2022 International Conference of Science and Information Technology in Smart Administration (ICSINTESA) | 978-1-6654-7288-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICSINTESA56431.2022.10041600

Webots Simulator for Lyapunov-based Cooperative Omnidirectional Mobile Robots Evaluation

1st Fransisco Jordan
Department of Electrical Engineering
University of Surabaya
Surabaya, Indonesia

2nd,* Hendi Wicaksono
Department of Electrical Engineering
University of Surabaya
Surabaya, Indonesia
hendi@staff.ubaya.ac.id, 0000-0001-8324-7911

3rd Veronica Indrawati
Department of Electrical Engineering
University of Surabaya
Surabaya, Indonesia
veronica@staff.ubaya.ac.id

***Abstract*—A multi-robot system is a set of robots that share a common objective and collaborate to achieve it. Multi-robot systems can address distributed and complicated real-world issues more effectively in various industries, including logistics, transportation, and industrial manufacturing. The higher system performance because of the collaborative efforts of numerous robots provides a substantial possible benefit of using a multi-robot system rather than a single robot. The Lyapunov control approach is one advanced approach to coordinating multiple robots in warehouse logistics applications. The method has been successfully simulated on point masses. However, to be physically implemented in robotic devices still requires several processes that are not simple. The case study focuses on finding the possibility of a distributed cooperative mobile robot using a LiDAR sensor to recognize the 'friend' robot and the obstacle. This paper shows that each robot can recognize well between its neighbor or obstacles and maintain its formation of 1 m during the trip with obstacle presence. The simulations were performed in Webots to verify the proposed algorithms. The mathematical analysis and the experiment prove the system's stability by seeing the robots' velocity.**

***Index Terms*—Cooperative robots, omnidirectional mobile robots, Lyapunov control, Webots simulator, LiDAR sensor**

I. Introduction

A multi-robot system is a set of robots that share a common objective and collaborate to achieve it, boosting the entire system's utility. Multi-robot systems can address distributed and complicated real-world issues more effectively in a range of industries, including logistics, transportation, industrial manufacturing, and disaster relief [1]. The higher system performance (e.g., when it comes to job completion time) because of the collaborative efforts of numerous (homogeneous or heterogeneous) robots with standard and/or distinct skills provides a substantial possible benefit of using a multi-robot system rather than a single robot [2].

An overview of the multi-robot cooperation approach is given in [3]. When creating a coordinating mechanism, three major aspects must be addressed: planning techniques, communication mechanisms, and decision-making systems [3]. Various solutions have been reported in the literature. Some of the approaches studied include cell decomposition, potential fields, roadmaps (Voronoi diagrams, sampling-based methods), communication-based coordination (e.g., hierarchical coordination), self-organizing coordination (e.g., ant colony optimization (ACO), particle swarm optimization (PSO)), and so on. There are multi-robot planning approaches for adaptive adjustment (e.g., reinforcement learning) [4]. Several advanced approaches to coordinating multiple robots in warehouse logistics applications are discussed in [5].

Assessing the effectiveness of multi-robot system coordination mechanisms in computer simulations is more convenient than testing their efficiency in real-life settings. As a simulation, this paper gives a case study on the collaboration of multi-robot systems. Therefore, it is necessary to verify the functionality of the mechanisms and/or mathematical algorithms developed for this system. The Webots, a robot simulation system, is well suited for this type of testing [6]. Webots is an advanced mobile robot simulation environment that allows you to construct virtual 3D environments with physical qualities like mass, joints, and coefficient of friction [6].

This work proposes a simulation model using the Webots robot simulator to realize a Lyapunov control for a cooperative mobile robot before hardware realization in industrial environments.

II. System Description

A. The Omnidirectional Mobile Robot

A three-wheel drive (3WD) mobile robot consisting of three Omni-wheels is presented in Fig. 1. The axes are at a 120-degree angle. The Omni-wheeled mobile robot can move in any direction and angle without a steering element. Some driving directions of the 3WD mobile robot are shown in Fig. 2.

