

Lidar-based Robot Detection and Positioning using Machine Learning Methods

Zahir Yilmaz and Levent Bayindir


Abstract— This paper presents a machine learning-based kin detection method for multi-robotic and swarm systems. Detecting surrounding objects and distinguishing robots from these objects (kin detection) are essential in most multi-robotic applications. While infrared, ultrasonic, and vision systems were mainly used for applying the robot detection and relative positioning task in the literature, studies using the Lidar-based approach are limited. The proposed method uses the Lidar sensor to discover the work area and determine the distance and the angle of all kin members relative to the observer robot. The main steps of the proposed method can be summarized as follows: 1) the Lidar distance points are read and stored as a vector with some preprocessing, 2) the acquired distance points representing different objects in the environment are separated from each other using a segmentation method, 3) to classify the segmented objects, the segment classification process starts with extracting five features for each object, then these features are fed to various machine learning classification algorithms to distinguish the kin robots, 4) the segments classified as a kin robot in the previous step are handled, and the relative position is found for each of them. A new mobile robot prototype has been modeled and equipped with a Lidar sensor using ROS platform. Lidar has been used to collect data, and four different classification methods have been tested to verify the efficiency of the method using Gazebo simulation platform.

Index Terms— kin detection, robot detection, relative positioning, machine learning, ROS, Gazebo


I. INTRODUCTION

IN THE PAST, the interest of scientists has been focused on finding ways to help humans in their daily life and facilitate some complex tasks they do. However, this attention has changed in the last decades to develop machines that can replace humans and replicate their actions. On the other side, while human takes days or even hours to perform a task with a possible error rate, robot, in turn, performs that task with much less time, more efficiently, and with a zero-error rate.

ZAHİR YILMAZ, is with Department of Computer Engineering, Ataturk University, Erzurum, Turkey, (e-mail: yilmaz92.tr@gmail.com).

 <https://orcid.org/0000-0002-5009-6763>

LEVENT BAYINDIR, is with Department of Computer Engineering, Ataturk University, Erzurum, Turkey, (e-mail: levent.bayindir@atauni.edu.tr).

 <https://orcid.org/0000-0001-7318-5884>

Manuscript received August 21, 2021; accepted April 28, 2022.
DOI: [10.17694/bajece.984744](https://doi.org/10.17694/bajece.984744)

Therefore, it is quite natural to witness a big revolution in robotics that is considered a branch of science that aims to design machines that can perform tasks on human commands or by themselves to make human work easier or more productive. Robotics is one of the most scientific fields that has seen significant progress in the last few decades. This progress ranges from personal assistant robots to multi-robotic and intelligent swarm robotics systems.

Robots are integrated machines usually made up of sensors, actuators, control systems, and some dedicated software and worked either depending on human commands or autonomously to perform various tasks. According to the degree of mobility of robots, there are two main types of robots: the fixed robots, which cannot move in their environment, and the mobile robots, which can travel in the environment by using various means of locomotion. However, mobile robots are more desirable nowadays due to their navigation and sensing capabilities useful for many different tasks [1, 2]. Mobile robots can be seen today in all aspects of life; autonomous vehicles, robot vacuums, assistant robot devices, search and rescue robots are some examples. Moreover, instead of working individually, mobile robots can now work in cooperation mode to accomplish critical tasks beyond a single robot's capabilities, as in the case of multi-robot and swarm robotic systems.

Swarm robotics is an innovative approach that allows large numbers of robots to collaborate and coordinate with each other to perform critical tasks that are beyond the capabilities of an individual robot. The robots used in the swarm are relatively simple and have limited abilities compared to standalone robot systems. The main idea of swarm robotics is inspired by studying the collective behavior of social animals that can cooperate and coordinate among themselves for solving everyday problems such as foraging and flocking [3]. The term "swarm" or "swarm intelligence" was first launched by Beni [4], who was interested in cellular automata systems at that time. Beni and others [5] used the term "cellular robots" to refer to a group of robots with unique characteristics that closely match the insect's swarms. After that, Beni chose the word "swarms" as a better term for cellular robots. Later, the term "swarm" evolved, and many new concepts and terms emerged from it, such as swarming, swarm optimization, "swarm engineering, and swarm robotics [6].

In the last few decades, and due to the tremendous technological progress, considerable development was achieved in swarm intelligence and multi-robot cooperation.

Swarm robotics studies have been developed primarily to address the tasks that require working according to cooperative behavior. Şahin [3] chose to classify tasks according to the nature of the task as follows:

- Tasks that cover a region such as surveillance and environmental monitoring
- Tasks that are critical and dangerous such as rescue and mine detection
- Tasks that scale up or scale down in time
- Tasks that require redundancy, such as establishing a dynamic communication network on the battlefield

Brambilla et al. [7] categorized tasks according to a slightly different perspective. They classify tasks according to behaviors where they proposed four categories of collective behaviors:

- Spatially-organizing behaviors including aggregation, pattern formation, chain formation, self-assembly and morphogenesis, and object clustering and assembling
- Navigation behaviors such as collective exploration, coordinated motion, and collective transport
- Collective decision-making such as agreement (consensus achievement) and specialization (task allocation)
- Other collective behaviors may contain some other works such as collective fault detection, group size regulation, and human–swarm interaction

However, regardless of the type of task required, in most cases, the robots in the swarm have to navigate in an environment populated by various objects. Provided that the robots are equipped with some sensors, it is crucial to distinguish the team members from the other objects and estimate their relative positions based on the sensors' data.

