A multi-robot simulator for the evaluation of formation control algorithms

Aakash Soni, Huosheng Hu
School of Computer Science and Electronic Engineering, University of Essex, Colchester, CO4 3SQ, United Kingdom
aakash.soni@essex.ac.uk, hhu@essex.ac.uk

***Abstract*—Autonomous/unmanned driving has the capability to provide numerous benefits such as better traffic management, increased safety, reduced emission, and enhanced transportation network. Once autonomous ground vehicles (AGVs) are deployed, they will have to interact with other such vehicles. Interaction between multiple AGVs is an important area of research where analysis on the performance of algorithms/control schemes of AGVs is carried out. Performing real-world experiments with teams of autonomous vehicles is a challenging task due to cost and complexity. On the other hand, a simulation can emulate reality and provide an inexpensive and less time-consuming development process compared to the real world robots' testing. Therefore, a simulation tool is developed for multi-robot navigation. This simulator is based on open-source Robot Operating Systems (ROS) and natively supported robotics simulator Gazebo.**

***Index Terms*—autonomous robots, platoon formation, laser scanning, simulation**

I. Introduction

Autonomous driving is state-of-the-art technology in the transportation industry. Many companies are working towards developing fully autonomous vehicles by achieving the fifth level of autonomy [1], [2]. As AGVs are being developed, the problem of collaboration between these vehicles needs to be studied. Vehicle platooning or formation control is a part of the collaboration problem. In this paper, a vehicle platooning using only laser sensor is addressed by proposing a platoon control algorithm. A vehicle platooning is a well-observed scenario of everyday transport networks. Human operated vehicles usually follow another vehicle while maintaining safe speed and distance hence forming a platoon formation. The aim of the platoon formation control is to confirm that all vehicles in a platoon move at the same speed while maintaining a desired formation shape or geometry, which is stated by a desired inter-vehicle spacing strategy [3].

The developed algorithms or controllers require a simulation platform to evaluate the performance of AGVs and their algorithms. A simulator is a useful tool to test robot hardware configuration, software integration and the interaction with other robots and the environment. Furthermore, simulation provides an idea of the robots' behaviour on implemented algorithms and is crucial to rapidly reproduce experiments. Several packages [4]–[8] were considered before finalising ROS and Gazebo platform. ROS is an open-source system and well adopted by the research community, whereas Gazebo is also an open-source simulator and offering the ability to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments.

The contribution of this paper is threefold. First, it presents a multi-robot simulator detailing virtual robot and world development using ROS-Gazebo framework. Second, it proposes a simple vehicle platoon formation algorithm capable of handling multiple AGVs based on laser scanner. Third, the proposed algorithm does not require any communication between AGVs thus, capable of guiding AGVs in an uncertain situation such as sudden loss of communication or time-varying communication delay. The application of this algorithm is not only limited to work discussed in this paper. This algorithm can be useful to guide warehouse mobile robots for goods transportation as well as to study the robot behaviour of follower robots in the laboratory-based environment.

The remainder of this paper is organised as follow, Section II presents the literature review on robotics simulators and platoon algorithms, Section III discusses a package description detailing simulation world, robot development and tools used to design proposed simulator, Section IV presents algorithms for proposed vehicle platoon control, Section V discusses the experimental results and section VI is a conclusion and future works.

II. Related Work

Several simulators are developed using Gazebo, Stage, Webots and Morse simulation packages for multi-robot operations. With a goal to assist elderly people through mobile robots, a simulator was developed capable of performing 3D mapping and autonomous navigation using ROS and Gazebo [9]. In this work, two robots were designed for the indoor simulation and experiments were carried out using the same software, developed for simulation, on physical robots. Another study developed a multi-robot simulator for underwater monitoring operations using ROS and Gazebo by contributing several plugins, underwater vehicles and simulation world to enable underwater applications [10].

A multi-robot simulator was developed using ROS and Stage for benchmarking algorithms for robot patrolling task [11]. In this work, algorithm testing was performed on both simulation and physical experiments, and the developed algorithm was able to work with other simulation packages. Another research developed multi-robot simulator using ROS and Morse to validate, benchmark and compare different algorithms [12]. To compute benchmarking process, several

parameters were considered, such as exploration time, cost, safety and efficiency, and map completeness and quality.

