use of high-quality meshes such as STereoLithography (STL) [17] and Collada [18] files which can easily be imported to the URDF file inside the <mesh> tag specified in <geometry> tags inside links. Joints also have some subtags to define the axis related to motion direction, the parent and child links for this joint and the origin of the joint. Figure 2 shows an example of these tags related to two links of the modeled robot and the joint between them.

On the other hand, in order to make simulation as close as possible to reality, there must be a mechanism that provides the robot with the ability to discover the surrounding environment and interact with it which is usually done by sensors and motors. This capability is also available in simulation software, and for our case, ROS has a package called gazebo_ros, which provides many plugins to support the status and control of the robot's motor and sensors. Gazebo plugins allow bidirectional communication between Gazebo and ROS and they support various sensors such as camera, laser, inertial navigation sensor, mobile platform control such as differential motors, skid steering drive, parallel movement, and ROS-Control where simulated sensor and physics data can stream from Gazebo to ROS, and actuator commands can stream from ROS back to Gazebo [19].



```
<link name="base_link">
    <visual>
        <geometry>
            <cylinder length="0.046" radius="0.1" />
        </geometry>
        <material name="silver" />
    </visual>
    <collision>
        <geometry>
            <cylinder length="0.046" radius="0.1" />
        </geometry>
        <material name="silver" />
    </collision>
    <inertial>
        <mass value="1.0"/>
        <inertia ixx="0.015" iyy="0.0375" izz="0.0375" ixy="0" ixz="0" iyz="0"/>
    </inertial>
</link>

<joint name="middle1_link_joint" type="fixed">
    <parent link="base_link"/>
    <child  link="middle1_link"/>
    <origin xyz="0 0 0.024"/>
</joint>

<link name="middle1_link">
    <visual>
        <geometry>
            <cylinder length="0.002" radius="0.1"/>
        </geometry>
        <material name="yellow" />
    </visual>
    <collision>
        <geometry>
            <cylinder length="0.002" radius="0.1"/>
        </geometry>
        <material name="yellow" />
    </collision>
    <inertial>
        <mass value="0.3"/>
        <inertia ixx="0.0015" iyy="0.00375" izz="0.00375" ixy="0" ixz="0" iyz="0"/>
    </inertial>
</link>
```

Figure 2. *A portion of code shows the tags been used related to the "base" and "middle_1" links and the link connects between them*

The essential components (links and joints) for our robot model are listed below and the visual state of these components as a result of interpreting the URDF file using RViz tool is shown in Figure 3:

- The base_link that represents the main chassis of the robot
- The middle1_link that come above the base_link
- A fixed joint that connects base_link with middle1_link
- The middle2_link connected with middle1_link by another fixed joint
- Lastly, we have the upper_link connected with the previous link by another fixed joint, and this link will carry the Lidar sensor above it

Additionally, in order to give the robot the ability to move, we have equipped the robot with three wheels:

- Front-wheel, attached to the front of the main body via caster
- Two rear wheels, attached to the back of chassis
- Revolute joints that connect wheels to the body and give the ability to rotate about a specified single axis.

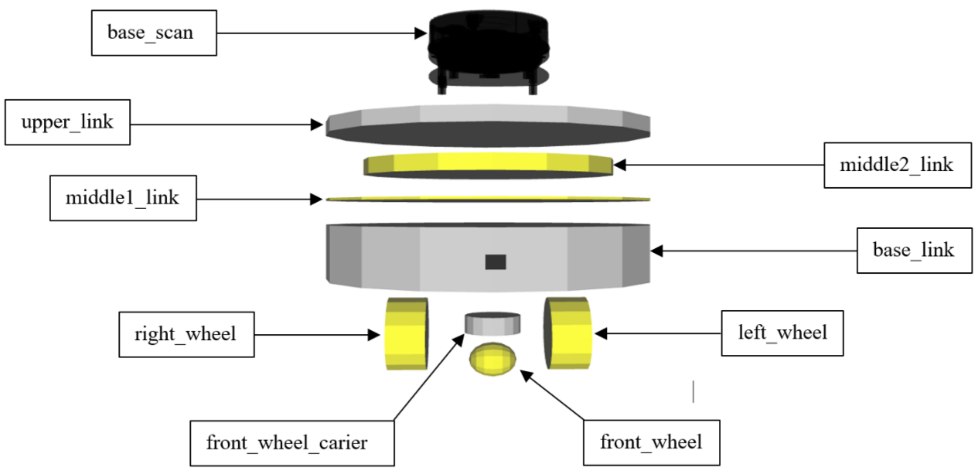


Figure 3. *The links that have been used to model our robot using URDF file in ROS*

In addition, to let robot interacts with the surrounding and to handle kin robot detection problem, we equipped the robot with a 2D laser sensor (Lidar) fixed on the top of the robot. Therefore, it is necessary too to add some links and joints to the robot, which will do the function of sensors when Gazebo plugins will be added. It is worth mentioning that these components forming a tree where the base link represents the root, with connections to each of the rear wheels and the front caster on the one hand, and connections to the middle link and camera link from another hand, and in turn each link of them is connected with other links respectively as illustrated in Figure 4.

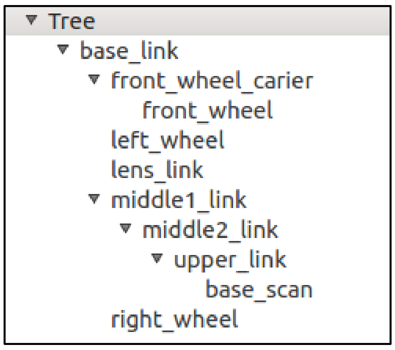


Figure 4. *The structure of the robot's components as a tree where each parent link has one or more child link*

4. Kin Detection Method

The proposed method uses a mobile robot prototype simulated using ROS and Gazebo platforms and equipped with a Lidar sensor to handle the kin detection task. The Lidar used to acquire surrounding data is the RPLIDAR A1. RPLIDAR A1 is a 360°, 2D laser scanner (Lidar) that uses laser triangulation ranging principle and uses high-speed vision acquisition and processing hardware developed by Slamtec [26]. The RPLIDAR A1 runs clockwise to perform a 360° scan within 12-meter range (see Figure 5). The produced 2D point cloud data can be used in mapping, localization, and object/environment modeling. The system measures distance data in more than 8000 times per second and with high-resolution distance output (<1% of the distance). The Lidar gives 360 distance points ($d_i, 0 \leq i \leq 359$) starting from the front of the sensor that corresponds to orientation 0° and goes clockwise to cover a 360° field-of-view. Lidar characteristics are shown in Table 2.