Detecting surrounding objects in the environment and distinguishing swarm members from these objects (often called "kin detection task") is essential for collective behaviors discussed above. It is also worthy to note that this process does not confine to detecting the robots, but it also means localization since the approximate coordinates of the robots can be recovered from the sensors data. Several studies have proposed solutions to tackle the kin detection and relative positioning task. Different sensors have been used to implement this task, such as infrared (IR), ultrasound, vision, Lidar sensors. The infrared-based approach is the most common approach for solving the robot detection and relative positioning task. A major advantage of using Lidars for robot detection and positioning is outdoor efficiency. Unlike IR-based systems, Lidars can be designed to work in outdoor environments more efficiently.

One of the IR-based robot detection and relative positioning systems was constructed by Kelly and Martinoli [15]. The proposed IR system can reveal the distance and orientation of each robot in the multi-robot system. Each robot in the team

was equipped with twelve IR Light Emitting Diodes (LEDs) and four photodiodes controlled by a PIC microcontroller. LEDs have been distributed around the robot's perimeter in a manner that ensures as much coverage as possible of the surroundings. The photodiode receivers have also been fixed at 90 degrees from each other to be able to sense signals in all directions. The distance from each robot to another and the bearing can be determined by comparing all four receivers' received signal strengths. The flocking task has been implemented using eight robots to test the method. Since the IR-based approach is the most method that has been used in literature, there are a lot of robot platforms that use this approach to tackle kin detection, relative positioning, and other related tasks [16-23].

As an example of using non-IR-based systems, Bolla et al. [8] developed a visual kin recognition and localization method that can detect and identify kin robots in a swarm robotic system. The main idea of the suggested method depends on using the Fast Fourier Transform (FFT) to extract the peak in the FFT spectrum related to the zebra pattern used to distinguish robots in the swarm. Moreover, the proposed method is not only able to detect the kin robots but can also estimate their distances and to make identification among them too. Other studies that have used the vision-based approach to achieve the detection and positioning problem for multi-robot systems are [9-11].

The relative positioning system, which uses ultrasonic sensors, developed by Rivard et al. [12], is another example of a non-IR-based system. The proposed system consists of one ultrasonic transmitter, three receivers, and one RF communication link. The detection method depends on evaluating the time-of-flight of ultrasonic pulses from the transmitter to the receiver that has revealed the signal first. The role of the RF link is to determine the time-of-flight of ultrasound pulses as it spreads much faster than ultrasound pulses. Other studies that use ultrasonic sensors are [13] and [14].

Although Lidars have been heavily used in robotics [24-26] and autonomous vehicles [27-29], Lidar-based kin detection and relative positioning methods are very limited in the literature, and most of these studies do not solve the entire problem. Premebida et al. [30] have presented some algorithms to perform segmentation and feature extraction based on data acquired from the Lidar sensor. For data segmentation, which aims to group segments into sets to distinguish the different objects in the environment, two different methods have been suggested: Point-Distance-Based Segmentation Methods (PDBS) and Kalman Filter-Based Segmentation Methods (KFBS). While in the feature extraction process, three different geometric primitives (lines, circles, and ellipses) have been used to formulate the extraction and fitting problem since these geometric shapes can be described as cases of conics. Teixidó et al. [31] have presented a proposal for circular marker detection and external mobile robot tracking. The proposed work uses an external fixed 2D Lidar to detect cylindrical objects attached to a mobile robot. Circle fitting based on the least-squares method with an algorithm for outlier avoidance

has been used in the cylindrical target detection. Several experiments have been accomplished to evaluate the positioning error obtained in the center estimation process of the cylindrical targets, where different distances, orientations, and target diameters have been used. In [32], two laser range finder sensors have been used to obtain the full field of view for a mobile robot. The proposed method lets a robot estimate a distance (range) and an angle (bearing) to another robot in the environment using measurements extracted from the raw data provided by the Lidar. The method uses circle fitting to find circular objects and filter out outliers in the set of detected circular objects. Kalman filter has also been used to improve the estimate of the relative robot position. Four distinct scenarios have been tested using MBot robots to determine the detection and tracking method's performance. In [33], Zhou et al. have chosen to use machine learning methods to implement the object detection task (similar to the kin detection task). Like others, they have used the Lidar to distinguish circular and straight objects from other targets. The proposed method depends on three main stages. The first stage is the segmentation process, where authors have proposed an improvement to the segmentation approach in [34] and called it the Improved Dietmayer method (IDIET). In the second stage, five different features have been defined for each. Finally, the Support Vector Machine (SVM) machine learning method has been used to detect the target circular objects. Another Lidar-based kin detection study was performed by us recently [35]. This study developed a geometric approach for kin detection and used the same simulation model and the Lidar for the proposed method.

This paper is organized as follows. Section 2 presents an overview of ROS, Gazebo, and URDF files used to simulate and model our robot. Section 3 introduces the proposed robot detection and positioning. Finally, we end this paper with the experiments and the experimental results.

II. ROBOT MODELING AND SIMULATION

In this study, a mobile robot model has been designed and simulated in the Gazebo simulation environment. The simulated robot has been equipped with a Lidar to detect different objects in the environment. The sensor data has been obtained using the ROS topics communication method. Then this raw data has been processed in ROS and utilized in the kin detection and relative positioning system. ROS, Gazebo, and the robot model are briefly explained in the following sections subsequently.

A. ROS

The ROS stands for Robot Operating System, 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 offers various tools and libraries which give the ability to build, write, debug, and run code across different workstations [36]. 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 can perform processes such as scheduling, data transmission/reception, loading, monitoring, and error handling by utilizing virtualization layer between applications and distributed computing resources. ROS has been built in small software modules called "nodes." Each node 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. Nodes exchange data using "messages" that could be either primitive or composed. ROS provides three mechanisms for communication between these nodes provided by passing messages: topics, services, and actions. Topics is an asynchronous unidirectional message transmission/reception method used to exchange data continuously. Services is a bidirectional synchronous communication method that depends on request/reply messages and is often used to command a robot to perform a specific action. Action is very similar to the service method, but it contains feedback messages that periodically report task states to the client.