A couple of studies have used Webots simulator for testing algorithms. Using Webots simulator, a distributed controller was proposed for the multi-lane heterogeneous vehicles by incorporating lateral controller for vehicles to stay in the lane and longitudinal controller for desired inter-vehicle distance [13]. The proposed controller was able to adapt to the shape of the curvilinear road shape. Another study presented a distributed and dynamic graph-based formation control approach for vehicles to join, leave, or change lanes without affecting the stability of the convoy. Here, the particle swarm optimization algorithm was implemented to optimize the parameters of the control law (lane keeping and obstacle avoidance) and to reduce the overall formation control errors [14]. In this paper, a vision sensor was used for the lane keeping, and LIDAR was used for obstacle avoidance.

A number of studies are carried out for vehicle formation control by proposing distributed and decentralized controllers [15]. An active vision-based adaptive leader-follower formation control was achieved in the absence of communication [16]. In this study, a follower robot was tracking the features of a leader robot through a camera by developing two controllers, a formation controller to maintain formation and a camera controller to provide visual measurements. Furthermore, a vision-based leader-follower formation control was achieved by developing a neural-dynamic optimization-based non-linear model predictive control (MPC) [17]. In this study, a camera on a follower robot was employed to track the features and to measure state and velocity of the leader robot.

The development of autonomous following lateral control techniques without inter-vehicle communication was proposed for follower vehicle by measuring the relative position of the preceding vehicle using laser range sensor [18]. Here, the longitudinal distance was assumed to be constant, and relative displacement and yaw angle were used to derive the proposed controller. Similar work was carried out in [19] where control algorithm was derived by measuring the relative position of the ego-vehicle with respect to its preceding vehicle using a laser sensor. Another work used onboard sensors for trajectory generation method in which follower robot's accurately replicate the leader robot's path in both on-road and off-road environments [20]. In this work, the trajectory for a follower robot was generated using sequential quadratic programming.

Several theoretical and real-world robot experiments are performed for leader-follower formation problem and several simulators are developed for multi-robot applications. Number of papers have addressed multi-robot navigation problem through simulation whereas few papers targets formation control problem using only sensors. The signification of this work is the combination of above discussed works. In this paper, a multi-robot simulator is proposed by implementing a platoon formation algorithm using laser sensors.

III. Package Description

The architecture of the multi-robot simulator includes three parts: a gazebo version-7 simulator, application specific robot controllers and ROS kinetic middleware. This architecture is flexible meaning that a user can develop or edit the existing world, different algorithms can be analysed, and new robots can be developed and deployed. Figure 1 shows the system overview.

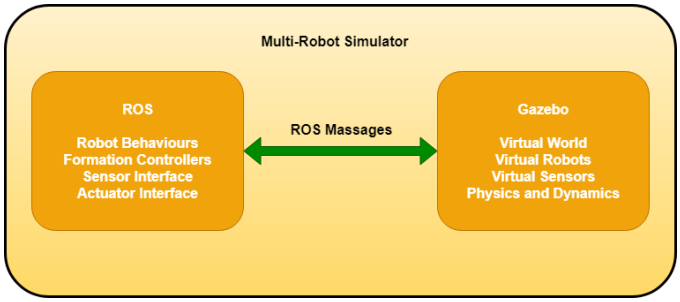

Figure 1. System Overview

A. Robot Operating Systems

Several robotics frameworks are developed [4], [5], [8] but ROS has proved to be a reliable and popular framework amongst research communities and has become the standard for the robotics research and development. The primary goal of ROS is to provide support for code reuse in robotics research. It should be noted that ROS is not a real-time framework but can be integrated with real-time code.

The main services provided by ROS are hardware abstraction, low-level device control, implementation of commonly-used functionalities, message-passing between processes and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers. Furthermore, ROS offers modularity that uncouples the control software from the drivers of the robot body, allowing to use exactly the same control software in simulations and in real robot experiments. The goal of the ROS framework can be described as:

- Sharing: Ability to share processes by grouping them into packages and stacks
- Collaboration: ROS supports a federated system of code repositories that enable collaboration to be distributed
- Thin: Code written for ROS can be used with other robot software frameworks
- ROS-agnostic libraries: The preferred development model is to write libraries with clean functional interfaces
- Language independence: Ability to use Python, C++, Java, Matlab and LISP programming languages
- Easy testing: Built-in test framework called rostest makes it easy to bring up and tear down test fixtures
- Scaling: Ability to handle large runtime systems and large development processes