Lidar-based studies use a common approach which depends on using circular robots or any circular object attached to the robot (As in our case where we use a circular-shaped Lidar) and then attempting to detect these robots by taking advantage of circularity and known diameter of these robots or objects attached to them. The main idea of these studies is to find segments from Lidar measurements corresponding objects in the environment and apply one of the circle fitting methods to find robots.

As Figure 12 shows, the proposed method implements kin detection task by applying the following steps: (1) acquisition of laser data and pre-processing, (2) segmentation of data using the point-distance-based segmentation method, (3) classification of segments by applying two levels of filtering: filtering by segment diameter which aims to eliminate segments that don't fit a certain size (Lidar size) using features for each segment, filtering by segment shape to check remaining segments to test if they fit the Lidar's shape (which is a circle with known radius) or not by using the circle fitting method, and (4) identify the position of kin relative to the observer robot.

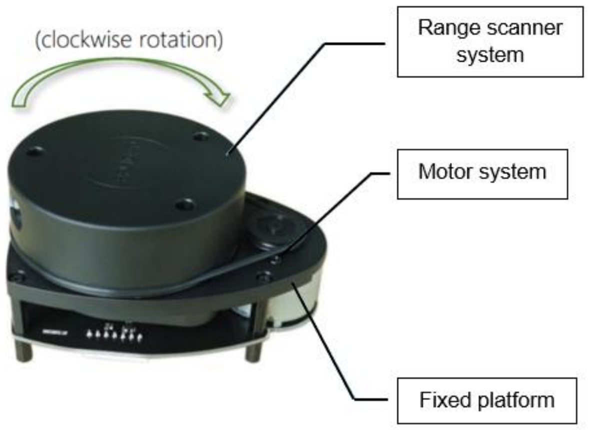


Figure 5. *RPLidar A1 system composition (source: RPLidar A1 datasheet).*

Table 2. RPLidar A1 specification

Item	**Description**	**Value**
RPLidar distance range	The minimum and maximum range of lidar beams in meter	0.15–12 m
Angular range	The angular range that Lidar can scan in degree	0–360°
Distance resolution	The smallest difference in distance that the LIDAR can measure	<1% of the distance
Angular resolution	The angular increment that LiDAR use to scan its surroundings	≤1
r	The radius of used Lidar	0.035 m

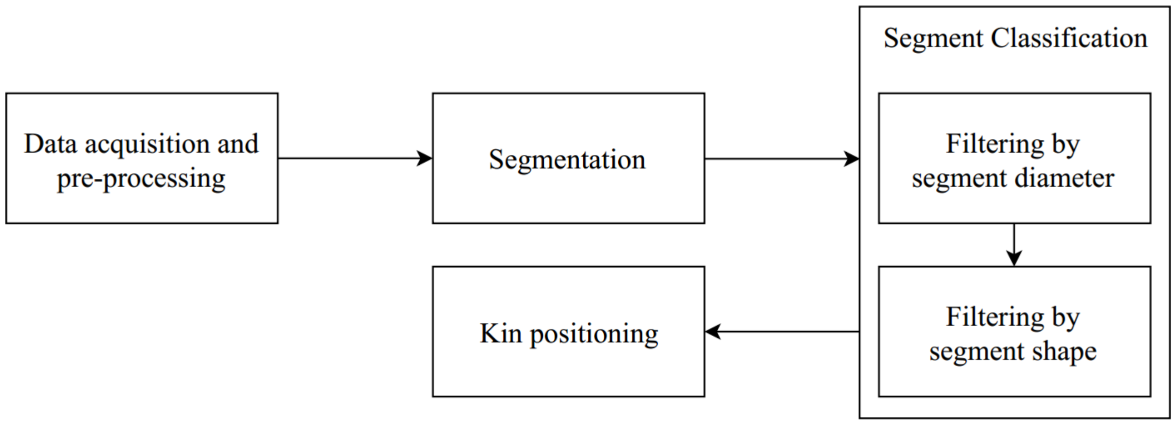

Data acquisition and pre-processing
Segmentation
Segment Classification
Filtering by segment diameter
Filtering by segment shape
Kin positioning

Figure 12. *Flowchart of the kin detection method proposed in this paper. Segment classification step is performed by filtering segments in two subsequent stages.*

4.1. Data Acquisition and Pre-Processing

As mentioned in the previous section, the Lidar being used measures 360 distance points. Each of the raw laser points is represented in the polar coordinate system as $\{(d_i, \theta_i); 0 \leq i \leq 359\}$, where d_i is the distance measured from the center of observer robot to the object and θ_i the relative angle of the measurement (see Figure 6). So as a first step, the acquired Lidar data are stored as a vector (d_i, θ_i), then the stored data is checked to convert the infinity scan values which means that there is no obstacle against the ray to the maximum range value that could be measured by the Lidar (d_{max}). In the same way, any object located at (d_{max}) from the observer robot will be ignored too. In a real-life situation, there will not be infinity values but instead, Lidar will directly give the max range value for objects outside its operating range. Moreover, it is also possible in this stage to apply some type of filtering to remove noise from the Lidar data. However, we did not apply any filtering in this studty as we did not add any noise to the simulated Lidar data.