B. Gazebo

Gazebo is one of the most popular simulators in the field of robotics. It has an integrated development environment that provides robots, sensors, environment models for 3D simulation and offers realistic simulation with its physics engine [37]. It has been selected as the official simulator of the DARPA Robotics Challenge in the US due to its high performance and having various plugins for different robot and sensor types. Using these plugins, we can read sensors data from Gazebo or send commands to motors by using APIs. Gazebo is developed and distributed by Open Robotics, which is in charge of ROS and its community, so it is compatible with ROS.

C. Robot description

In the world of robot simulation, it is so common to have the ability to model your robot, handle various sensor data received from this robot, control the robot by actuators, and test or evaluate algorithms. In ROS, robot models could be described in an XML format called Unified Robot Description Format (URDF). 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 format is designed to represent a wide variety of robots, from a two-wheeled toy to a walking humanoid. Regardless of the complexity of the robot, there are two essential elements to model any robot: 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 a 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) [30]. Additionally, each link or joint can have some subtags that can define the characteristics of this link or joint. For instance, the link mass and inertia over different axes could be determined

inside the link tags. Fig. 1 shows one link and one joint of the modeled robot as an example of these tags.

```

<link name="base_link">
  <visual>
    <geometry>
      <cyylinder length="0.046" radius="0.1" />
    </geometry>
    <material name="silver" />
  </visual>
  <collision>
    <geometry>
      <cyylinder 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>

```

Fig.1. A portion of the URDF file containing one link (base_link) and one joint (middle1_link_joint) of the modeled robot.

The essential components (links and joints) for our robot model and the visual state of these components as a result of interpreting the URDF file using the RViz tool (a 3D visualization tool of ROS) are shown in Fig. 2.

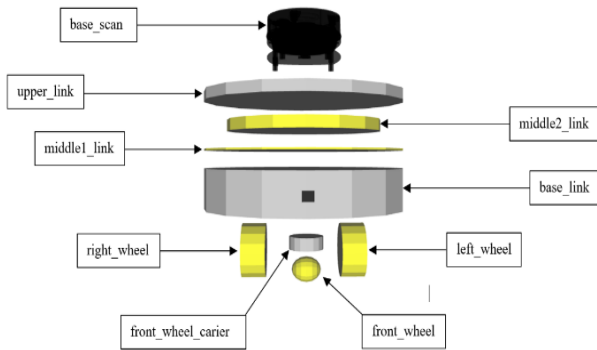


Fig.2. The URDF links that have been used to model our robot in ROS

After building the kinematics model of the robot in the previous stage, the model has to have some sensors to interact with the environment and solve our robot detection and relative positioning problem. The robot model has equipped with a differential driver and a Lidar sensor. The Lidar used to acquire surrounding data is the RPLidar A1, a 2D laser that can scan area ranges from 0.15cm up to 12m in all directions (360° of angular range) [39]. The RPLidar A1 runs clockwise to perform a 360° scan within a 12-meter range. The system measures distance data in more than 8000 times per second and with high-resolution distance output ($<1\%$ of the distance) and (≤ 1 of the angle). In this study, we have used a gazebo plugin related to

the laser sensor, where we have adjusted the plugin parameters to meet the specifications of the RPLidar sensor.

III. KIN DETECTION AND POSITIONING METHOD

The proposed method uses the Lidar sensor to collect the dataset and handle the kin detection and relative positioning task. As mentioned above, 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-based studies commonly use circular robots or any circular object attached to the robot (as in our case where we use a circular-shaped Lidar) and take advantage of the circularity of these robots or objects attached to them for implementing robot detection. The main idea of these studies is to find segments from Lidar measurements corresponding to objects in the environment and apply geometric or machine learning methods to find member robots. As Fig. 3 shows, our method implements kin detection task by applying the following steps: (1) acquisition of laser data and preprocessing (2) segmentation of data using the point-distance-based segmentation method, (3) extracting five features for each segment and applying the machine learning methods to classify segment and distinguish the kin robots from them, and (4) finding the relative position of the segments classified as a kin robot in the previous step.

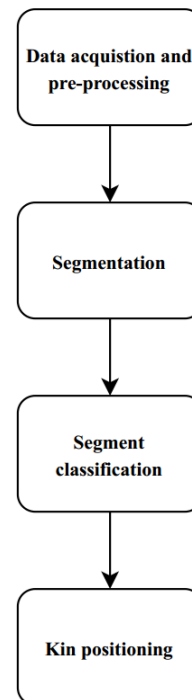


Fig.3. Flowchart of the kin detection method proposed in this paper

There are two assumptions for the proposed method.

- **Known radius:** The radius of the circular object/robot is known. In our case, the radius of Lidar RPLidar A1 is 0.035m.

- **No occlusion:** Each robot in the swarm is completely visible to the observer robot.

A. Data acquisition and preprocessing

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 the observer robot to the object and θ_i the relative angle of the measurement (see Fig. 4). First, the acquired Lidar data are stored as vectors (d_i, θ_i) .

There is a difference between the simulated Lidar model and the real Lidar. In the real world, the Lidar returns the maximum range value for objects outside its operating range (which means that there is no obstacle against the laser beams). However, for the laser sensor plugin that we used, the simulated Lidar gives an infinity value for such a case. So, to tackle that, we convert the infinity values obtained from the laser sensor plugin to the Lidar max range value (d_{max}). In the same way, any object located at (d_{max}) from the observer robot will not be visible to the Lidar.