Kinematic Model. To determine the robot's motion, the coordinate vector of the robot is defined as $\mathbf{q} = [q_1\ q_2\ q_3]^T$, and the vector of velocity on the world's axis is the derivative of $\mathbf{q}$. By the following Eq. 1, the velocity vector mentioned above can be transformed into a velocity on the robot's axis. Fig. 3 shows the coordinate frame of the Omni-wheel mobile robot.

$$\begin{bmatrix}\dot{q}_1\\ \dot{q}_2\\ \dot{q}_3\end{bmatrix} = \begin{bmatrix}\mathrm{c}(\theta_1+\varphi) & \mathrm{c}(\theta_2+\varphi) & \mathrm{c}(\theta_3+\varphi)\\ \mathrm{s}(\theta_1+\varphi) & \mathrm{s}(\theta_2+\varphi) & \mathrm{s}(\theta_3+\varphi)\\ 1 & 1 & 1\end{bmatrix}\begin{bmatrix}v_1\\ v_2\\ v_3\end{bmatrix}$$
$$\mathrm{s}(\bullet) = \sin(\bullet),\ \mathrm{c}(\bullet) = \cos(\bullet) \quad (1)$$

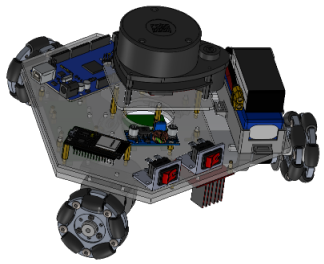

Fig. 1. The 3WD Omni-wheel robot design

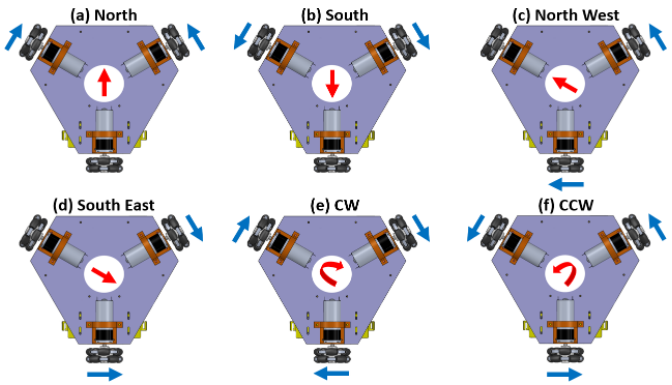

Fig. 2. Driving directions of 3WD Omni-wheel

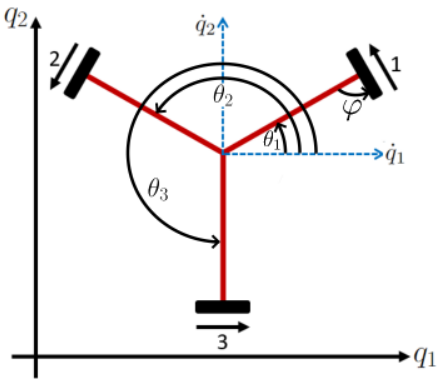

Fig. 3. The coordinates of the Omni-wheeled mobile robot system

where v_1, v_2, and v_3 are velocities of the first-, second-, and third wheel, respectively. The values of $\varphi, \theta_1, \theta_2$, and θ_3 are 90^0, 30^0, 150^0, and 270^0, respectively.

B. Lyapunov Function Components

Consider cooperative of $n \in \mathbb{N}$ robots as point masses. Let the position of the ith robot be $\mathbf{q}_i = (q_{1i}, q_{2i})$, for all $i \in \{1, 2, ..., n\}$, with $\mathbf{q}(0) = (q_{1i}(0), q_{2i}(0))$ as the initial position. Let the center of the cooperative robots be defined as

$$\mathbf{q}_c = \left(\frac{1}{n} \sum_{k=1}^{n} q_{1k}, \frac{1}{n} \sum_{k=1}^{n} q_{2k} \right) \tag{2}$$