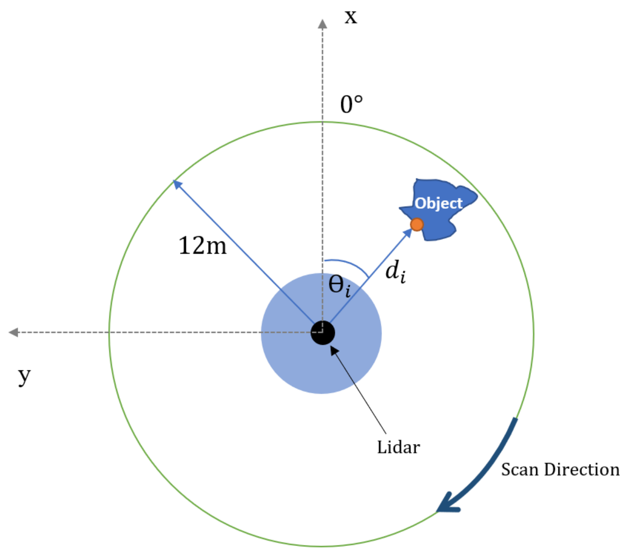


Figure 6. *RPLIDAR A1 scanning system. The maximum value of d_i (d_{max}) is 12m for RPLidar A1 Lidar.*

4.2. Segmentation

Segmentation is the process of transforming the raw laser points into primal groups of segments (useful data) which could be a robot, human or other things. So, segments can be defined as a set of range measurements (points) in the plane close to each other and probably belonging to one single object. Segmentation methods can be classified into two categories: Point-Distance-Based Segmentation Methods (PDBS) and KF-Based Segmentation Methods (KFBS) [37]. The first is related to those methods based in the Euclidean distance between points as a breakpoint criteria condition, and the last is related to Kalman Filter approaches.

Our method uses the PDBS method. So, after reading the raw laser data and storing values as a vector in the previous step, we attempt to detect the different possible objects in the environment by using derivative of data (∂_i) which can be calculated by finding the difference between each distance point (d_i) and the one before it (d_{i-1}), then compare the result with a specific threshold value (d_{th}) which allows to ignore the small changes in scan data as following:

$$\partial_i = \{d_i - d_{i-1} \; if \; |d_i - d_{i-1}| > d_{th}, 0 \; otherwise\}; 1 \leq i \leq 359 \qquad (1)$$

As a result of that, for each individual object in the scene we will have a falling slope $(\partial_i < -d_{th} < 0)$ in derivative that represents moving from a big to smaller value in data, and a rising slope $(0 < d_{th} < \partial_i)$ which represents moving from a small to bigger value in data (as shown in Figure 7). So, by combining each falling and rising edge from derivatives we could have a probable segment which represent an object in the environment. Figure 7 shows representation of Lidar scan data for two different objects (cuboid and cylinder) and the differences between each scan point and the previous one according to the ray number.

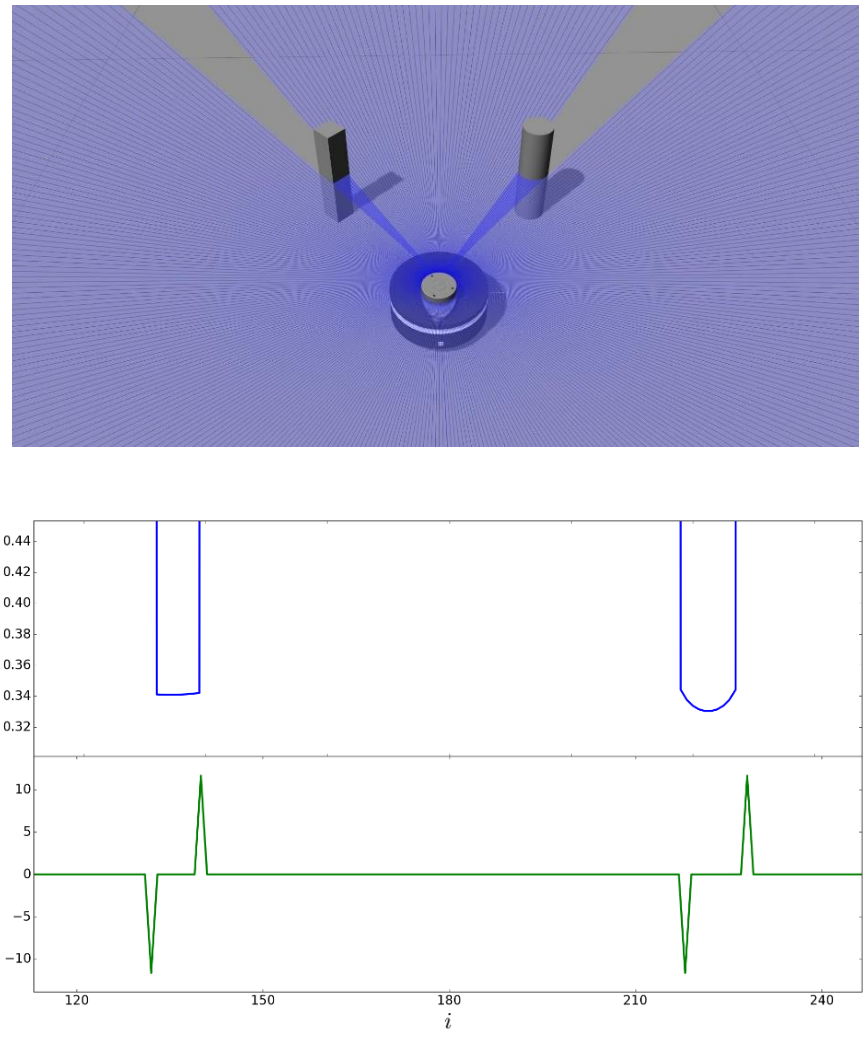


Figure 7. *Top: simulation of a robot with two different objects (cuboid at the left and cylinder at the right) in Gazebo simulator, Bottom: representation of Lidar scan data (blue part) and the differences between each scan point and the previous one according to the ray number (green part)*

4.3. Segment Classification

Classification is the act or process of dividing things into groups according to their type which is in our case kin robot or not. So, after applying segmentation and getting segments for different obstacles in the previous stage, it is the time to classify these objects (segments) to distinguish the kin robots from the other objects. In this study segment classification process is implemented using geometric properties of the lidar in two stages:

- ***Filtering by segment diameter:*** segments which do not fit the Lidar size are excluded at this stage.
- ***Filtering by segment shape:*** segments which do not fit the Lidar shape (circle) are excluded at this level.