It is also possible to apply some filtering to remove noise from the Lidar data in this stage. However, we did not apply any filtering in this study because we did not add noise to the simulated Lidar data. Fig. 4 represents the scanning system of the Lidar used, where one object has been placed at a distance (d_i) and an angle (θ_i).

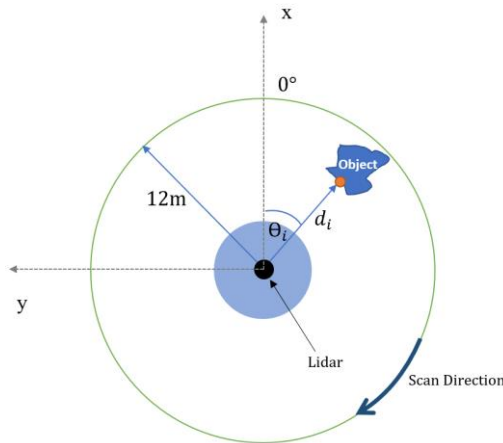


Fig.4. RPLidar A1 scanning system, where (d_i, θ_i) are the distance and the angle of the object relative to Lidar (Adapted from [30])

B. Segmentation

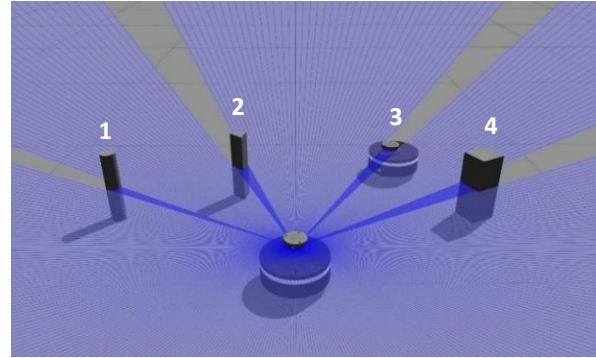
Segmentation is the process of transforming the raw laser points into groups of segments (useful data), which could be robots, humans, or other things. Segments can be defined as a set of range measurements (points) close to each other and probably belonging to one object. In this work, a PDBS based method is used for segmentation [30].

Segments are detected by using the derivative (∂_i) which can be calculated by finding the difference between each distance point (d_i) and the one before it (d_{i-1}). Then, small derivative values are filtered using a threshold value (d_{th}) which allows ignoring the small changes in scan data:

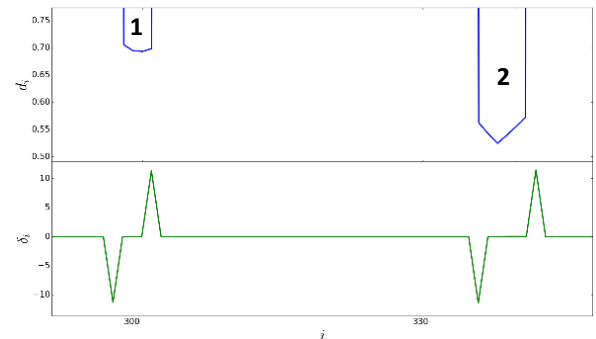
$$\partial_i = \{d_i - d_{i-1} \text{ if } |d_i - d_{i-1}| > d_{th}, 0 \text{ otherwise}\} \quad (1)$$

$$1 \leq i \leq 359$$

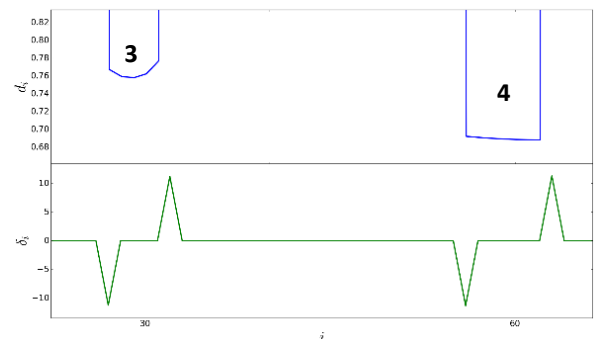
As a result of this operation, for each object in the environment, a falling slope ($\partial_i < -d_{th} < 0$) and a rising slope ($0 < d_{th} < \partial_i$) is detected (See Fig. 5 B and 5 C). By associating and combining each falling and rising edge, we obtain probable segments representing an object in the environment. Fig. 5 illustrates this step for four different objects (cube, kin, triangular prism, and cylinder) in the Gazebo simulator.



A. The observer robot and four different objects in the Gazebo simulator. The objects (from left to right) are: cylinder (1), triangular prism (2), kin robot (3), and a cube (4)



B. Representation of the Lidar scan data (blue lines) and the differences between each scan point and the previous one according to the ray number (green lines) for the two objects at the left of Fig. 5. A (object 1 and 2)



C. Representation of the Lidar scan data (blue lines) and the differences between each scan point and the previous one according to the ray number (green lines) for the two objects at the right of Fig. 5. A (object 3 and 4)

Fig.5. A representation of the segmentation process using four different objects in Gazebo simulator

C. Segment classification

Classification is the act or process of dividing things into groups according to their type (kin robot or not). Segment classification can be implemented using geometric [30] or machine learning [33] methods. In this study, the segment classification process has been performed using machine learning methods.

In machine learning, classification is a supervised learning approach. The computer program develops a model from the sample data (data set) and then uses this model to classify new observations. This data set may be bi-class (like our case where we attempt to identify whether the object is a robot or not) or multi-class.

This step aims to classify segments to distinguish the kin robots from the other objects. The classification process starts with collecting the segments' raw data and then converting these data to features. Five features have been extracted for each segment in the classification stage. Then various machine learning classification algorithms have been applied over these features, and results have been compared according to several performance measures (as described in the next section).

