Robots Cooperation and Exploration of Objects in Their Environment

Hanafi Hanafi Wissam Wissam, Mohammed Tamali

Robots Cooperation and Exploration of Objects in Their Environment.

WISSAM HANAFI[1*] and TAMALI MOHAMMED[1]

Bechar University, ENERGARID Lab. - SimulIA Team, Po Box 417, Kenadsa street, Bechar, Algeria

***Corresponding author(s). E-mail(s): hanafi.wissam@univ-bechar.dz;**
Contributing authors: tamali.mohammed@univ-bechar.dz;

Abstract

The world has recently witnessed a development that we are experiencing in various fields, especially in the technical field and the findings of modern technologies in the fields of artificial intelligence and robotics programming and its latest digital technologies and cloud computing technology, and it became a trend to use the latest concepts in the world of digitization, such as automation, artificial intelligence techniques, and data analysis, as it became possible for robots to cooperate together in carrying out certain tasks assigned to them while overcoming obstacles in their environment and passing through with ease. Also, the robots are able to work independently of each other, so the loss of one of the robots will not be a big problem because the other robots will complete the task assigned to them without being affected by the loss of this robot. In this paper, three robots have been created that have four wheels, in Webots, and are placed in an environment in Gazebo. Tasks have been assigned to these robots, including adding Lidar and a Kinect camera. The distributed cooperation between robots has been made so that no robot can cross any path that a robot has crossed before. It was given to the three robots the same goal, and the first one to reach the goal tells the rest of the robots the end of the task, and on their way to the goal, they will recognize the objects around them, so the Yolo algorithm (you look only once) was used, which is one of the best algorithms to discover things in the environment in which they are. As for the results obtained in the simulation, our robots carried out all the tasks assigned to them. In our view, these tasks are an ideal model for experimenting with collaboration between robots.

Keywords: Multi-robots, distributed robot system, Path planning, Collaboration, ROS, Gazebo, YOLO.

1. Introduction

The nature of our topic that we touched on is one of the most important topics discussed during the past few years, and it is the cooperation of robots and artificial intelligence, by virtue of which robots are now able to act on their own, according to the commands applied to them, and through cooperative intelligence that makes these robots cooperate with each other to achieve specific goals, as when they are given different tasks, they will complete them to the fullest. Whether central planning is applied, where a single computer can make decisions for a team of robots, or decentralization is applied, where each robot makes its own decisions based on local feedback, or a distributed system, where the computational task is distributed among robots within a team, in the current study, a distributed control approach was used. The robots can move from one point to another so that they don't collide with each other and avoid obstacles in the environment. From here, the definition of a multi-robot system is a group of robots arranged in a multi-agent architecture to perform a common task, and it has received attention in recent years due to the special capabilities offered, such as collaborative behavior, robustness, parallel operation, and scalability [1]. Its importance compared to a single robot system is to perform a set of tasks faster and more powerfully, because even if a particular robot fails to perform a task, it doesn't affect the overall score.

The importance of the research lies in the ability of robots to accomplish the tasks assigned to them, whatever the environment they are in. Among these tasks is the detection of objects in their environment on their way to the goal that was set for them, and as for the path, no robot can cross any path that has been crossed before by another robot.

Therefore, the aim of this study is to create robots that integrate their actions with each other using algorithmic intelligence, regardless of the environment.

As for the research problem, how do robots cooperate to discover the environment and the things in it? What are the communication tools between these robots to exchange information with each other in our case? This will be discussed in detail in this paper.

In this work, a four-wheeled robot was created in Webots, from which the URDF (Unified Robot Description Format) model of this robot was exported to make it move in the Gazebo, where three models of it were repeated, Lidar and the Kinect camera were placed on it, and an environment was created in the Gazebo for these robots to be placed in this environment, and

in turn, the robots explored Environment, where a map in rviz is generated by the SLAM algorithm (Synchronous Localization and Mapping). These robots plan the path using intelligence algorithms when given targets. There are many object detection algorithms that researchers have applied in their work; for us in this work, YOLO (You Only Look Once) was chosen, which is one of the most important object detection algorithms that researchers have used recently. This paper will be organized as follows:

First, highlight the most important advances and recent works in the domain of multi-robots, distributed collaboration utilization, and the Yolo algorithm, afterwards, start by addressing the YOLO algorithm, summarizing and explaining its most important points, importance and principle, then, show how to communicate between robots, after that, explain the choice of case study model, later, show simulation experiments and work phases, then go into more detail about the experiment setup that contains our robot in Webots, the URDF for it, and the robots in the gazebo, then evaluate our case according to the Yolo algorithm, later, implement and present the simulation results with discussion and analysis of the results, finally, conclude our work with a conclusion.

2. Related work

In this section, some important research based on the domain of multi-robots, distributed collaboration utilization, and the Yolo algorithm will be mentioned.

In [3], the authors show that many collaboration problems between multiple robots can be raised and solved within the framework of distributed optimization, such as concurrent localization, planning (SLAM), goal tracking, and task mapping problems. They identified three broad classes of distributed optimization algorithms: distributed first-order methods, sequential convex distributed programming, and the alternating directional multiplier method (ADMM). They also provided representative algorithms within each category. And in the simulation, they used multiple drones that cooperatively tracked a ground vehicle. They compared solutions to this problem using a number of different distributed optimization algorithms. They also implemented a hardware distributed optimization algorithm on networked Rasberry Pis that communicate with XBee modules to demonstrate robustness to real-world networking challenges, and they demonstrated the effectiveness of distributed optimization algorithms.

The authors in [4] presented a distributed optimization algorithm for multi-robot systems, and based on it, the output regularization method was used to solve the coordination optimization problem for general linear dynamical systems. Then they theoretically prove the convergence of the proposed algorithm. Numerical simulations and real-time physical robot experiments were performed to validate the effectiveness of the proposed cooperative control algorithms. For their experimental results to evaluate the effectiveness of the proposed distributed optimization algorithm, numerical case studies for testing the algorithm on multiple robots with linear systems were presented. Then, the proposed algorithm was validated and applied to the real robot Turtlebot, where all source code, distributed algorithm details, and environment setup files are available for their project site, where each robot only communicates with its neighbor robot. They also implemented the algorithm for a group of four agents, then expanded it to a network of ten agents to test the scalability of the proposed algorithm.

In [5], the authors put forward a prototype of a tank military robot with target detection and tracking functions based on computer vision using a library in OpenCV and simulated its turret firing.

The authors in [6] proposed a new Monte Carlo method for estimating relativistic states by outfitting robots with only one UWB transceiver or dynamically integrating simultaneous spatial detections from stereo cameras. They also addressed the challenges of mitigating UWB band errors with a study on LSTM to estimate the range error. They demonstrated that their proposed approach has multiple benefits. They show that a single scale is sufficient to estimate the exact relative states of two robots when their distance measurements are combined. They also show how ROS 2 and Zenoh can integrate to build a scalable wireless communication solution for multi-robot systems and autonomous navigation based on the method of relative positioning. According to experimental results, they demonstrate that their approach is clearly superior to multilateralism for relative state estimation. Moreover, LSTM and external spatial information can improve the performance of the proposed approach using UWB. They have proven their approach and implemented ROS 2 and Zenoh on many popular laptops. They found that their method has low CPU and memory consumption, which allows for scalability.

In [7], the authors focus on first building the basic framework of a neurorobotics platform for an autonomous driving simulator application and benchmark. Including the self-driving

template, they realize the functions of object detection and core character search to build a simulation experience on the NRP-Core platform. They started by creating the physical model of the vehicle and its corresponding environment in an SDF file, then building the "closed loop" autonomous driving based upon PID control along a predetermined path, and finally the "open loop" where objects were detected based on the YoloV5 algorithm. They succeeded in achieving the goal of showing the detected current frame image in the window and running it as a screen camera. The NRP-Core platform showed huge potential for building a multi-agent simulation world.