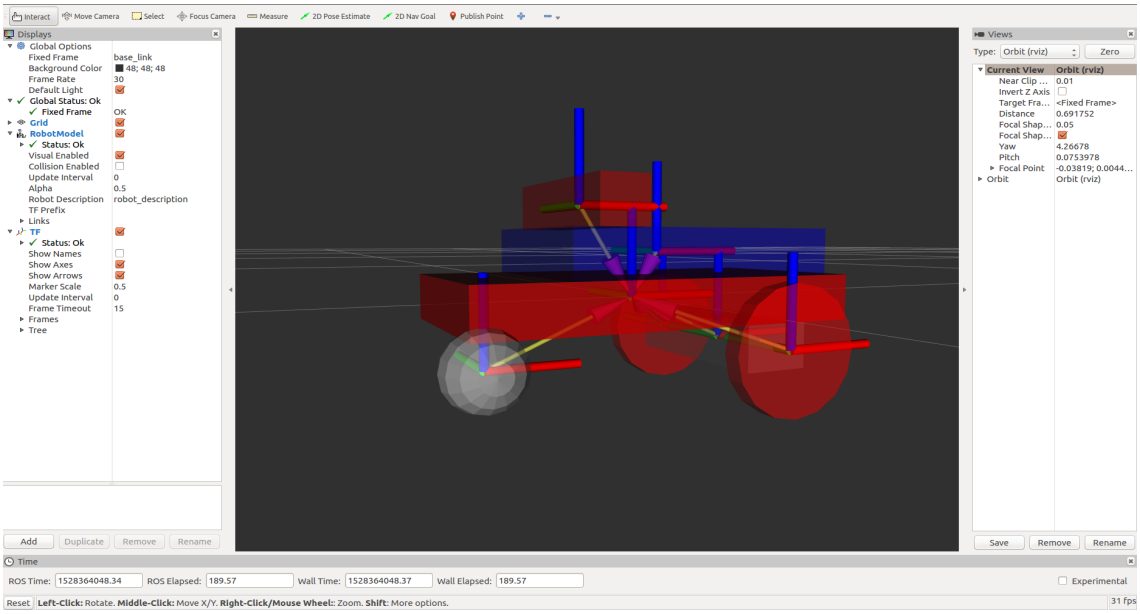

Figure 2. Robot Development

B. Gazebo Simulation Package

Gazebo is an open source 3D dynamic simulator with the ability to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments while offering a higher degree of fidelity, a suite of sensors, and interfaces for both users and programs. This effective, scalable and simple tool can be used to test robotics algorithms, designing robots and performing regression testing with realistic scenarios.

Gazebo supports four different physics engine, ODE, Bullet, Simbody and DART, which makes Gazebo capable of rigid-body dynamics simulation. Out of these four physics engines, one engine must be defined within the world model description. Moreover, Gazebo offers a rich library of models such as robots, sensors, actuators and arbitrary objects. For these models, Gazebo maintains a simple API and the necessary hooks for interaction with client programs. A layer below this API resides the third party libraries that handle both the physics simulation and visualization.

C. Robot Development

For simulation, a robot model is developed using Unified Robot Description Format (URDF) which is an XML format for representing a robot model. Here, XML format describes robot's joints, visualisation, appearance and integration of various controllers and their parameters. Figure 2 shows the developed robot in RVIZ (ROS Visualisation) software. This URDF file is then converted into the XACRO (XML Macros) format. The benefit of using XACRO is that it produces a more readable and shorter version of XML files. Furthermore, RVIZ is a 3D visualiser for displaying sensor data and state information from ROS. RVIZ also provide a chance to visualise the robot's joint, how they move/rotate and their connections to each other.

This robot has a differential steering locomotion system meaning that the robot movement changes by varying the speed of the two rear wheels individually. A caster wheel is placed in the front which balances and supports the robot movement. The wheels are controlled by the plugin called libgazebo_ros_diff.so and their parameters are application dependent and need to be defined in XACRO file. The robot design is inspired by the EMoRo Robot[1] developed by Inovatic ICT.