The features used are fitness, symmetry, radius error, circularity, and straightness, denoted by F_{fit} , F_{sym} , F_{rer} , F_{cir} and F_{str} respectively. These features have been suggested by [33], who have used these features to detect circular objects in polar coordinates. They have chosen SVM for classification, and just two object types (plastic cast and rectangle paper box) have been used to test the algorithm. However, this paper suggests using these features to distinguish kin robots from other possible obstacles and find their relative positions. We have tested several situations and scenarios for various objects using four different classification algorithms. The features used have been defined as follows:

1) Fitness

The fitness function evaluates how close a given solution is to the optimum solution of the desired problem. It determines how much a solution is fit. In our case, we can benefit from the fitness function to measure how the distance points related to a segment (object) fit the circular object representing the kin member that we are looking for by using the following equation.

$$F_{fit} = \frac{1}{\max(\theta_q) - \min(\theta_q)} \sum_{i=1}^N |\rho_q(i) - \rho(i)| \quad (2)$$

Where $(\rho(i), \theta(i))$ are the corresponding polar coordinates for scan distance points, and N is the number of these distance points. This feature could be used to extract circles and lines. If the target object is a circle or line, the corresponding F_{fit} is small. Otherwise, F_{fit} is a large number.

2) Circularity

The circularity feature describes how close an object is to a true circle. So, we can use this feature to show how an object is close to a circle with a known radius by using the following equations:

$$\alpha_{ct} = \sin^{-1} \frac{R_r}{\rho_{min} + R_r} \quad (3)$$

$$F_{cir} = |\alpha_{ct} - \alpha| \quad (4)$$

R_r is the radius of the Lidar used and ρ_{min} represents the minimal distance between Lidar and the object. While α represents the measured value of the angle between the maximal and the minimal distance between LRF and target object, α_{ct} denotes the theoretical calculation value of that angle. So, the difference of α_{ct} and α defines the circularity feature.

3) Radius error

For circle, we can calculate the difference between the theoretical calculation value of the circle radius (R_{est}) and the original value (R_r) which is in our case 0.035cm to define the radius error estimation feature as follows:

$$R_{est} = \frac{\rho_{min} \sin \alpha}{1 - \sin \alpha} \quad (5)$$

$$F_{rer} = |R_r - R_{est}| \quad (6)$$

In theory, for a circle, F_{cir} and F_{rer} are equal to zero. In practice, there are measurement errors, so F_{cir} and F_{rer} are not exactly equal to zero but are still close to zero. If the detected object is not a circle, F_{cir} and F_{rer} are large values.

4) Straightness

It is possible to calculate the straightness degree of the target object using the following equations:

$$\alpha_{lt} = \cos^{-1} \frac{\rho_{min}}{\rho_{max}} \quad (7)$$

$$F_{str} = |\alpha_{lt} - \alpha| \quad (8)$$

Where ρ_{min} and ρ_{max} represent the minimal and the maximal distance between Lidar and the target object. While (α) represents the measured value of the angle between the maximal and the minimal distance between LRF and the target object, (α_{lt}) denotes the theoretical calculation value of that angle. The difference of α_{lt} and α defines the straightness feature.

For both lines and circles, F_{str} should have a small value close to zero too. For objects with other shapes, F_{str} becomes a larger value.

5) Symmetry

Mathematically, symmetry means that one shape becomes exactly like another when you move it in some way: turn, flip, or slide. For two objects to be symmetrical, they must be the same size and shape but have a different orientation from each other. Since the Lidar used to distinguish robots is a cylindrical

object, the symmetry value we will obtain from the formula described below will measure the symmetry of a segment with a cylindrical object in Lidar size.

Generally, symmetry level is measured by the Hausdorff distance. However, in this paper, F_{sym} is simply computed by:

$$F_{sym} = \left| \frac{\max(\theta_q) - \min(\theta_q)}{2} - \theta_{fit} \right| \quad (9)$$

where θ_{fit} is the axis of symmetry, and it equals to $\theta_{fit} = -\frac{p_1}{2p_2}$.

According to the equations described above, it could be noticed that features will be affected by two factors:

- Whether an object is circular, line or not
- Whether the radius of the detected circular object is equal to the kin robot's lidar radius or not.

D. Kin positioning

After classifying the segments related to kin robots, it is necessary to identify the relative position of these kins. The kin positioning process involves estimating the distance and relative angle between the observer robot and other kins. As we have the distance points of the kin robot, we can obtain the relative position of the kin robot by getting the smallest value among the distance points. Finally, the index value corresponding to the smallest value can be easily used to obtain the relative angle of the kin robot.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

Experiments are performed in three stages. In the first stage, raw data collection is performed. In order to verify the validity of the proposed method, various object types (including the kins that we are looking for) have been used in the data collection stage (see Fig. 6). Different distances, positions, and sizes have been used for each object type. The data collection process covered the distance 0.2 m (the minimum distance that the Lidar can detect) to 1 m. The collected raw data contains the index of the first and the last rays that hit the object (or segment), the distance values in that range, and a class label ("1" for kins and "0" for non-kin objects). Some samples of the raw data set are shown in Table 1.

In the second stage, the raw data set is transformed into a feature data set. The feature data set contains five feature values for each segment and is stored in a CSV file. Some samples of the feature data set are shown in Table 2.

In the last stage, the feature data set is fed to different machine learning classifiers, where we used four different classification algorithms for this study which are: random forest (RF), SVM, k-nearest neighbor (kNN), and decision tree (DT). The collected datasets contain 2230 segments (850 frames for kins and 1380 for other shapes). The feature data set was divided into training and testing sets using the 20-fold cross-validation method. Different hyperparameters have been tried for each classifier to get the best result. Table 3 shows the best parameters obtained for each classification algorithm using the

grid search method. Table 4 shows the performance measures of the classification algorithms used where the best algorithm was the random forest with an 86% accuracy score.