4.3.1 Filtering by segment diameter

Since we are looking for kin robots with a known Lidar size, we assume that all segments are circular and exclude the segments that do not fit the Lidar size (0.07m) from the set of segments. First, we estimate the distance to the center of this object ($d_{c_{est}}$) by looking for the distance point that has the minimum value from the set of segment's distance points, then we will add the radius of used Lidar (r) as equation 2 shows:

$$d_{c_{est}} = \min(d_i) + \mathrm{r} \tag{2}$$

Next, we calculate the minimum estimation for the object'diameter (ω_{min}) that is specified by the first and last distance points of the segment (in other words, first and last laser beams hitting the object) and the maximum estimation for the object'diameter (ω_{max}) that is specified by the laser beam before the first laser beam hitting the object and the laser beam after the last laser beam hitting the object. Finally, we check whether this object (segment) meets the equation 3 or not. Note that 0.07 in this equation corresponds to the diameter of the lidar. These calculations are illustrated in Figure 8.

$$\omega_{min} \leq 0.07 < \omega_{max} \tag{3}$$

The calculation of $\omega_{min}, \omega_{max}$ depends on the distance between the observer robot and the center of the object that has been estimated in equation 2. It also depends on the corresponding angles among the rays. However, note that the number of rays hitting the object from both sides may be different. Therefore, the minimum estimation of the object's diameter (ω_{min}) will be the result of combining the left (ω_{min_l}) and the right (ω_{min_r}) parts of the ω_{min} as illustrated at the left of Figure 8:

$$\begin{gathered} \omega_{min_l} = d_{c_{est}} \times tan(\alpha_{min_l}) \\ \omega_{min_r} = d_{c_{est}} \times tan(\alpha_{min_r}) \\ \omega_{min} = \omega_{min_l} + \omega_{min_r} \end{gathered} \tag{4}$$

In a similar way the maximum estimation for the object'diameter (ω_{max}) will be the result of combining the left (ω_{max_l}) and the right (ω_{max_r}) parts of the ω_{max} as illustrated at the right of Figure 8:

$$\begin{gathered} \omega_{max_l} = d_{c_{est}} \times tan(\alpha_{max_l}) \\ \omega_{max_r} = d_{c_{est}} \times tan(\alpha_{max_r}) \\ \omega_{max} = \omega_{max_l} + \omega_{max_r} \end{gathered} \tag{5}$$

As shown in Figure 8, the main idea depends on finding the minimum (ω_{min}) and the maximum (ω_{max}) possible diameters of the object (segment). If the lidar's known diameter (7 cm) resides between the estimated minimum and maximum diameters, this object is not eliminated and passes to the next filter described in the following section.

4.3.2 Filtering by segment shape

In this stage, segments that have passed the previous filtering step is checked in terms of shape. Filtering by segment shape aims to eliminate the non-circular segments which represent in turn non-kin objects. Therefore, we will test whether segments fit a circle or not by using one of the circle fitting methods. There are many methods for circle fitting [38], but we used Least Squares Method with a modification of the Levenberg-Marquardt Algorithm [30] in the polar coordinate system (Equation 6). In equation 6, (r_i, θ_i) is the polar coordinates of a subset of data scan points that represent a segment, r is the radius of lidar (3.5 cm) and (r_c, θ_c) is the polar coordinates of the estimated circle center which can be calculated as in Equation 7:

$$F = \sum_{i=1}^{n} \sqrt{r_i^2 + r_c^2 - 2r_i r_c \cos(\theta_i - \theta_c)} - r \tag{6}$$

$$r_c = \min(d_i) + r \tag{7}$$

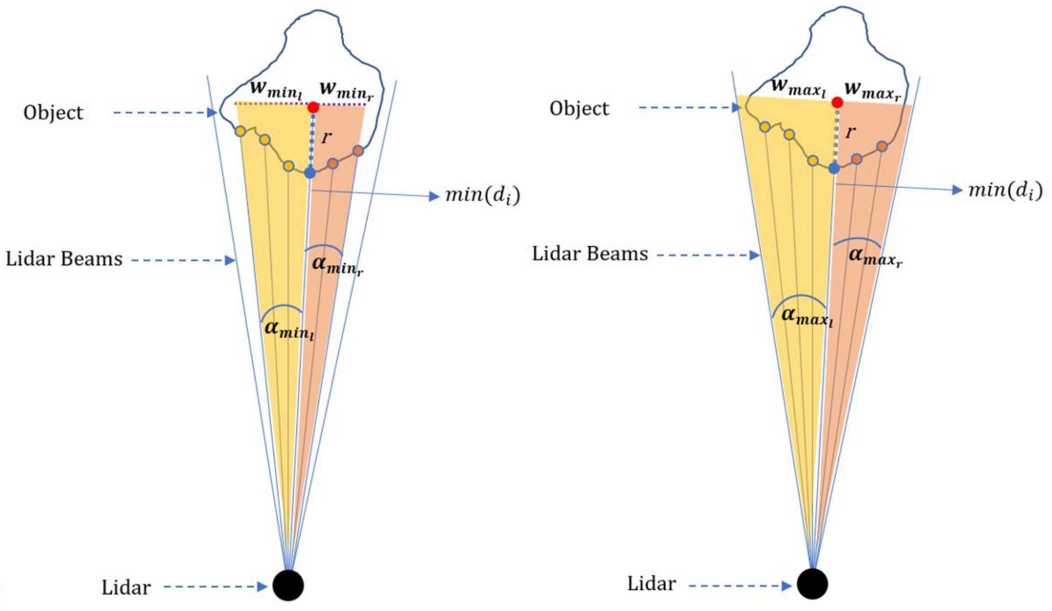


Figure 8. *Calculating the minimum (at the left) and maximum (at the right) diameter of the object for a certain segment. Note that number of laser beams hitting this object is different for left and right areas which are specified by the laser beam with minimum distance (denoted with min(d_i))*

So, by utilizing the knowledge of Lidar radius r of the target teammate robot and by finding the distance between the estimated center and the selected points (As shown in Figure 9 and Equation 7) we can decide if a segment is circular or not according to a certain threshold (which is 0.006 empirically) where F function will be close to zero in the case of detecting a robot (circle) and will have relatively big value for other objects due to estimation error of the object's center.