3. Proposed Algorithm

In this section, our proposed algorithm will be presented:

3.1 YOLO Algorithm: (You only look once)

YOLO is popular because of its speed and accuracy of results, its superior performance over object detection techniques, and the ease of its principle of two stages: the first is searching for the object and identifying the object in a specific image, and the second is determining the exact location of the object within the image. As it uses convolutional neural networks (CNN) to detect objects in real time, the algorithm requires only one forward propagation through a neural network to detect objects. It has been used in many applications to detect people, animals, and many different shapes to be searched for [2]. The YOLO algorithm, like other algorithms, is not without some disadvantages, such as the fact that it is not ideal at detecting very large or very small objects or things that are too far away.

4. Control and connection of the multi-robot system

While the robots communicate with each other, controlling each robot is the main challenge of this system. As the loss of communication leads to the failure of the task, it is necessary to implement continuous communication [8]. In a multi-robot system, the robots cooperate with each other, and each robot must take into account the task requirements and local environmental constraints [9]. The control of multi-robot systems can be categorized as centralized, decentralized, or distributed networks. According to [10], in the centralized multi-robot system, all decisions are made by the master to determine whether a computational task should be performed by a specific robot or by the master. In the decentralized multi-robot

system, the computational task is performed by one or more robots in the team. There is no specified master throughout the mission, which therefore allows all other robots to proceed with their calculations even if one robot fails. In a distributed multi-robot system, the computational task is distributed among the robots present within the team, and this approach was used in this study. The centralized network effectively eliminates conflict between multiple robots, and all robots have real-time communication with the central console. However, this system has a drawback when the main controller (leader) fails to communicate. Conversely, the decision-making of each robot in a decentralized or distributed network is solely based on information from its sensors. That is, each robot only takes into account its own movement, regardless of the movements of other robots [11, 12]. Studies show that the distributed approach is more beneficial than the centralized approach in terms of scalability and practicality. Therefore, the distributed control approach was used in the current study.

5. Choice of a case study model

To create the environment, the Gazebo build editor was used, and then a new model was created with different objects. Fig.1, is the environment created in the gazebo, and Fig.2, is the environment created in Rviz.

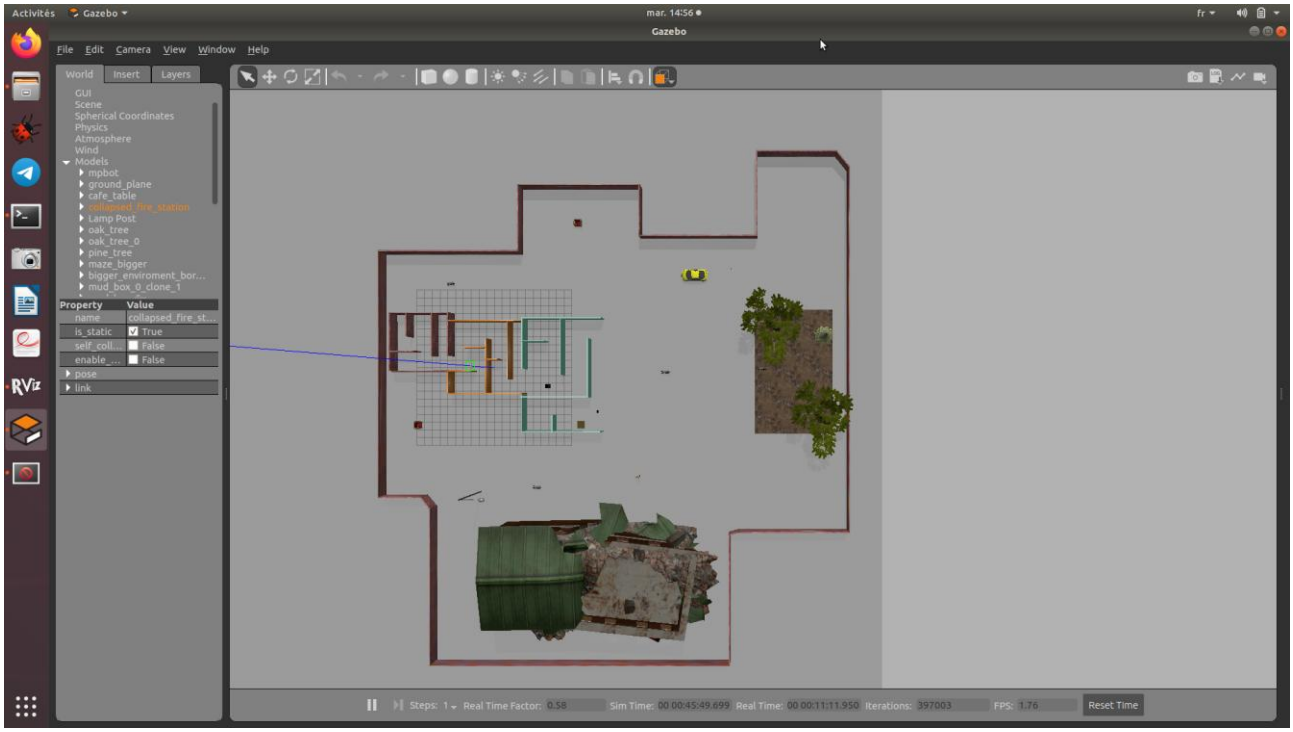

Fig. 1 The whole environment in Gazebo

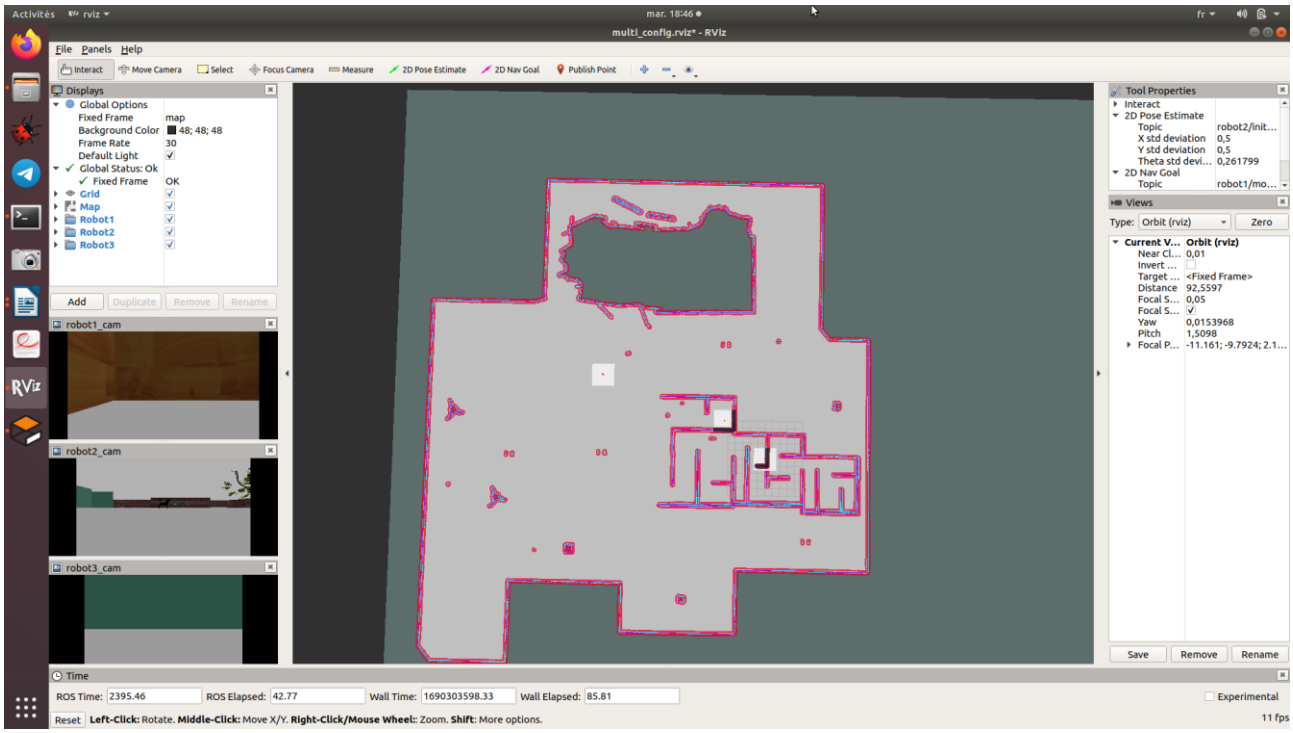

Fig. 2 The whole environment in Rviz

6. Simulation experiments

In this part, the steps of the experimental work done will be listed, mentioning the setup experiment, evaluating the experiment through the gazebo, the simulation used, the way the robots communicate in detail, applying YOLO algorithms to these robots, implementing that in various environments, and displaying the results obtained.