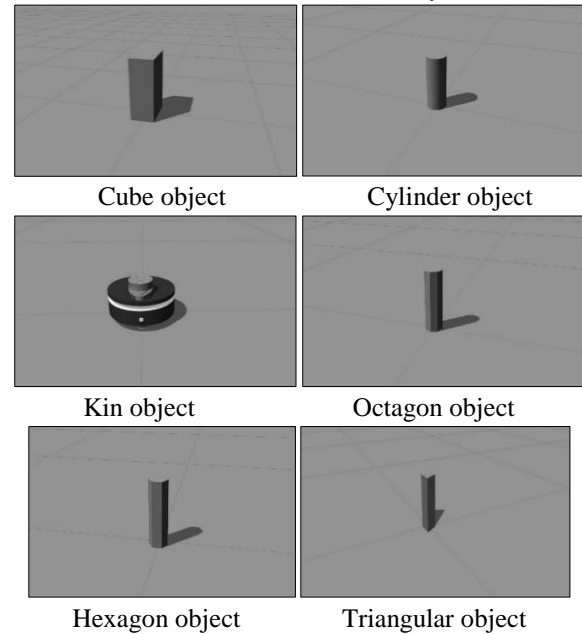


Fig.6. The objects used to collect data and extract the features in the segment classification process

TABLE I
SAMPLES OF THE GENERATED RAW DATA SET OBTAINED BY LIDAR FOR KINS AND OTHER OBJECTS. THE KIN ROBOTS WERE LABELLED WITH (1), AND NON-KINS WERE LABELLED WITH (0)

Seg No	Start index	End index	Distance Points			Kin or not	
1	173	184	0.21	0.21	0.21	...	0
2	264	270	0.43	0.42	0.41	...	0
3	239	242	0.77	0.76	0.76	...	0
4	174	183	0.38	0.37	0.37	...	0
5	201	204	1.09	1.08	1.07	...	0
6	267	272	0.70	0.69	0.68	...	0
7	155	171	0.23	0.22	0.22	...	1
8	165	171	0.59	0.58	0.57	...	1
9	121	125	0.75	0.74	0.74	...	1
10	204	208	0.81	0.79	0.79	...	0
11	216	225	0.35	0.35	0.34	...	0
12	263	267	0.89	0.87	0.87	...	0
13	174	184	0.34	0.33	0.33	...	1
14	177	181	0.88	0.86	0.86	...	1
15	262	265	0.98	0.96	0.96	...	1
16	174	183	0.29	0.29	0.29	...	0
17	270	274	0.73	0.73	0.73	...	0
18	272	286	0.22	0.22	0.22	...	0
19	143	151	0.47	0.46	0.45	...	1
20	100	104	0.75	0.75	0.74	...	1
21	340	346	0.53	0.52	0.52	...	1
22	172	182	0.32	0.32	0.32	...	0
23	175	180	0.62	0.62	0.61	...	0

TABLE II

THE FIVE FEATURES EXTRACTED FOR THE SEGMENTS SHOWN IN TABLE 1. THE FEATURES ARE FITNESS, SYMMETRY, ESTIMATE RADIUS ERROR, CIRCULARITY, AND STRAIGHTNESS DENOTED BY F_{fit} , F_{sym} , F_{rer} , F_{cir} AND F_{str}

Seg No	F_{fit}	F_{cir}	F_{rer}	F_{str}	F_{sym}	Kin or not
1	0.03	0.04	0.01	0.25	3.11	0
2	0.04	0.03	0.01	0.23	4.66	0
3	0.03	0.01	0.01	0.17	4.19	0
4	0.12	0.00	0.00	0.30	3.11	0
5	0.33	0.01	0.02	0.24	3.53	0
6	0.12	0.00	0.00	0.24	4.70	0
7	0.08	0.01	0.00	0.34	2.84	1
8	0.06	0.01	0.00	0.24	2.93	1
9	0.04	0.01	0.01	0.19	2.14	1
10	0.04	0.01	0.01	0.16	3.59	0
11	0.07	0.01	0.00	0.26	3.84	0
12	0.02	0.00	0.00	0.21	4.62	0
13	0.05	0.01	0.00	0.27	3.12	1
14	0.05	0.00	0.00	0.20	3.12	1
15	0.02	0.00	0.00	0.14	4.60	1
16	0.00	0.05	0.02	0.26	3.11	0
17	0.00	0.02	0.02	0.02	4.74	0
18	0.00	0.10	0.03	0.05	4.87	0
19	0.09	0.00	0.00	0.27	2.56	1
20	0.03	0.01	0.01	0.18	1.77	1
21	0.02	0.01	0.01	0.19	5.98	1
22	0.12	0.03	0.01	0.35	3.08	0
23	0.15	0.02	0.01	0.24	3.09	0

TABLE III

THE BEST VALUES OF HYPERPARAMETERS OBTAINED USING THE GRID SEARCH METHOD

Algorithm	Parameter	Value
Decision tree	criterion	entropy
	max depth	6
	min samples leaf	8
Random forest	criterion	entropy
	number of trees	100
KNN	number of neighbours	38
SVM	kernel function	rbf
	regularization parameter	9

This paper proposed a kin detection and positioning algorithm. Due to the lack of studies that use the Lidar sensor to implement such a task, we have chosen to use the lidar-based approach to tackle this task. Our method uses a robot model equipped with a Lidar sensor to detect the different objects in an environment, distinguish the kin robot member and find the relative position of each kin using the machine learning

classification methods. The features used in the classification process have been suggested by Zhou et al. [33] to detect circular objects using SVM. However, in this study, we have recommended using these features to distinguish kin robots equipped with Lidar sensors from other possible obstacles and then find the relative positions of each detected kin.