D. Laser Sensor

A laser sensor plugin used in this experiments is called libgazebo_ros_laser.so and is based on hokuyo UTM-30LX sensor. For a leader robot, the scanning range for this sensor is set between [-90°, 90°] or 720 samples and the measurement range is set between [0.30m, 3.0m]. For follower robots, the scanning range is [-11.5°, 11.5°] or 720 samples and the measurement range is [0.30m, 1.0m]. The reason behind setting a lower scanning range for follower robots compared to the leader robot is because follower robots only measure the range between the preceding robot and robot itself. Furthermore, setting a lower measurement range for follower robots ensures that the distance between leader and follower do not exceed more than one meter.

E. Differential Drive Controller

In this experiment, an inbuilt differential drive controller plugin called libgazebo_ros_diff_drive.so is used to control the rear wheels of the robot. This controller accepts velocity command and sent on the two wheels of a differential drive

[1]http://www.emoro.eu/

wheelbase. The controller extracts the x component of the linear velocity and z components of angular velocity while ignoring the velocities of other components.

F. World Development

A simulation world is used to analyse the robots' movement. The world created for this study includes several features such as double lanes, two-way traffic lanes and intersections. This world is reconfigurable, depending on the test requirement. After developing the world, ten robots were deployed in the environment, as shown in Figure 3. In this work, the simulation world is made as simple as possible using resources available though Gazebo. However, numerous models are available in Gazebo model library and can be added to this world such as buildings, sign-posts, barriers, and tree.

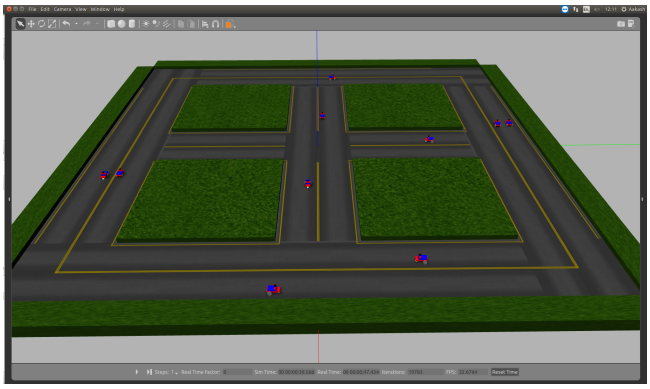

Figure 3. Simulation World with Robots

IV. The Proposed Approach

Figure 4 shows the predecessor following topology. For sensor-based platoon formation, it is a better choice compared to other topologies presented in [15] because each robot's sensory measurement is derived by scanning the preceding robot as discussed in this approach. Moreover, this form of topology is usually created under communication loss between vehicles.

Figure 4. Predecessor Following Topology

A. Leader Robot Algorithm

The algorithm for a leader robot's navigation is based on simple if-else conditions. Based on the laser sensor's reading, actuators output relative linear and angular velocity. The leader robot tries to stay in the right lane and turns left when near to obstacle or wall. The laser sensor has samples range of 720 and is divided into five regions. The sample range between [0:143] is called right region, [144:287] is called front right region (fright in algorithm), [288:431] is called front region, [432:575] is called front left region (fleft in algorithm) and [576:719] is called left region. Based on these regions, relative linear velocity and angular velocity is applied to the leader.

Algorithm 1: Leader Robot

Input : Laser Data in Sample Ranges 0 to 719
Output: Motor Actuation
`/* Decide a region and move, Distance/Range is in meter and velocity is in meter/second */`
Determine region in which leader robot is located
if *Front* $>$ *1 and Fleft* $>$ *1 and Fright* $>$ *0.5* **then**
  linearVel = 0.1
  angularVel = 0
end
else if *Front* $<$ *2 and Fleft* $<$ *3 and Fright* $>$ *0.5* **then**
  linearVel = 0.1
  angularVel = 0
end
else if *Front* $<$ *1 and Fleft* $<$ *1 and Fright* $<$ *1* **then**
  linearVel = 0
  angularVel = -0.1
end
else if *Front* $<$ *1 and Fleft* $>$ *1 and Fright* $<$ *1* **then**
  linearVel = 0
  angularVel = -0.1
end
else if *Fright* $>$ *0.7 and Front* $>$ *1 and Fleft* $>$ *1* **then**
  linearVel = 0
  angularVel = 0.1
end
else if *Fright* $<$ *0.5 and Front* $>$ *1 and Fleft* $>$ *1* **then**
  linearVel = 0
  angularVel = -0.1
end
else
  linearVel = 0
  angularVel = 0
end

B. Follower Robot Algorithm

The follower robots receive information about preceding robots via a laser sensor. In this algorithm, the scanning range of follower robots is set between [-11.5°, 11.5°], therefore they only scan robot in front of them. For this algorithm, ranges are stored for polar coordinate analysis. These coordinates are used to calculate the width of the preceding robot. The next step is to calculate the distance between the follower robot and preceding robot for longitudinal control. If the distance between these robots is sufficient, linear velocity and angular velocity are calculated and applied to follower robots.