6.1 Work phases and steps

Here the phases of the work will be mentioned, which are :

1 : The design of a four-wheeled robot in the Webots software ;

2 : The exporting of the URDF model of this robot ;

3 : Making the robot move in the gazebo ;

4 : The construction of the three models of this robot in order to be able to simulate it in the Gazebo simulator ;

5 : The creation of environments in the gazebo to try to test and run these robots in ;

6 : The application of navigation, where robots have to map out the environment in Rviz and are present in it ;

7 : Using a distributed approach to communicate between robots ;

8 : The implementation of artificial intelligence algorithms for these robots.

The following task steps are followed and implemented:

For send the three robots to the same goal, and when the first robot arrives at the goal, it cancels the other robot's mission.

For prevent any robot from crossing over the visited area, when sending the goal to the robots, when the first robot crosses the path, the second robot detects this and returns to the initial pose.

For detect the objects, the robots will be equipped with cameras and use the YOLO algorithm.

6.2 Experiments Setup

6.2.1 Robots in Webots

The intelligent algorithms are implemented on three robots created in Webots [13], using the Robot Operating System (ROS) [14]. Webots is used to create a robot with four wheels. Fig.3 and Fig.4, show the generated robot. The robot is a four-wheeled robot on which the following sensors are placed:

Lidar : Light detection and ranging, or laser imaging detection and ranging, is a remote measurement technique based on the analysis of the properties of a beam of light returned to its emitter.

The Kinect camera : Is a motion-sensing device with a camera that uses proprietary technology to track and translate body movements.

IMU: (Inertial Measurement Unit) Measurement and determination of the forces acting on the body and its angular acceleration.

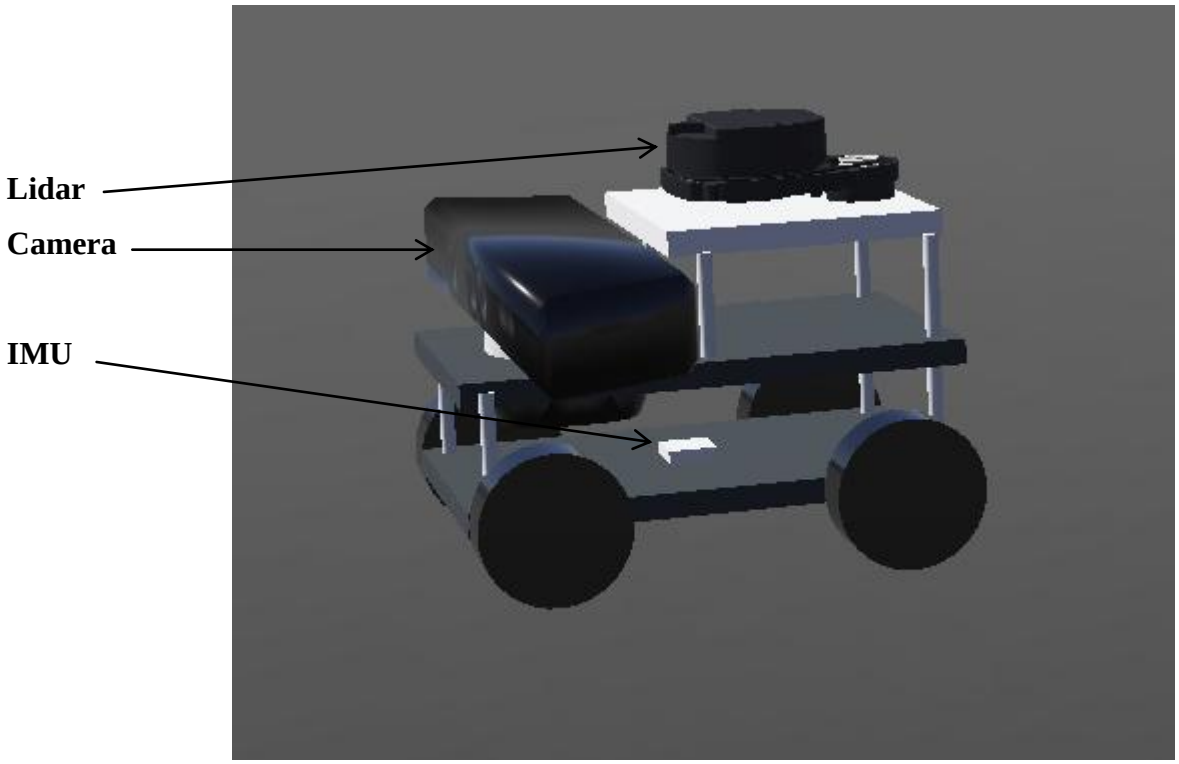

Fig. 3 Four-wheeled robot

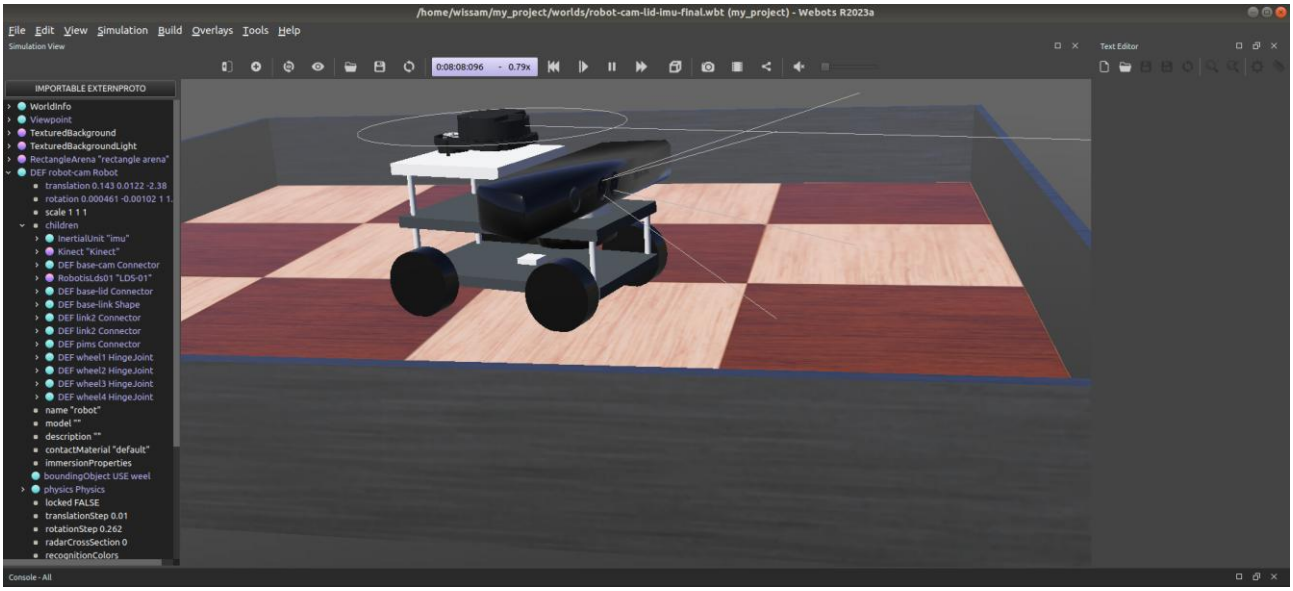

Fig. 4 Four-wheeled robot with lidar and camera rays in Webots.