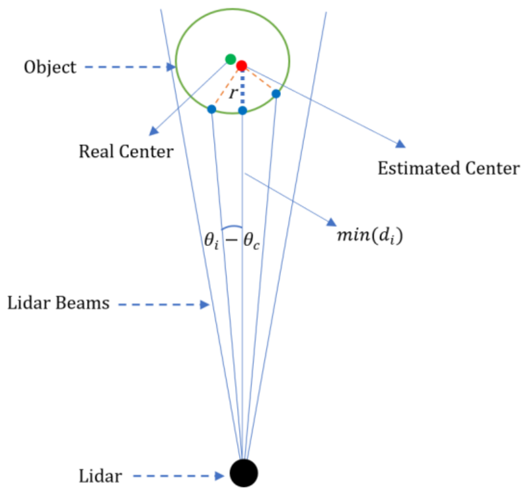


Figure 9. *Illustration for calculating the F (error) function for the estimated center and the relevant data points (blue points). Red dot represents the estimated center and the green dot represents the real center for the circular object (e.g. lidar).*

4.4. Kin Positioning

After segment filtering, it is important to use remaining segments to identify relative position of kins. Kin positioning process involves estimating the distance and relative angle between the observer robot and other kins. As we have used a geometrical method to detect valid segments for kins, we already obtained relative positions of kins in the segment filtering step.

5. Experiments

We considered two different scenarios to evaluate our study and verify the validity of the results. In the first scenario, we placed one kin robot and some cylindrical objects around an observer robot. The distance from the observer robot to the kin robot and cylindrical objects were the same, but the diameters of the cylindrical objects were different (Figure 10 - top). In this experiment, we wanted to test the ability of our kin detection method to eliminate objects (segments) which don't fit the size of lidar. While the first scenario was intended to test the first filter of the the segment classification stage, the second scenario was designed to test the second filter of the segment classification stage. In the second scenario we test the ability of the algorithm to distinguish kin robots from other objects that have the same dimensions as the Lidar by using the circle fitting algorithm. The objects used in the second experiments had different shapes (Figure 11 - top).

5.1. Scenario I: Test for filtering by segment diameter

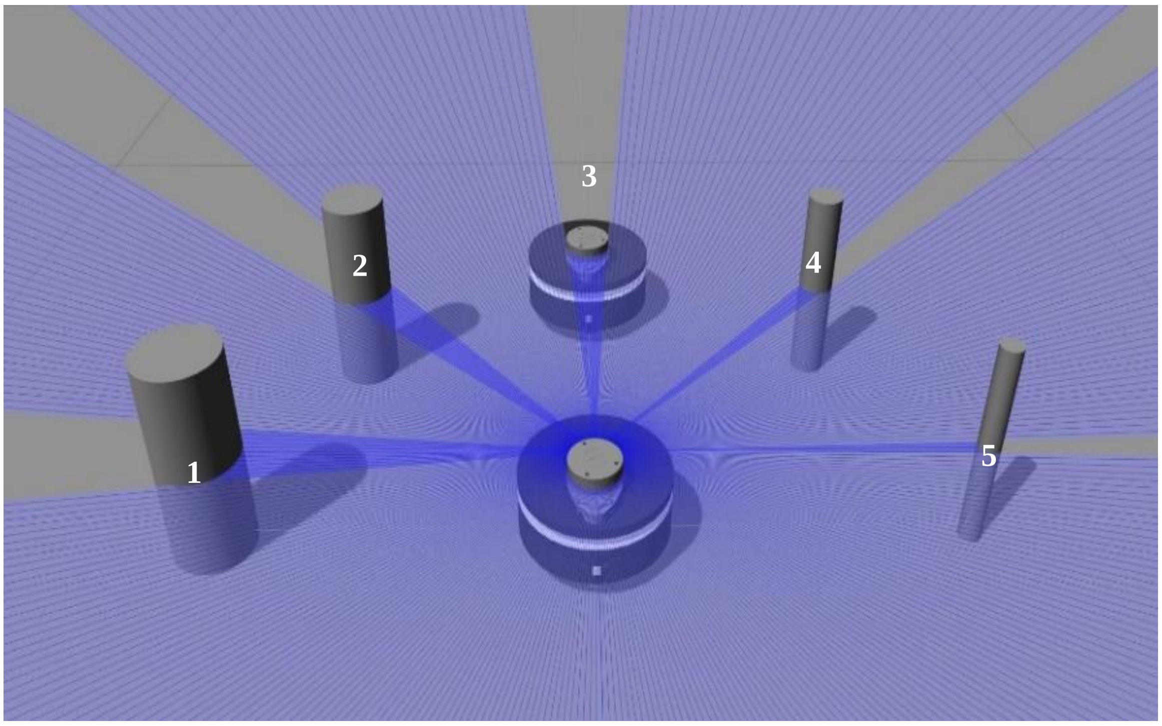
In this scenario, we want to test the ability of our kin detection method to eliminate objects (segments) which don't fit the size of Lidar using filtering by segment diameter stage. We have used one kin robot with four cylindrical objects. These objects have a different radius from our lidar (3, 5, 9 and 11cm) and all of them are located at the same distances from the observer robot. The distances being tested are 50 (as shown in Figure 10), 80 and 120 cm. As a result of this experiment, the object which has a radius differs from the radius of the kin's lidar is not be identified as a kin candidate and therefore it is filtered directly in filtering by segment diameter stage.

As illustrated in Figure 10 (top), there are four cylindrical objects differs in diameter size around the observer robot. There is also one kin robot located in the middle of the scene. The difference in the size of the objects affects the number of rays hitting these objects, as well as the values of the lidar readings. As a result of that, we can notice from the chart (Figure 10 - bottom), that holes which formed as a result of hitting laser beams with these obstacles are dissimilar in size. It can be also seen that the observer robot succeeded to identify the teammate robot located in the middle (green point) from other obstacles (red stars).

5.2. Scenario II: Test for filtering by segment shape

This scenario aims to test the second filtering level of the segment classification stage. In this experiment, we test the ability of the kin detection method to eliminate objects (segments) which don't meet the shape of Lidar. We have one cylindrical object (at the left side of the kin robot) and five non-cylindrical obstacles (cuboid, triangular, pentagon, and two hexagonal prisms). These objects have been placed within the same distances of the previous scenario but this time the same Lidar radius has been used for them (as shown in Figure 11 – top).

As can be noticed from Figure 11 (top), we have one robot teammate in the middle of the scene and some other obstacles around the observer robot. All objects have the same size of Lidar, so the segment size will not be useful to distinguish kins from the other objects. Therefore, in this scenario, we depended on the shape of the segments (holes) instead of their sizes. By using circle fitting method, we have succeeded to distinguish between the kin robot and the other objects as illustrated in Figure 11 (bottom). However, we also have a wrong detection for the cylindrical obstacle, which has the same size and shape of lidar.