TABLE IV

PERFORMANCE MEASURES OBTAINED ACCORDING TO THE CLASSIFICATION METHODS USED FOR THE KIN DETECTION TASK WHERE THE RANDOM FOREST ALGORITHM GIVES THE BEST ACCURACY SCORE (86%).

	Accuracy	Precision	Recall	F1_score
DT	85%	87%	85%	84%
RF	86%	86%	86%	86%
kNN	84%	87%	84%	83%
SVM	82%	86%	82%	81%

Several situations and scenarios for various objects are tested using four different classification algorithms. The algorithms used are decision tree, random forest, k-nearest neighbors, and SVM. The random forest algorithm gave the best result for the segment classification with an 86% accuracy score.

The robot model used in the experiments is a model of an early version of the swarm robotic platform called Layka, developed by the second author. The Layka swarm robotic platform being developed is designed to be used with or without ROS. There are some advantages of using ROS for developing swarm robotic systems. First, tools and mechanisms supported by ROS can simplify debugging swarm robotic systems considerably. Second, ROS mechanisms and abstractions allow developing modular and platform-independent components. Third, ROS has many high-level libraries (such as navigation and planning) that can be valuable for swarm robotic systems. In future works, we plan to finalize the development of the Layka swarm robotic platform, test proposed methods on the real robot platform, and analyze the effect of sensor noise which is one of the limitations of this study and the previous one [35].

REFERENCES

- [1] R. S. Ortigoza, M. Marcelino-Aranda, G. S. Ortigoza, V. M. H. Guzman, M. A. Molina-Vilchis, G. Saldana-Gonzalez, J. C. Herrera-Lozada, M. Olguin-Carbajal. "Wheeled mobile robots: a review." *IEEE Latin America Transactions*, 10(6), 2012, pp2209-2217.
- [2] F. Rubio, F. Valero, C. Llopi-Albert. "A review of mobile robots: Concepts, methods, theoretical framework, and applications." *International Journal of Advanced Robotic Systems*, 16(2), 2019, 1729881419839596.
- [3] E. Şahin, "Swarm Robotics: From sources of inspiration to domains of application," in *Swarm Robotics*, E. Şahin and W. M. Spears, Eds. 2005, pp. 1–9.

- [4] G. Beni, "The concept of cellular robotic system," in *IEEE International Symposium on Intelligent Control*, 1988, pp. 57–62, DOI: 10.1109/isic.1988.65405.
- [5] T. Fukuda and S. Nakagawa, "Approach to the dynamically reconfigurable robotic system," *Journal of Intelligent and Robotic Systems*, vol. 1, no. 1, pp. 55–72, 1988, DOI: 10.1007/bf00437320.
- [6] G. Beni, "From swarm intelligence to swarm robotics," in *International Workshop on Swarm Robotics*, 2004: Springer, pp. 1-9.
- [7] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: a review from the swarm engineering perspective," *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, 2013, DOI: 10.1007/s11721-012-0075-2.
- [8] K. Bolla, T. Kovacs, and G. Fazekas, "Compact Image Processing Based Kin Recognition, Distance Measurement and Identification Method in a Robot Swarm," in *International Joint Conference on Computational Cybernetics and Technical Informatics*, 2010, pp. 419–424, DOI: 10.1109/icccyb.2010.5491237.
- [9] I. Rekleitis, G. Dudek, and E. Milios, "Multi-robot collaboration for robust exploration," *Annals of Mathematics and Artificial Intelligence*, vol. 31, no. 1–4, pp. 7–40, 2001, DOI: 10.1023/a:1016636024246.
- [10] Y. Han and H. Hahn, "Visual tracking of a moving target using active contour based SSD algorithm," *Robotics and Autonomous Systems*, vol. 53, no. 3–4, pp. 265–281, 2005, DOI: 10.1016/j.robot.2005.09.005.
- [11] K. Bolla, Z. Istenes, T. Kovacs, and G. Fazekas, "A Fast Image Processing Based Robot Identification Method for Surveyor SRV-1 Robots," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2011, pp. 1003–1009, DOI: 10.1109/aim.2011.6027147.
- [12] F. Rivard, J. Bisson, F. Michaud, and D. Létourneau, "Ultrasonic Relative Positioning for Multi-Robot Systems," in *IEEE International Conference on Robotics and Automation*, 2008, pp. 323–328, doi: 10.1109/robot.2008.4543228.
- [13] L. E. Navarro-Serment, C. J. Paredis, and P. K. Khosla, "A beacon system for the localization of distributed robotic teams," in *International Conference on Field and Service Robotics*, 1999, vol. 6, pp. 1–6.
- [14] C.-J. Wu and C.-C. Tsai, "Localization of an Autonomous Mobile Robot Based on Ultrasonic Sensory Information," *Journal of Intelligent and Robotic Systems*, vol. 30, no. 3, pp. 267–277, 2001, DOI: 10.1023/a:1008154910876.
- [15] I. Kelly and A. Martinoli, "A scalable, on-board localisation and communication system for indoor multi-robot experiments," *Sensor Review*, vol. Volume 24, no. Issue 2, pp. 167–180, 2004, DOI: 10.1108/02602280410525968.
- [16] G. Caprari and R. Siegwart, "Design and control of the mobile micro robot alice," in *2nd International Symposium on Autonomous Minirobots for Research and Edutainment*, 2003, pp. 23--32.
- [17] F. Arvin, K. Samsudin, and A. R. Ramli, "A Short-Range Infrared Communication for Swarm Mobile Robots," in *International conference on signal processing systems*, 2009, pp. 454–458, DOI: 10.1109/icsps.2009.88.
- [18] F. Mondada *et al.*, "The e-puck, a robot designed for education in engineering," in *9th conference on autonomous robot systems and competitions*, 2009, vol. 1, pp. 59--65.
- [19] S. Kornienko, "IR-based Communication and Perception in Microrobotic Swarms," in *7th Workshop on Collective & Swarm Robotics*, 2010.
- [20] A. E. Turgut, F. Gokce, H. Celikkanat, L. Bayindir, and E. Sahin, "Kobot: A mobile robot designed specifically for swarm robotics research," *METU-CENG-TR Tech. Rep*, vol. 5, no. 2007, Middle East Technical University, Ankara, Turkey, 2007.
- [21] F. Mondada, A. Guignard, M. Bonani, D. Bär, M. Lauria, and D. Floreano, "SWARM-BOT: From Concept to Implementation," 2003, vol. 2, pp. 1626–1631, doi: 10.1109/iros.2003.1248877.
- [22] F. Arvin, J. Murray, C. Zhang, and S. Yue, "Colias: An Autonomous Micro Robot for Swarm Robotic Applications," *International Journal of Advanced Robotic Systems*, vol. 11, no. 7, p. 113, 2014, DOI: 10.5772/58730.
- [23] J. McLurkin, A. McMullen, N. Robbins, G. Habibi, A. Becker, A. Chou, H. Li, M. John, N. Okeke, J. Rykowski, S. Kim, W. Xie, T. Vaughn, Y. Zhou, J. Shen, N. Chen, Q. Kaseman, L. Langford, J. Hunt, A. Boone, K. Koch. A robot system design for low-cost multi-robot manipulation. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)*, 2014.
- [24] F. B. Malavazi, R. Guyonneau, J. B. Fasquel, S. Lagrange, F. Mercier. "LiDAR-only based navigation algorithm for an autonomous agricultural robot.",