6.2.2 URDF of the robot

Every robot consists of links that are connected with joints. The URDF (Unified Robot Description Format) [15] is our robot chassis. As it accurately describes all the parts of the robot and even the sensors added to it, in our case the imu, lidar, camera, and encoder on wheels are added. There is a file related to Gazebo; it must add a plugin in Gazebo and add the link and the joint of the camera in the URDF, and it must be connected to the robot at the basis, that is, the base_link, and the same thing for lidar and the IMU; for the encoder on wheels, it is with a plugin that belongs to the robot. There are three ways to create the URDF file: either build a robot on Gazebo, type it manually, or export it from Webots; the last one was chosen. Fig.5 shows that :

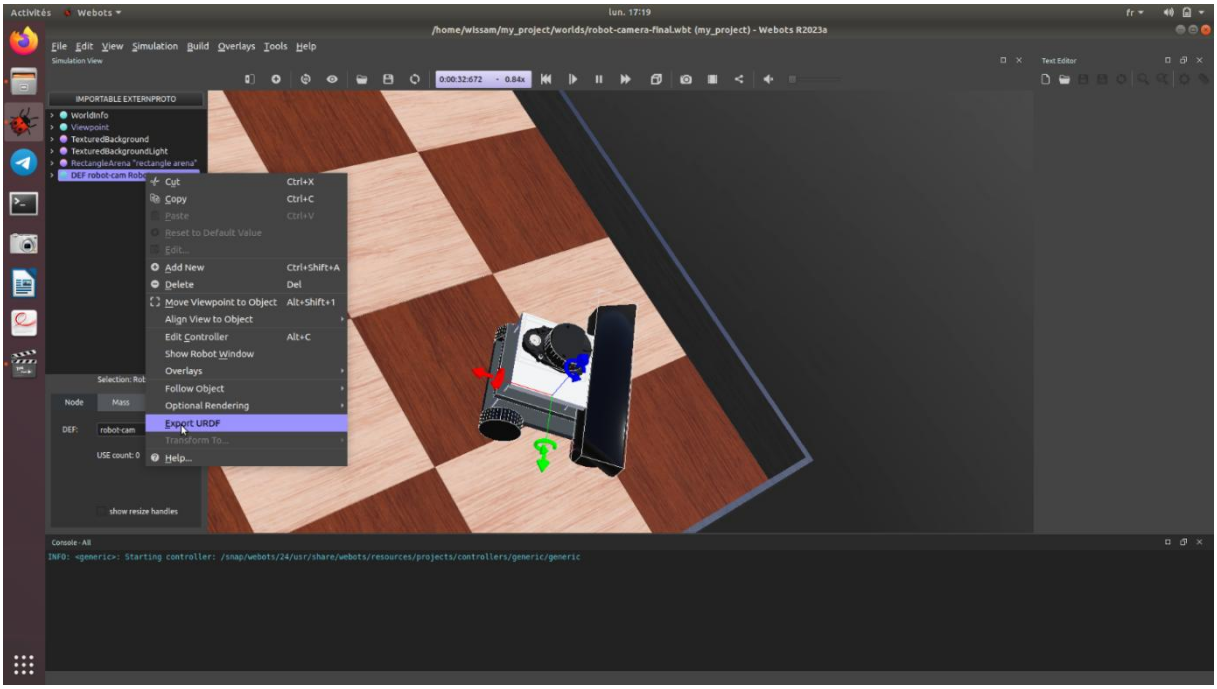

Fig. 5 Exporting URDF from Webots

Fig.6 shows the parts of our robot.

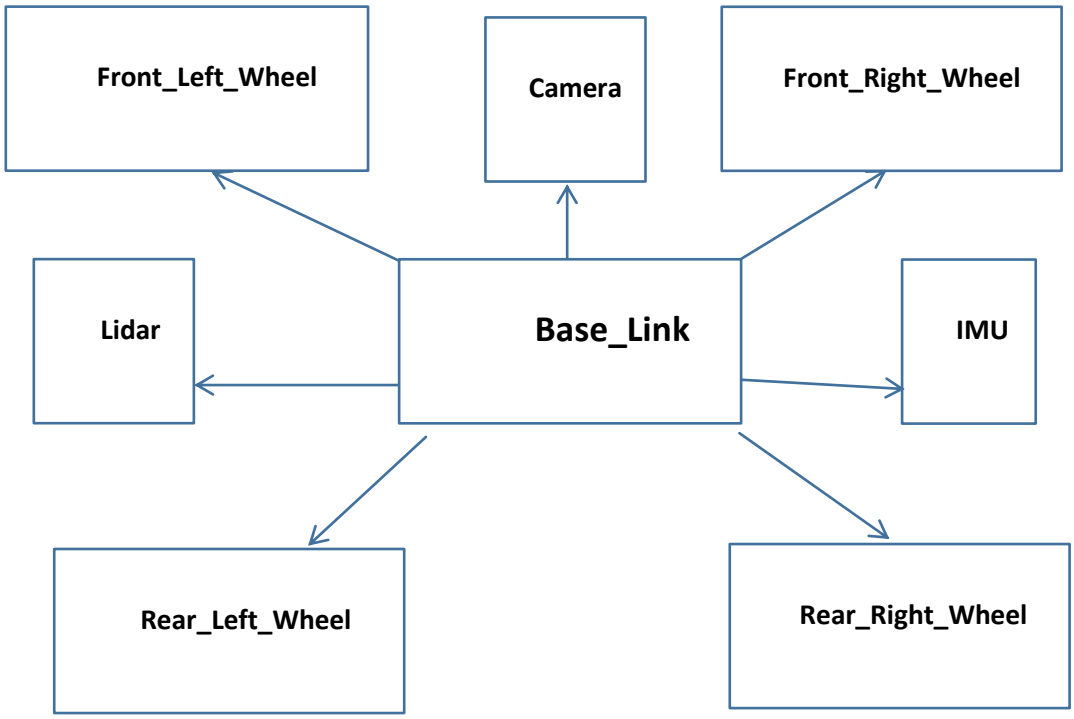

Fig. 6 Robot components

6.2.3 Robots in Gazebo

For our robot, to run it in Gazebo, four Gazebo plug-ins were used: IMU, laser, camera, and differential motor [16]. To start the robot in Gazebo, a folder must be created to run in src, and our files to run are placed in it. In Fig.7, below, it will see the robot in the gazebo after creating its URDF file.