V. Experimental Results

The proposed controller is implemented using Python language. As shown in Figure 5, readings from laser sensors drive

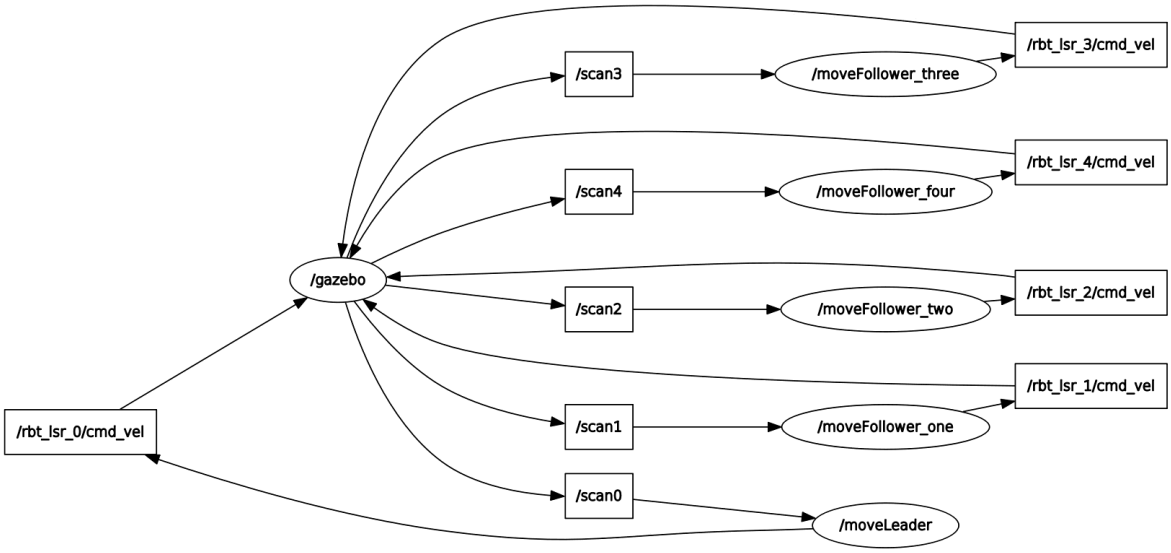

Figure 5. RQT Graph of Leader-follower Navigation

Algorithm 2: Follower Robots

Input : Laser Data in Sample Range 0 to 719
Output: Motor Actuation

```
/* Determine the leader's position */
for i = 0 to 719 do
    if sampleRange[i+1] - sampleRange[i] > 0.5 then
        Store the index i in diff
    end
end
for i = 0 to len(diff) do
    x1Coord = laser(diff)*cos(diff)
    y1Coord = laser(diff)*sin(diff)
    x2Coord = laser(diff+1)*cos(diff+1)
    y2Coord = laser(diff+1)*sin(diff+1)
    dist = d((x1Coord, y1Coord), (x2Coord, y2Coord))
    if dist > 0.5m then
        x1 = x1Coord, x2 = x2Coord
        y1 = y1Coord, y2 = y2Coord
    end
end
xclose = (x1[i+1] + x2[i+2]) / 2
yclose = (y1[i+1] + y2[i+2]) / 2
distClose = d(xclose, yclose)
if distClose > 0.1m then
    theta = atan2(yclose/xclose)
    angvel = theta - 90
    r = distclose / (2 * sin(theta/2))
    linvel = r * theta
end
```

the robots around the simulation world. For this experiment, five robots were deployed to test the developed controller. A leader robot has a predefined path, and its associated controller is /moveLeader based on leader robot's algorithm whereas follower robots' associated controllers are (/moveFollower_one to /moveFollower_four) based on follower robot's algorithm. All five robots are subscribed to their laser scan topics (/scan0 to /scan4). Here, a simulation world (/gazebo) provides laser readings from robots (/scan0 to /scan4) to the developed robot controllers and the controllers issue relevant velocity (/rbt_lsr_0/cmd_vel to /rbt_lsr_4/cmd_vel) to the robots in the simulation world (/gazebo).