- Computers and electronics in agriculture*, 154, 2018, 71-79.
- [25] I. Belkin, A. Abramenko, D. Yudin. "Real-time lidar-based localization of mobile ground robot." *Procedia Computer Science*, 186, 2021, 440-448.
- [26] D. Zhang, J. Cao, G. Dobie, C. MacLeod. "A framework of using customized LIDAR to localize robot for nuclear reactor inspections." *IEEE Sensors Journal*, 2021.
- [27] C. Benedek, A. Majdik, B. Nagy, Z. Rozsa, T. Sziranyi. "Positioning and perception in LIDAR point clouds." *Digital Signal Processing*, 119, 2021, 103193.
- [28] S. Royo, M. Ballesta-Garcia. "An overview of lidar imaging systems for autonomous vehicles." *Applied sciences*, 9(19), 4093.
- [29] Y. Li, L. Ma, Z. Zhong, F. Liu, M. A. Chapman, D. Cao, J., Li. "Deep learning for LiDAR point clouds in autonomous driving: a review." *IEEE Transactions on Neural Networks and Learning Systems*, 32(8), 2020.
- [30] C. Premebida and U. Nunes, "Segmentation and geometric primitives extraction from 2d laser range data for mobile robot applications," *Robotica*, vol. 2005, pp. 17--25, 2005.
- [31] M. Teixidó, T. Pallejà, D. Font, M. Tresanchez, J. Moreno, and J. Palacín, "Two-Dimensional Radial Laser Scanning for Circular Marker Detection and External Mobile Robot Tracking," *Sensors*, vol. 12, no. 12, pp. 16482–16497, 2012, doi: 10.3390/s121216482.
- [32] A. Wąsik, R. Ventura, J. N. Pereira, P. U. Lima, and A. Martinoli, "Lidar-based relative position estimation and tracking for multi-robot systems," in *Robot 2015: Second Iberian Robotics Conference*, 2016, pp. 3–16, DOI: 10.1007/978-3-319-27146-0_1.
- [33] X. Zhou, Y. Wang, Q. Zhu, and Z. Miao, "Circular object detection in polar coordinates for 2D LIDAR data," in *Chinese Conference on Pattern Recognition*, 2016, pp. 65–78, DOI: 10.1007/978-981-10-3002-4_6.
- [34] K. Dietmayer, "Model-Based Object Classification and Object Tracking in Traffic Scenes from Range-Images," in *IV2001*, 2001, pp. 25--30.
- [35] Z. Yılmaz and L. Bayındır, "Simulation of Lidar-Based Robot Detection Task using ROS and Gazebo," *European Journal of Science and Technology*, pp. 513–529, 2019, doi: 10.31590/ejosat.642840.
- [36] M. Quigley *et al.*, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, 2009, vol. 3, p. 5.
- [37] N. Koenig and A. Howard, "Design and use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, 2004, vol. 3, pp. 2149–2154, DOI: 10.1109/iros.2004.1389727.
- [38] L. Kunze, T. Roehm, and M. Beetz, "Towards Semantic Robot Description Languages," 2011, vol. 1, pp. 5589–5595, DOI: 10.1109/icra.2011.5980170.
- [39] "Laser range scanner RPLIDAR A1 Datasheet." [Online]. Available: <https://download.slamtec.com/api/download/rplidar-a1m8datasheet/2.1?lang=en>. [Accessed: 14-Mar-2020]

BIOGRAPHIES



ZAHİR YILMAZ graduated from Damascus University in Syria Department of Computer Engineering and Automation in 2015. He received his MSc in Computer Engineering from Ataturk University in Turkey in 2020. His primary research interests are swarm robotics and machine learning.



LEVENT BAYINDIR received his PhD in Computer Engineering from Middle East Technical University, Turkey. He is currently an assistant professor of Computer Engineering at Atatürk University in Erzurum, Turkey. His research interests include swarm robotics and pervasive computing.