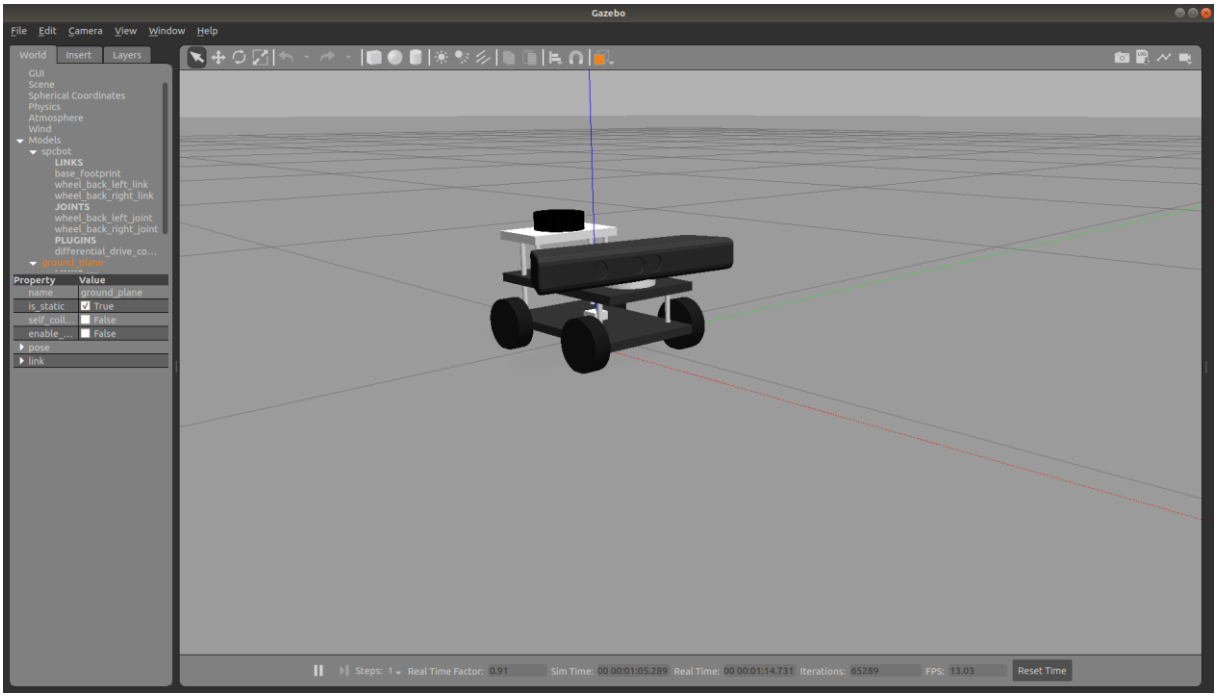

Fig. 7 Four-wheeled robot in Gazebo

In order to create an environment, the algorithm of gmapping will be used, which is a package. The map will be drawn by teleop from the launch file, which uses the slam_gmapping algorithm to build 2D maps [21]. After each robot creates a local map, all local maps are combined into a global map. In order to enable the robot to move towards the goal point while safely avoiding obstacles between the robots, the ROS move_base package was used [17]. This package implements the A-star global path planning algorithm and the Dynamic Window Approach (DWA) algorithm to achieve dynamic local obstacle avoidance.

The global planner draws the path according to the map and avoids static obstacles, whereas the local planner draws the path based on the global path while avoiding dynamic obstacles; hence, the global planner builds a map of the environment while the local planner only works with the information it gets from the sensors and plans the path. In fact, the A-star algorithm only creates the path (the global planner), while the DWE algorithm avoids new obstacles (the local planner). In Ros navigation, the two are used together, so it will be observed that a star avoids obstacles, and it is basically a second algorithm that works with a star whose job it is to avoid new obstacles. In rviz, the position and direction are indicated with an arrow to set the robot's orientation. The goal can be given by rviz; there is something called nav to goal, and by using this tool in rviz in ros, it can be given for the robot to reach a goal. So the robot will be given a specific location, and it will discover the environment in which it is located first. Navigation will be done, meaning a map will be created, and then its location and place will

be determined on this map. When given a goal, the robot will be able to reach it. In order for the three robots to navigate and simulate in the environment, a package containing the launch files for simulating the robots will be written first, and one map_server must be launched for all the robots, and one amcl_package and one move_base for each robot, where map_server provides map data and allows us to save created maps [18]. AMCL is a probabilistic localization system for a moving robot that uses a particle filter to track the robot's position against a known map. Because every robot is in a different place, the three robots need different topics for control, sensors, speed, and odometry [19]. The move_base node links together a global and local planner to accomplish its global move navigation and also maintains two costmaps, one for the global and one for the local planner, that are used to accomplish the navigation tasks [17]. And therefore:

Three move_base files are created for path planning.

Three AMCL files are created for localize the robot on the map.

The A-star algorithm was used because it is faster and is found mainly in the navigation package. Just make a change in the parameter (a star=true).

6.3 Evaluation of the case according to the YOLO algorithm

In this work, the YOLO (you only look once) algorithm is implemented on the three robots for detecting several objects; Fig.8 shows that.