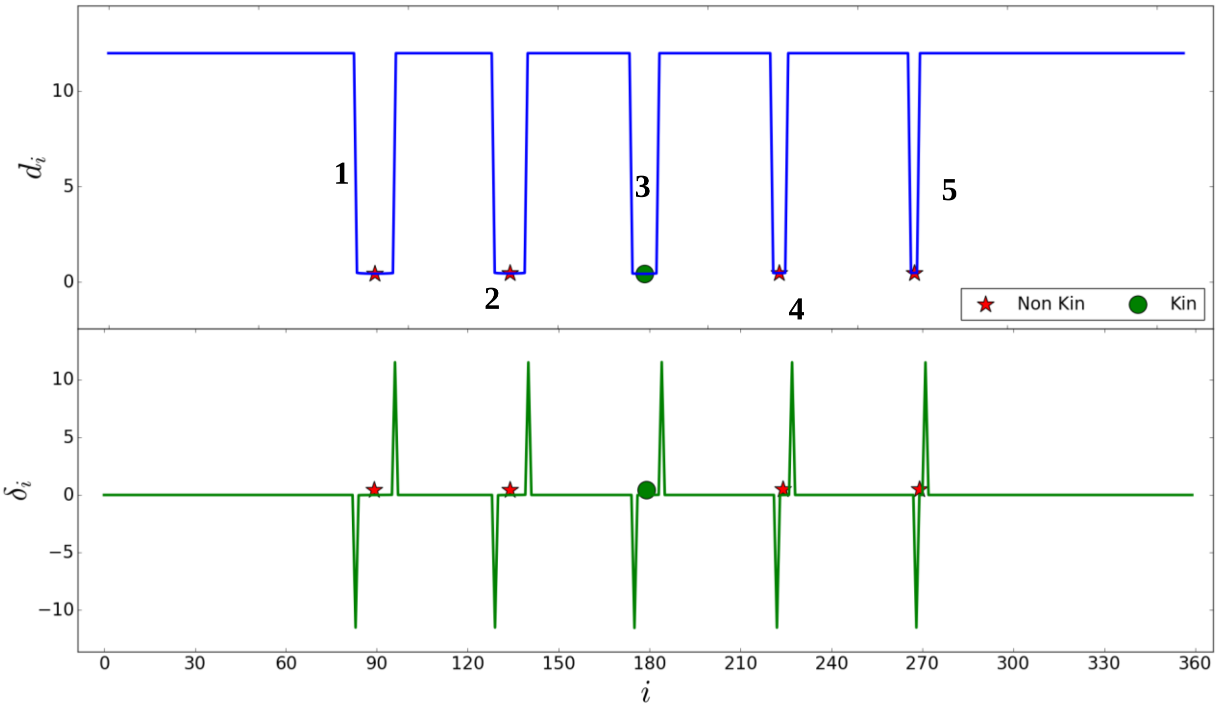
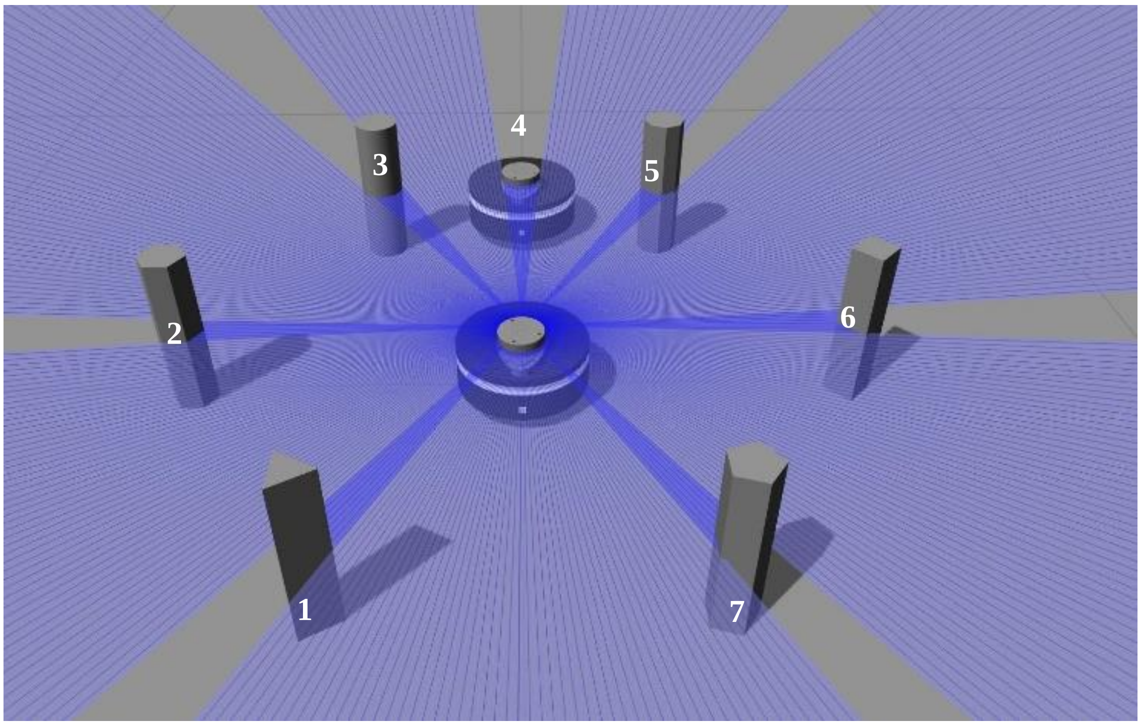


Figure 10. *Top: simulation of a kin robot with four other cylindrical objects having the following size (from right to left): 3, 5, 9, 11 cm and located on 50cm from the observer robot in Gazebo environment; Bottom: representation of scan data and derivative with successful detection of the kin robot (green circle)*