Figure 6 shows the simulated scenario containing five mobile robots. While moving through the environment, this platoon stays in the right lane and turns without collision at the end of the lane. Here, the follower robots follow the trajectories of preceding robots and leader robot plans the trajectory based on the sensor's reading. Moreover, both linear and angular velocity are calculated and applied. For this experiment, the velocity for a leader robot is set at minimum to achieve accurate results from the algorithm. While trying the higher velocity, follower robots were not able to track the leader robot.

It is possible to add more robots for this experiment. However, adding more robots increases the simulation time and drops the frame rate of the simulator due to the limited computing power. Furthermore, having a platoon formation of a maximum of five vehicles is sufficient when considering a real-world scenario and the second platoon can be formed for the next five vehicles.

VI. Conclusion and Future Work

In this paper, a multi-robot simulator is developed and used to simulate formation control algorithm. A laser scanner based platoon formation algorithm is presented and simulated using

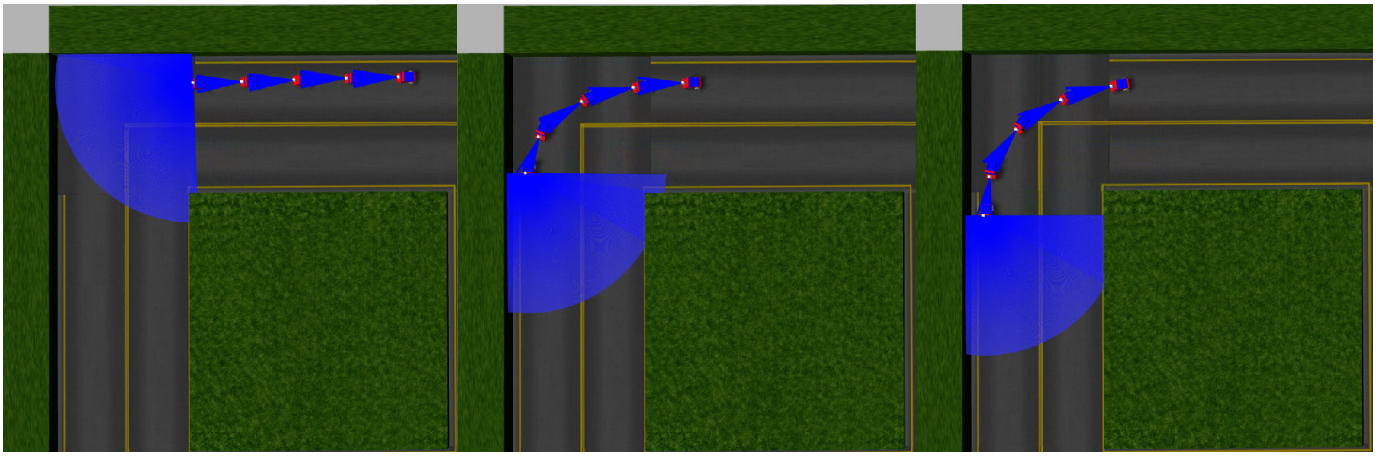

Figure 6. Platoon Navigation

predecessor following topology. The proposed approach could be useful for vehicle platoon formation during communication loss between connected vehicles. The experiment performed in this paper shows the ability to maintain platoon formation through linear and angular velocity.

The major outcome of this work is the ability to simulate platoon formation using laser sensor. In future, this algorithm will be made adaptable to a higher speed by considering the local properties of the road, dynamic obstacle will be simulated and their effects will be studied, and multiple platoons will be simulated to study the interaction between multiple vehicles and platoons. Moreover, this simulator will be made open-source.

REFERENCES

[1] The SAE On-Road Automated Vehicle Standards Committee. *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*; SAE International: Warrendale, PA, USA, 2018.

[2] UK Government. Centre for Connected and Autonomous Vehicles. Available online: https://www.gov.uk/government/organisations/centre-for-connected-and-autonomous-vehicles (accessed on 25 May 2019).

[3] Kavathekar, P.; Chen, Y.Q. Vehicle Platooning: A Brief Survey and Categorization. In Proceedings of the 2011 ASME/IEEE International Conference on Mechatronic and Embedded Systems and Applications, Washington, DC, USA, August 2011.