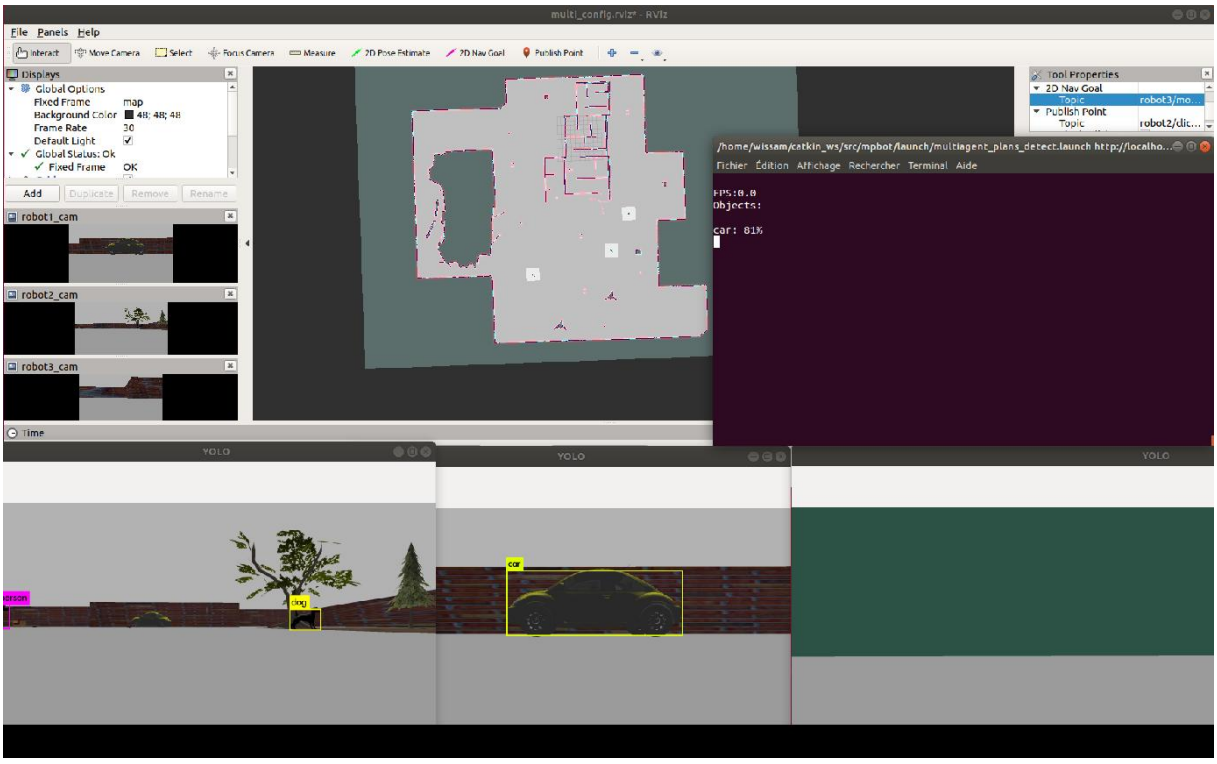

Fig. 8 Application of YOLO to the three robots

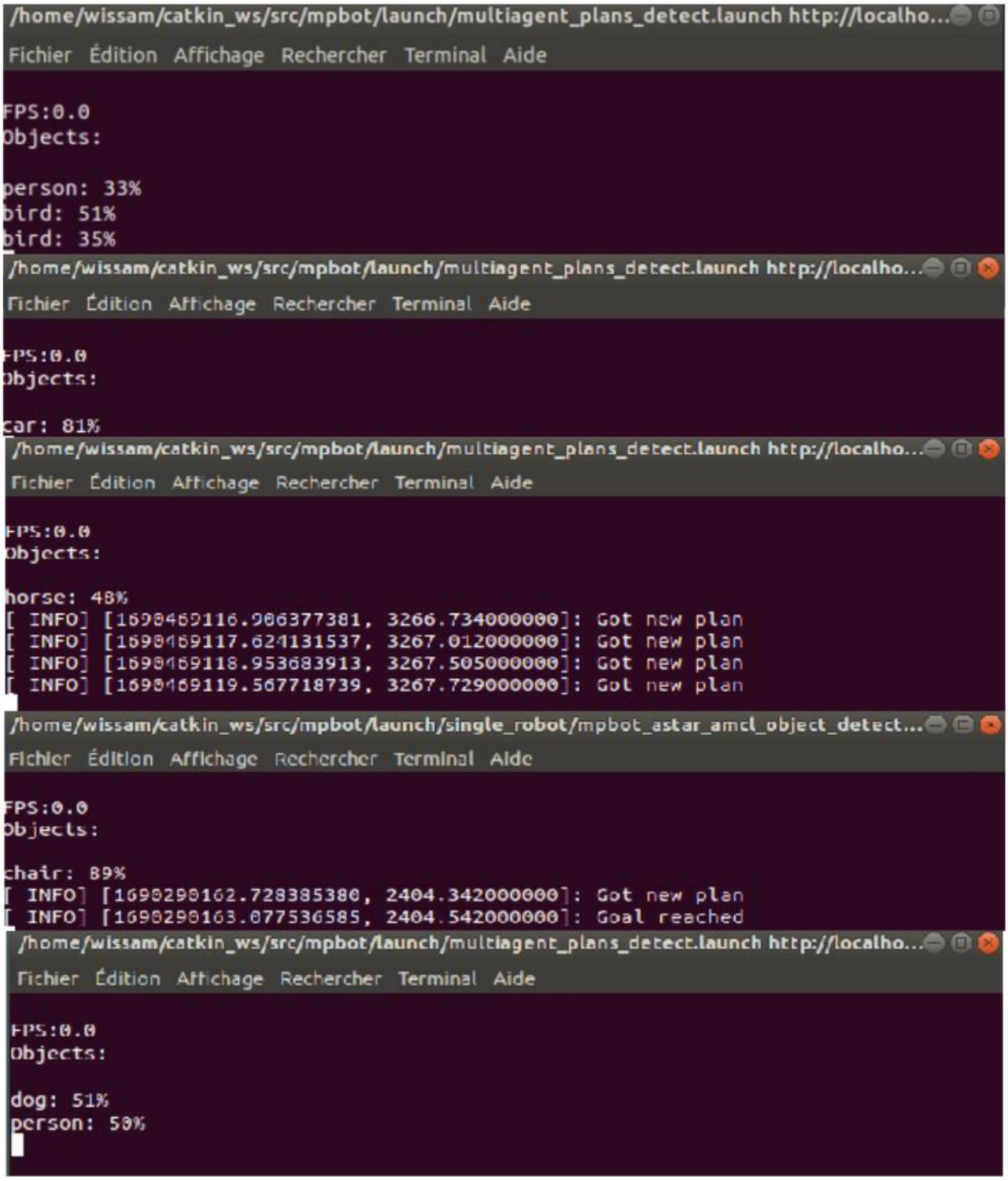

Fig. 9 The objects detected by the algorithm YOLO

Yolo give an approximation of similarity to the objects in their classes. In Fig.9, several objects are detected by the algorithm YOLO, which are: a chair, a car, a person, a bird, a dog, and a horse. The basic files in Darknet_Ros are modified. The names of all topics for the three robots are changed since each robot file will repeat three times.

6.4 Implementation

ROS Melodic is installed on Ubuntu 18.04. In our case, the Gazebo was used, which is one of the main tools used in the ROS community. It is a complete simulation, even in complex environments. Gazebo also has a powerful library of sensor models, including laser, inertial measurement unit (IMU), and camera, which in our case will be used in robotics, allowing algorithms to be quickly tested [20]. The following diagram, shown in Fig. 10, explains a summary of the robot's path from giving the order until reaching the goal.

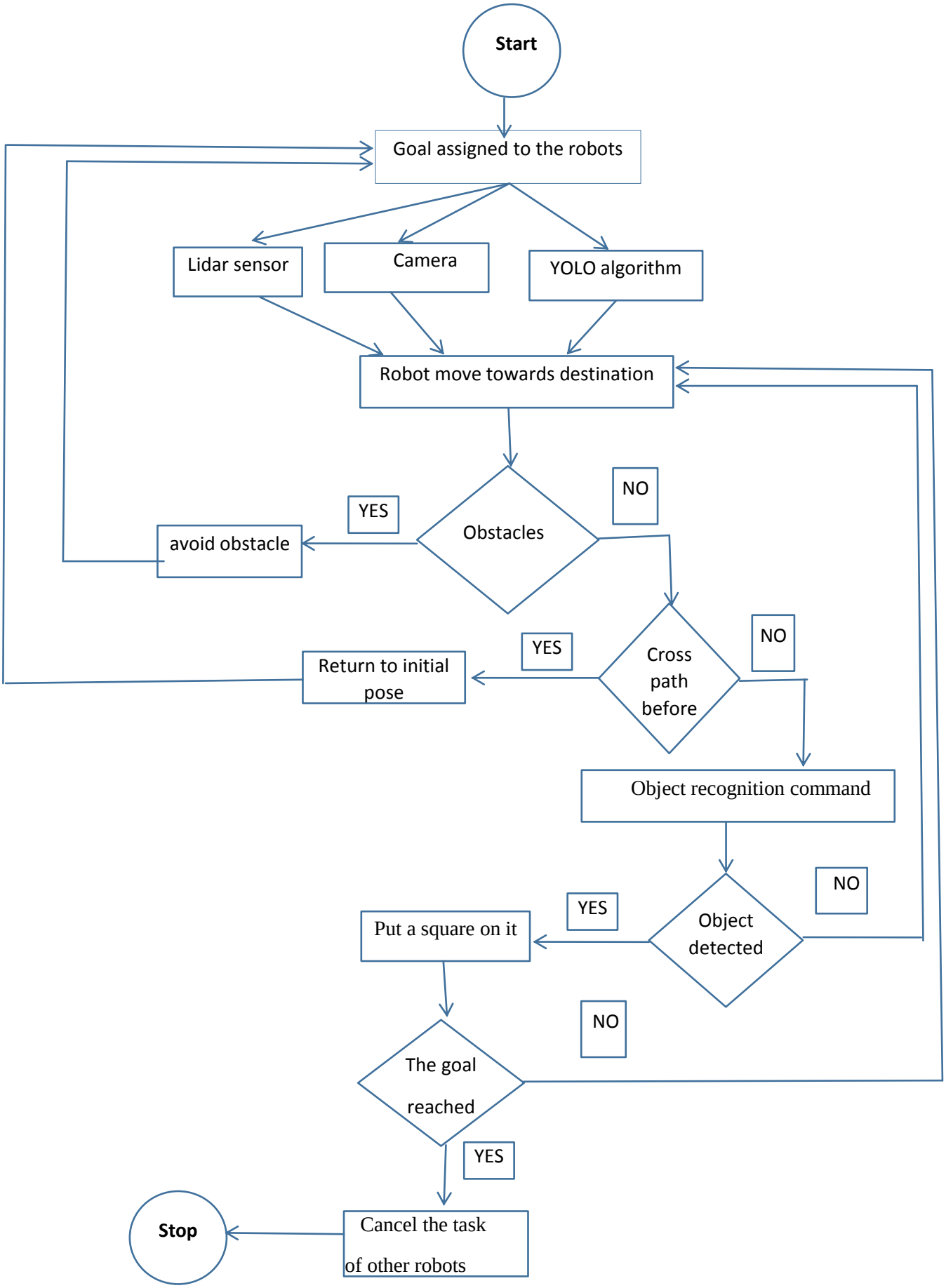

Fig. 10 Flowchart for Path Planning

6.5 Discussion and Analysis of Simulation Results

To implement the distributed cooperation between the robots, two tasks are performed: the first is to send the three robots to the same goal, and the second is to prevent any robot from crossing a path it has passed before. For the first task of sending the three robots to the same goal, the distributed collaboration is used so that when the first robot arrives at the goal, it cancels the other robot's mission, so the two other robots stop (velocities are zero).