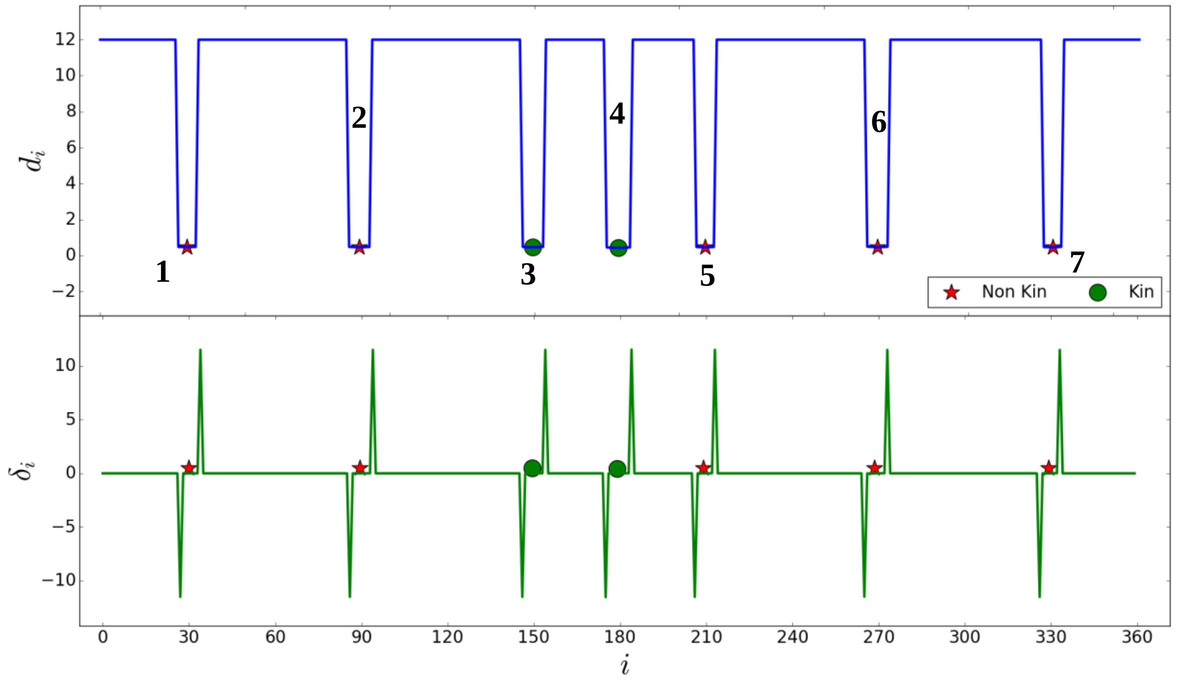


Figure 11. *Top: simulation of a kin robot with one cylindrical and five non-cylindrical objects having the same Lidar's size (7 cm) and located on 50cm from the observer robot in Gazebo environment; Bottom: representation of scan data and derivative with successful detection of the kin robot (green circle, No: 4) and wrong detection of cylindrical object with radius similar to Lidar's radius (the second green circle at the left of previous one, No: 3)*

6. Conclusion and Future Work

Robot Operating System integrated with high fidelity simulator like Gazebo is a powerful tool in the robotics field. In this study, we have developed a simulation model of a new mobile robot for research and educational purposes, where the robot provided with a lidar sensor. The lidar on the robot is used for kin detection which is considered as one of the most important issues in the field of multi-robot and swarm robotic systems, where dozens of robots have to collaborate with each other to implement some critical tasks.

While this connection between robots can be provided by Wi-Fi networks and GPS systems, the unreliability of these networks and the amount of communicated data that grows gradually with the number of robots in the team keeps the need for an alternative solution based on local sensing. Several studies have dealt with this subject using a variety of sensors and methods [20, 21, 22], but there are few studies performing kin detection using lidar [23, 24, 25]. For this reason, as a case study for our new robot, we proposed a new geometric kin detection method and tested it with two different scenarios.

Lidar-based kin detection method proposed in this study depends on the following steps: (1) acquisition of laser data and pre-processing, (2) segmentation of data using the point-distance-based segmentation method, (3) classification of segments by applying two levels of filtering: filtering by segment diameter which aims to eliminate segments that don't fit a certain size (lidar size) using features for each segment, and filtering by segment shape to check remaining segments to test if they fit the lidar's shape (which is a circle with known radius) or not by using the circle fitting method, and (4) identify the position of kin relative to the observer robot. As results show, we were able to detect different objects in the environment and to distinguish the robot team members from other objects. But it is important to mention that the success of detection and obtaining accurate results depends largely on the distance of the member robots and other objects from the observer robot. As the distance between the observer robot and other objects increases, the problem becomes more difficult because the number of Lidar rays that collide with these obstacles becomes less and the area between colliding laser rays becomes wider when the distance between observer robot and the object/robot increases. This effects performance of segment classification phase.

In future works, further research and more systematic experiments will be performed to improve the kin detection algorithm. One example of such works is reimplementing the segment classification stage using machine learning algorithms as in [25].

References

[1] Şahin, E. (2004, July). Swarm robotics: From sources of inspiration to domains of application. In International workshop on swarm robotics (pp. 10-20). Springer, Berlin, Heidelberg.

[2] Arvin, F., Samsudin, K., & Ramli, A. R. (2009). Development of a miniature robot for swarm robotic application. International Journal of Computer and Electrical Engineering, 1(4), 436-442.

[3] McLurkin, J., McMullen, A., Robbins, N., Habibi, G., Becker, A., Chou, A., ... & Kim, S. (2014, September). A robot system design for low-cost multi-robot manipulation. In 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 912-918). IEEE.

[4] Gupta, M., & Singh, K. (2010, November). AutoBot: a low-cost platform for swarm research applications. In 2010 3rd International Conference on Emerging Trends in Engineering and Technology (pp. 33-36). IEEE.

[5] Arvin, F., Murray, J., Zhang, C., & Yue, S. (2014). Colias: An autonomous micro robot for swarm robotic applications. International Journal of Advanced Robotic Systems, 11(7), 113.

[6] Kernbach, S., Thenius, R., Kernbach, O., & Schmickl, T. (2009). Re-embodiment of honeybee aggregation behavior in an artificial micro-robotic system. Adaptive Behavior, 17(3), 237-259.

[7] Turgut, A. E., Çelikkanat, H., Gökçe, F., & Şahin, E. (2008). Self-organized flocking in mobile robot swarms. Swarm Intelligence, 2(2-4), 97-120.