Then, $(v_i, w_i) = (\dot{q}_{1i}, \dot{q}_{2i})$ be the instantaneous velocity of the ith robot. Consider $\mathbf{q}_i = (q_{1i}, q_{2i}) \in \mathbb{R}^2$, and $\mathbf{q} = (\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_n) \in \mathbb{R}^{2n}$ be the state vectors. If the velocity (v_i, w_i) has a state feedback law of the form, $(v_i, w_i) = (-\mu_i f_i(\mathbf{q}), -\varphi_i g_i(\mathbf{q}))$, for $\mu_i, \varphi_i > 0$ and functions $f_i(\mathbf{q})$ and $g_i(\mathbf{q})$, to be designed appropriately later. If $\mathbf{g}_i(\mathbf{q}) = (-\mu_i f_i(\mathbf{q}), -\varphi_i g_i(\mathbf{q})) \in \mathbb{R}^2$ and $\mathbf{G}(\mathbf{q}) = (\mathbf{g}_1(\mathbf{q}), \mathbf{g}_2(\mathbf{q}), ..., \mathbf{g}_n(\mathbf{q})) \in \mathbb{R}^{2n}$, then the cooperative of n robots is

$$\dot{\mathbf{q}} = \mathbf{G}(\mathbf{q}), \mathbf{q}_{t=0} = \mathbf{q}(0) \tag{3}$$

If the system has an equilibrium point and is denoted by $\mathbf{q}_e = (\mathbf{q}_{e1}, \mathbf{q}_{e2}, ..., \mathbf{q}_{en}) \in \mathbb{R}^{2n}$. The stability of $\mathbf{q}_e$ will be analyzed by the Direct Method of Lyapunov.

Attraction to the Center of Formation. The attraction function is a component of the system's Lyapunov function based on the concept that the cooperative mobile robot must retain formation for effective material transport. The attraction function is the distance measurement between a robot and the center of the formation, defined as

$$R_i(\mathbf{q}) = \frac{1}{2} \left[(q_{1i} - q_{1c})^2 + (q_{2i} - q_{2c})^2 \right] \tag{4}$$

The Target of the Formation of $n \in \mathbb{N}$ Robots. The target of formation is a disk with center (a, b) and radius r_τ. The target attraction function measuring the distance between the center of the system and the target will be included as components of the Lyapunov function for the system as follows

$$T(\mathbf{q}) = \frac{1}{2} \left[(q_{1c} - a)^2 + (q_{2c} - b)^2 \right] \tag{5}$$

Inter-Robots Collision Avoidance. For short-range repulsion between the ith and jth robot, $j \neq i, i, j \in \{1, 2, ..., n\}$, the function described as

$$Q_{ij}(\mathbf{q}) = \frac{1}{2} \left[(q_{1i} - q_{1j})^2 + (q_{2i} - q_{2j})^2 - (2r_a)^2 \right] \tag{6}$$

where r_a is the radius of the disk in which the robot is residing.

Static Obstacles Avoidance. Consider the configuration space of the system (3) clustered with $m \in \mathbb{N}$ static obstacles. For the purpose of avoiding possible collision with the kth static obstacle where $i \in \{1, 2, ..., n\}$ and $k \in \{1, 2, ..., m\}$, the obstacle avoidance function for the ith robot added.

$$W_{ik}(\mathbf{q}) = \frac{1}{2} \left[(q_{1i} - o_{k1})^2 + (q_{2i} - o_{k2})^2 - (r_{ok} + r_a)^2 \right] \tag{7}$$

where r_{ok} is the radius of the obstacle.

C. A Tentative Lyapunov Function

Let there be constants $\alpha > 0, \gamma_i > 0, \beta_{ij} > 0$ and $\lambda_{ik} > 0$, and define, for $i, j \in \{1, 2, ..., n\}$ a tentative Lyapunov function for system (3),

$$L(\mathbf{q}) = \alpha T(\mathbf{q}) + \sum_{i=1}^{n} T(\mathbf{q}) \left(\gamma_i R_i(\mathbf{q}) + \sum_{j=1, j \neq i}^{n} \frac{\beta_{ij}}{Q_{ij}(\mathbf{q})} + \sum_{k=1}^{m} \frac{\lambda_{ik}}{W_{ik}(\mathbf{q})} \right) \tag{8}$$

The time-derivative of the Lyapunov function along the trajectories of the system (3) is

$$\dot{L}(\mathbf{q}) = \sum_{i=1}^{n} \left[f_l(\mathbf{q}) v_i + g_l(\mathbf{q}) w_i \right] \tag{9}$$

Due to page limitation, the details of $f_l(\mathbf{q})$ and $g_l(\mathbf{q})$ can be found in [7].