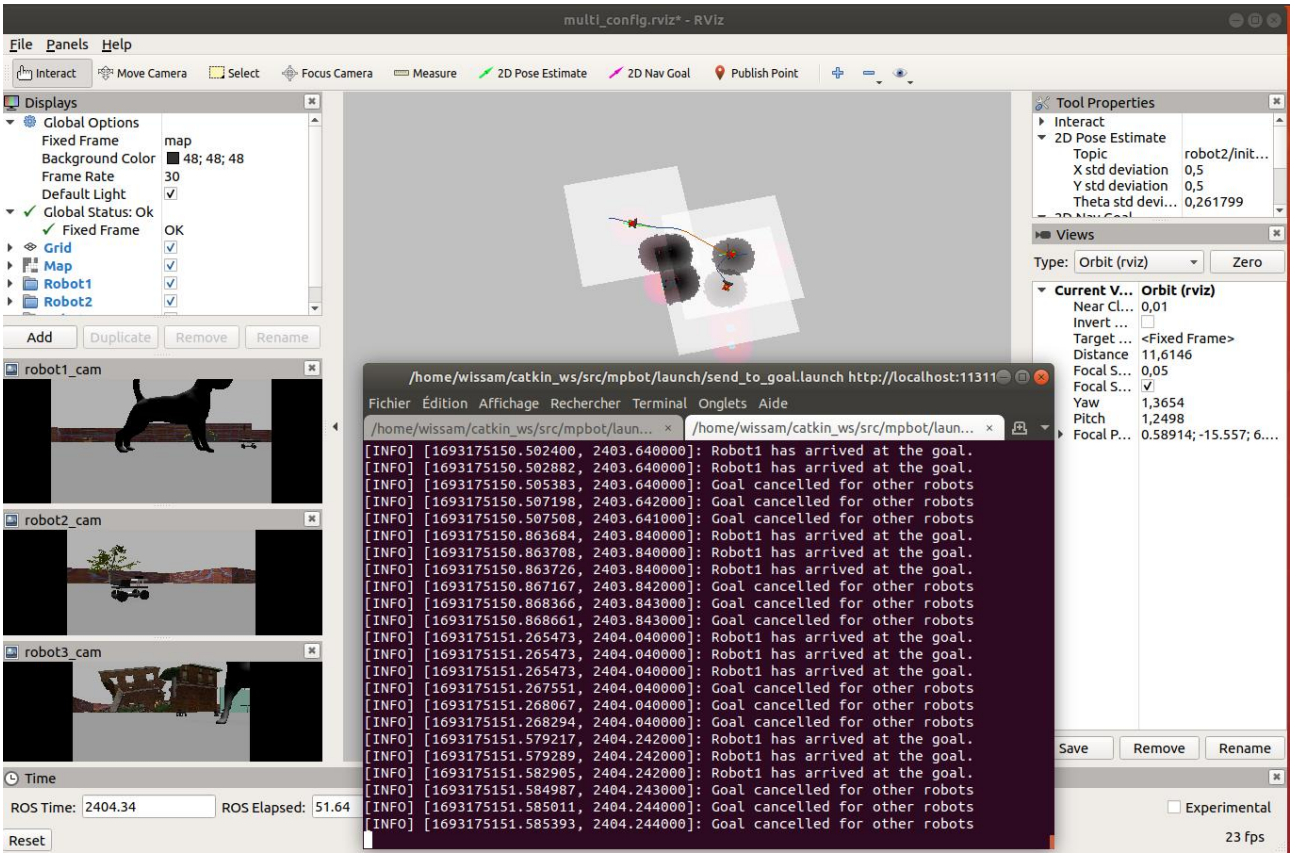

Fig. 11 Sending the three robots to the same goal

Fig. 11 shows that both robots2 and robot3 were prevented from crossing the path by robot1, which in turn reached the goal when sending the three robots to the same goal. For the second mission of preventing any robot from crossing over a visited area, when sending the goal to the robots, and when the first robot crosses the path, the second robot detects this and returns to the initial pose.