[8] Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., ... & Martinoli, A. (2009). The e-puck, a robot designed for education in engineering. In Proceedings of the 9th conference on autonomous robot systems and competitions (Vol. 1, No. CONF, pp. 59-65). IPCB: Instituto Politécnico de Castelo Branco.

[9] Hilder, J., Naylor, R., Rizihs, A., Franks, D., & Timmis, J. (2014, September). The pi swarm: A low-cost platform for swarm robotics research and education. In Conference Towards Autonomous Robotic Systems (pp. 151-162). Springer, Cham.

[10] Castillo-Pizarro, P., Arredondo, T. V., & Torres-Torriti, M. (2010, October). Introductory survey to open-source mobile robot simulation software. In 2010 Latin American Robotics Symposium and Intelligent Robotics Meeting (pp. 150-155). IEEE.

[11] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., ... & Ng, A. Y. (2009, May). ROS: an open-source Robot Operating System. In ICRA workshop on open source software (Vol. 3, No. 3.2, p. 5).

[12] W. Garage, ROS: Robot Operating System, 2011[J]. URL: http://www. ros.org, 2011

[13] Koenig, N., & Howard, A. (2004, September). Design and use paradigms for gazebo, an open-source multi-robot simulator. In 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566) (Vol. 3, pp. 2149-2154). IEEE.

[14] Quigley, M., Gerkey, B., & Smart, W. D. (2015). Programming Robots with ROS: a practical introduction to the Robot Operating System. O'Reilly Media, Inc..

[15] Hershberger, D., Gossow, D., & Faust, J. RViz, 3D visualization tool for ROS. URL: http://wiki. ros. org/rviz [cited 06-08-2016].

[16] Kunze, L., Roehm, T., & Beetz, M. (2011, May). Towards semantic robot description languages. In 2011 IEEE International Conference on Robotics and Automation (pp. 5589-5595). IEEE.

[17] Lee, K. H., Woo, H., & Suk, T. (2001). Data reduction methods for reverse engineering. The International Journal of Advanced Manufacturing Technology, 17(10), 735-743.

[18] Miyahara, K., & Okada, Y. (2009, March). COLLADA-based File Format Supporting Various Attributes of Realistic Objects for VR Applications. In 2009 International Conference on Complex, Intelligent and Software Intensive Systems (pp. 971-976). IEEE.

[19] http://gazebosim.org/tutorials?tut=ros_gzplugins

[20] Pugh, J., & Martinoli, A. Local Range and Bearing Sensing Using Infrared Transceivers in Mobile Robotics.

[21] Rivard, F., Bisson, J., Michaud, F., & Létourneau, D. (2008, May). Ultrasonic relative positioning for multi-robot systems. In 2008 IEEE International Conference on Robotics and Automation (pp. 323-328). IEEE.

[22] Bolla, K., Kovacs, T., & Fazekas, G. (2010, May). Compact image processing-based kin recognition, distance measurement and identification method in a robot swarm. In 2010 International Joint Conference on Computational Cybernetics and Technical Informatics (pp. 419-424). IEEE.

[23] Wąsik, A., Ventura, R., Pereira, J. N., Lima, P. U., & Martinoli, A. (2016). Lidar-based relative position estimation and tracking for multi-robot systems. In Robot 2015: Second Iberian Robotics Conference (pp. 3-16). Springer, Cham.

[24] Teixidó, M., Pallejà, T., Font, D., Tresanchez, M., Moreno, J., & Palacín, J. (2012). Two-dimensional radial laser scanning for circular marker detection and external mobile robot tracking. Sensors, 12(12), 16482-16497.

[25] Zhou, X., Wang, Y., Zhu, Q., & Miao, Z. (2016, November). Circular object detection in polar coordinates for 2D LIDAR data. In Chinese Conference on Pattern Recognition (pp. 65-78). Springer, Singapore.

[26] Laser range scanner RPLIDAR A1 Datasheet is available online at: https://download.slamtec.com/api/download/rplidar-a1m8-datasheet/2.1?lang=en

[27] Trianni, V., Groß, R., Labella, T. H., Şahin, E., & Dorigo, M. (2003, September). Evolving aggregation behaviors in a swarm of robots. In European Conference on Artificial Life (pp. 865-874). Springer, Berlin, Heidelberg.

[28] Sahin, E., Labella, T. H., Trianni, V., Deneubourg, J. L., Rasse, P., Floreano, D., ... & Dorigo, M. (2002, October). SWARM-BOT: Pattern formation in a swarm of self-assembling mobile robots. In IEEE International Conference on Systems, Man and Cybernetics (Vol. 4, pp. 6-pp). IEEE.

[29] Turgut, A. E., Çelikkanat, H., Gökçe, F., & Şahin, E. (2008). Self-organized flocking in mobile robot swarms. Swarm Intelligence, 2(2-4), 97-120.

[30] Moré, J. J. (1978). The Levenberg-Marquardt algorithm: implementation and theory. In Numerical analysis (pp. 105-116). Springer, Berlin, Heidelberg.

[31] Gazebo: The Player Project, Free Software Tools for Robot and Sensor Applications. http://playerstage.sourceforge.net/

[32] Rohmer, E., Singh, S. P., & Freese, M. (2013, November). V-REP: A versatile and scalable robot simulation framework. In 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 1321-1326). IEEE.

[33] Michel, O. (2004). Cyberbotics Ltd. Webots™: professional mobile robot simulation. International Journal of Advanced Robotic Systems, 1(1), 5.

[34] AnyKode Marilou - Modeling and simulation environment for Robotic. Anykode.com. Retrieved October 3, 2019, from http://www.anykode.com

[35] 4D-virtualiz Simulator. 4d-virtualiz.com. Retrieved October 3, 2019, from https://www.4d-virtualiz.com

[36] Gazebo and ROS Integration. Retrieved October 3, 2019, from http://gazebosim.org/tutorials/?tut=ros_overview

[37] Premebida, C., & Nunes, U. (2005). Segmentation and geometric primitives extraction from 2d laser range data for mobile robot applications. Robotica, 2005, 17-25.

[38] Rusu, C., Tico, M., Kuosmanen, P., & Delp, E. J. (2003). Classical geometrical approach to circle fitting--review and new developments. Journal of Electronic Imaging, 12(1), 179-194.