[4] Harris, A.; Conrad, J.M. "Survey of popular robotics simulators, frameworks, and toolkits." In Proceedings of IEEE Southeastcon, Nashville, TN, USA, pp.243–249, March 2011.

[5] Staranowicz, A.; Mariottini, G.L. "A survey and comparison of commercial and open-source robotic simulator software." In Proceedings of the 4th International Conference on PErvasive Technologies Related to Assistive Environments, Heraklion, Crete, Greece, May 2011.

[6] Cook, D.; Vardy, A.; Lewis, R. "A survey of AUV and robot simulators for multi-vehicle operations." In IEEE/OES Autonomous Underwater Vehicles, Oxford, MS, USA, pp. 1–8, October 2014.

[7] Ivaldi, S.; Padois, V.; Nori, F. "Tools for dynamics simulation of robots: a survey based on user feedback," arXiv, 2014, arXiv:1402.7050.

[8] Hentout, A.; Maoudj, A.; Bouzouia, B. "A Survey of Development Frameworks for Robotics" In 8th International Conference on Modelling, Identification and Control (ICMIC-2016), Algiers, Algeria, pp. 67–72, November 2016.

[9] Takaya, K.; Asai, T.; Kroumov, V.; Smarandache, F. "Simulation environment for mobile robots testing using ROS and Gazebo," In 20th International Conference on System Theory, Control and Computing (ICSTCC), Sinaia, Romania, pp. 96–101, October-2016.

[10] Manhães, M.M.M.; Scherer, S.A.; Voss, M.; Douat, L.R.; Rauschenbach, T. "UUV Simulator: A Gazebo-based package for underwater intervention and multi-robot simulation," In OCEANS 2016 MTS/IEEE Monterey, Monterey, CA, USA, pp. 1–8, September-2016.

[11] Portugal, D.; Iocchi, L.; Farinelli, A. "A ROS-Based Framework for Simulation and Benchmarking of Multi-robot Patrolling Algorithms," In *Robot Operating System (ROS) The Complete Reference (Volume 3)*; Kacprzyk, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2019; pp. 3–27; ISBN 978-3-319-91589-0.

[12] Yan, Z.; Fabresse, L.; Laval, J.; Bouraqadi, N. "Building a ROS-Based Testbed for Realistic Multi-Robot Simulation: Taking the Exploration as an Example," Robotics 2017, 6, 21.

[13] Navarro, I.; Zimmermann, F.; Vasic, M.; Martinoli, A. "Distributed graph-based control of convoys of heterogeneous vehicles using curvilinear road coordinates," In 19th International Conference on Intelligent Transportation Systems (ITSC), Rio de Janeiro, Brazil, pp. 879–886, November-2016.

[14] Marjovi, A.; Vasic, M.; Lemaitre, J.; Martinoli, A. "Distributed graph-based convoy control for networked intelligent vehicles," In Intelligent Vehicles Symposium (IV), Seoul, South Korea, pp. 138–143, July-2015.

[15] Soni, A.; Hu, H. "Formation Control for a Fleet of Autonomous Ground Vehicles: A Survey." Robotics 2018, 7, 67.

[16] Chen, X.; Jia, Y. "Adaptive leader-follower formation control of non-holonomic mobile robots using active vision." In IET Control Theory Appl. 2014, 9, pp. 1302–1311.

[17] Li, Z.; Yuan, W.; Chen, Y.; Ke, F.; Chu, X.; Chen, C.L. "Neural-Dynamic Optimization-Based Model Predictive Control for Tracking and Formation of Nonholonomic Multirobot Systems." In IEEE Trans. Neural Netw. Learn. Syst., 2018, pp. 1–10.

[18] White, R.; Tomizuka, M. "Autonomous following lateral control of heavy vehicles using laser scanning radar," In Proceedings of the 2001 American Control Conference. (Cat. No.01CH37148), Arlington, VA, USA, pp. 2333-2338, June-2001.

[19] Lu, G.; Tomizuka, M. "A laser scanning radar based autonomous lateral vehicle following control scheme for automated highways," In Proceedings of the 2003 American Control Conference, Denver, CO, USA, pp. 30–35, June-2003.

[20] Fassbender, D.; Heinrich, B.C.; Luettel, T.; Wuensche, H. "An optimization approach to trajectory generation for autonomous vehicle following," In International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, pp. 3675–3680, September-2017.