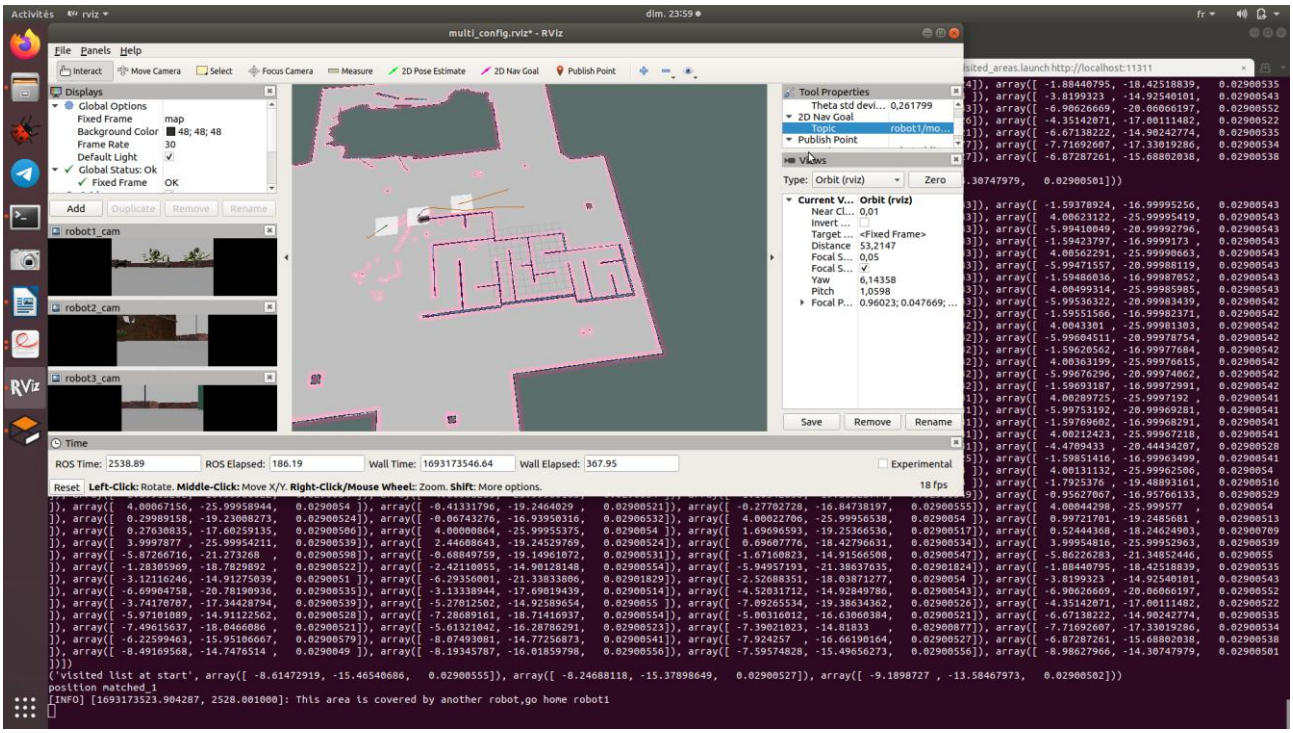

Fig. 12a Preventing robot1 from crossing over the visited area

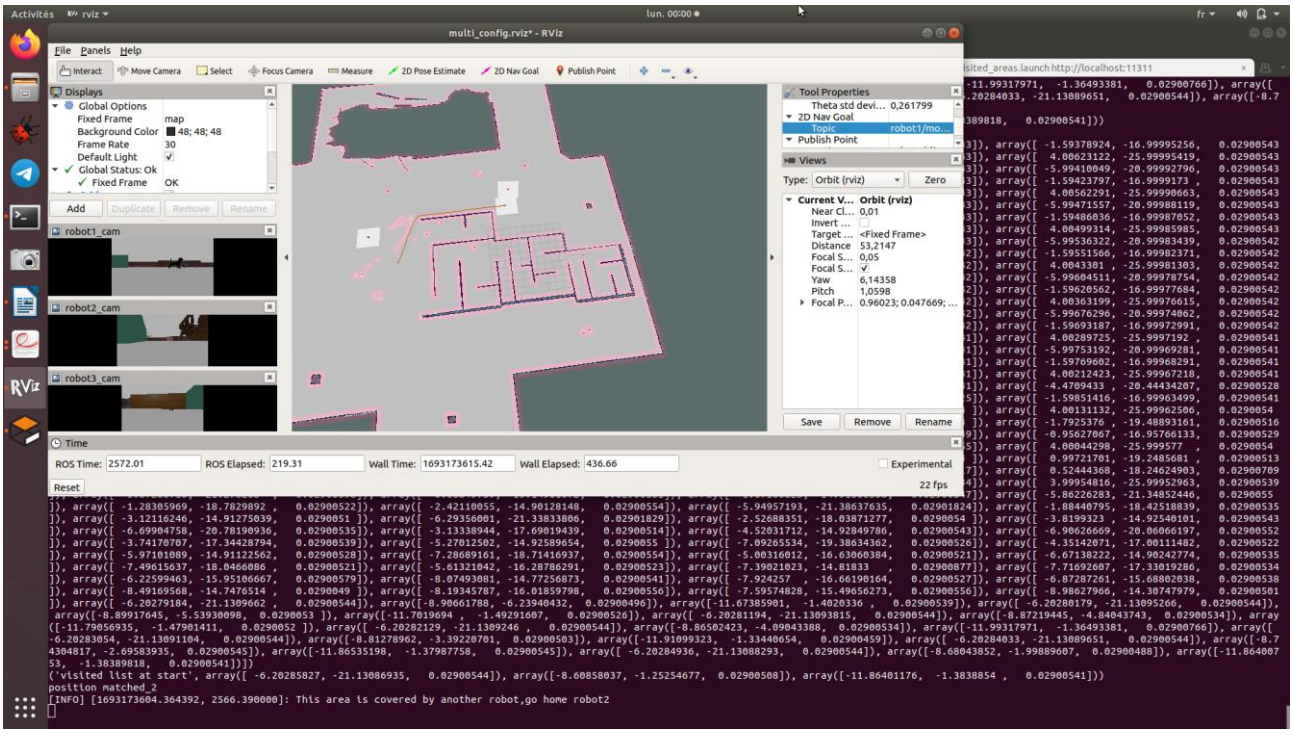

Fig. 12b Preventing robot2 from crossing over the visited area

(Fig. 12a) and (Fig. 12b) show the cancellation of other robot missions and their return to the starting point once robot1 reaches the goal.

7. Conclusion

In this paper, a four-wheeled robot is designed in Webots and exported as an URDF model for this robot in order to be able to simulate it in the environments that we created in the gazebo simulator. Then, three models of this robot were constructed, and distributed collaboration was used as a way of communicating between them. Then a navigation system was implemented, with the robots mapping the environment to the Rviz while they were on it, and imu, lidar, a Kinect camera, and an encoder on wheels were added to these robots. Also, the robots are assigned tasks, including that no robot should cross an environment that has been traversed by a robot before it. The same goal was presented to the three robots, and when the first robot reaches the goal, it tells the rest of the robots that it has reached the goal; from here, the task ends. And on their way to the goal, they will recognize the objects around them, so to implement this, we used the YOLO (You Only Look Once) algorithm to detect the objects existing in the environment.

According to the results obtained, our robots have performed all the tasks assigned to them.

In the future, it is planned to apply what was done in simulations to real robots.

Declarations

Acknowledgments

I would like to express my gratitude to Tamali Mohammed for their collaboration.

Funding

The research was conducted independently, with no external funding or financial support.

Conflicts of interests

Authors declare that they have no conflicts of interest in this work.

Authors' contributions

All authors contributed equally to this research and paper. They collaborated closely on all aspects of the study, including the research design, data collection and analysis, interpretation of results, and the preparation of the manuscript. All authors have read and approved the final version of the paper.

References

[1] Ying Wang and Clarence W. de Silva. "A machine-learning approach to multi-robot coordination". https://doi.org/10.1016/j.engappai.2007.05.006. Elsevier. Engineering Applications of Artificial Intelligence. Volume 21, Issue 3, April 2008, Pages 470-484.

[2] https://arxiv.org/pdf/1506.02640.pdf.

[3] Ola Shorinwa, Trevor Halsted, Javier Yu, Mac Schwager. " Distributed Optimization Methods for Multi-Robot Systems: Part I — A Tutorial ". arXiv preprint arXiv:2301.11313. 2023.

[4] Yi Dong , Zhongguo Li , Xingyu Zhao , Zhengtao Ding , Xiaowei Huang. "Decentralised and Cooperative Control of Multi-Robot Systems through Distributed Optimisation". In Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023, IFAAMAS, 9 pages.

[5] W. Budiharto, V. Andreas, J. S. Suroso, A. A. S. Gunawan, and E. Irwansyah, ‘‘Development of tank-based military robot and object tracker,’’ in Proc. 4th Asia–Pacific Conf. Intell. Robot Syst. (ACIRS), Jul. 2019, pp. 221–224.

[6] Xianjia Yu , Paola Torrico Morón , Sahar Salimpour , Jorge Peña Queralta , Tomi Westerlund. " Loosely Coupled Odometry, UWB Ranging, and Cooperative Spatial Detection for Relative Monte-Carlo Multi-Robot Localization". arXiv:2304.06264v1 [cs.RO] 13 Apr 2023.

[7] Wei Cao, " Autonomous Driving Simulator and Benchmark on Neurorobotics Platform". Chair of Robotics, Artificial Intelligence and Real-Time Systems. TUM School of Computation, Information and Technology. Technical University of Munich. 2022.

[8] E. F. Flushing, L. M. Gambardella, G. A. D. Caro, Simultaneous task allocation, data routing, and transmission scheduling in mobile multi-robot teams, in: IEEE International Conference on Robotics & Automation, 2017, pp. 1861–1868.

[9] U. Jain, R. Tiwari, W. W. Godfrey, Comparative study of frontier based exploration methods, in: 2017 Conference on Information and Communication Technology (CICT’17), 2017, pp. 1–5.

[10] Herdawatie, Abdul Kadir. 'Multi Robot System (Part 1: Introduction)'. Urrg.eng.usm.my. N.p., 2015. Web. 26 Apr. 2015.

[11] H. Min, F. Sun, F. Niu, Decentralized uav formation tracking flight control using gyroscopic force, in: IEEE International Conference on Computational Intelligence for Measurement Systems and Applications, 2009, pp.91–96.

[12] B. Ranjbar-Sahraei, F. Shabaninia, A. Nemati, S.-D. Stan, A novel robust decentralized adaptive fuzzy control for swarm formation of multiagent systems, IEEE Transactions on Industrial Electronics 59 (8) (2012) 3124–3134.

[13] webots, https://cyberbotics.com/

[14] Ros, https://www.ros.org.

[15] urdf , http://wiki.ros.org/urdf/Tutorials

[16] gazebo plugin, https://classic.gazebosim.org/tutorials?tut=ros_gzplugins

[17] Move base package summary, http://wiki.ros.org/move_base

[18] map_server, http://wiki.ros.org/map_server

[19] amcl, http://wiki.ros.org/amcl

[20] Gazebo simulator, http://gazebosim.org.

[21] Ros gmapping, http://wiki.ros.org/gmapping.