D. Velocity Controllers

Let there be constants $\mu_i > 0$ and $\varphi > 0$ and define the instantaneous velocity components of the system (3) as $v_i = -\mu_i f_l(\mathbf{q}), w_i = -\varphi_i g_l(\mathbf{q})$. Then

$$\begin{aligned} \dot{L}(\mathbf{q}) &= -\sum_{i=1}^{n} \left[\mu_i \left(f_l(\mathbf{q}) \right)^2 + \varphi_i \left(g_l(\mathbf{q}) \right)^2 \right] \\ &= -\sum_{i=1}^{n} \left[\frac{v_i^2}{\mu_i} + \frac{w_i^2}{\varphi_i} \right] \leq 0 \end{aligned} \tag{10}$$

Stability Analysis. At the target, where $(q_{1c}, q_{2c}) = (a, b)$, the velocities, v_i and w_i, are zero because $f_l = 0$ and $g_l = 0$. As a result, the individuals maintain a consistent arrangement around the goal. As a result, their fixed locations constitute components of an equilibrium $\mathbf{q}_e$ of system (3). It is easy to see that $L(\mathbf{q}_e) = 0, L(\mathbf{q}) > 0 \; \forall \mathbf{q} \neq \mathbf{q}_e$, and $\dot{L}(\mathbf{q}) \leq 0$. Hence, it can be concluded that (8) is the Lyapunov function that guarantees the stability of the system (3).

III. Webots Simulation Environment

Webots is an open-source robotics simulator developed at EPFL, Switzerland. This platform is widely used in industry, education, and research. It simulates the dynamics of rigid bodies and their collisions using the open dynamics engine (ODE) [8].

Webots offers a variety of stylish robot models and helps developers create simulation environments and model their robots to perform simulations in specific scenarios. The new model's design must consider geometric properties (shape, size, position, orientation, color, texture) and physical properties (mass, friction, springs, and damping constant). Webots offer sensors commonly used in robotics: proximity, light, touch, force, accelerometer, gyroscope, compass, camera, a global positioning system (GPS), and LiDAR. This software also provides actuators, servos (rotational or linear), grippers, and LEDs [8].

Webots provides an interactive virtual world based on VRML that may be written in C, C++, Matlab, Python, Java, or Urbi to make programming and controlling the robot easier for developers. Simulation output can be exported as a PNG picture (screenshot) or as a MPEG/AVI movie. The user and the virtual environment can interact even when the simulation runs. Developers may add basic passive or active items to the simulated environment to make it more realistic [8]. The complete structure of Webots, consisting of three parts, such as virtual reality modeling language (VRML) world description, components, and controllers, is presented in Fig 4.

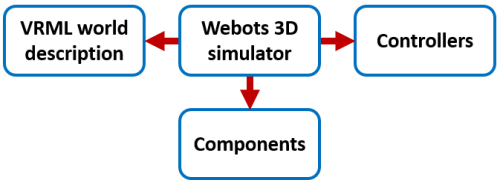

Fig. 4. The complete structure of Webots consists of three parts

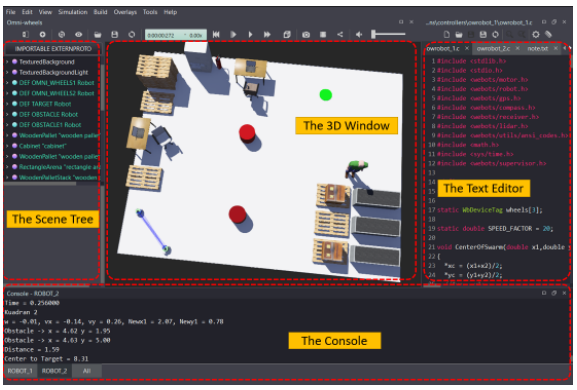

Fig. 5. Interface of Webots simulator

A real-world is modeled for algorithm certification for the first time. The world mainly contains the surrounding environment, robot models, and related sensors. Since Webots is a realistic simulator, it can provide rapid prototype environments in 3D virtual worlds.

Using the Webots simulation platform can quickly test your algorithms' pros and cons without damaging the robot. Still, it also provides an interface with the actual robot so you can use your algorithms to detect real robots in Webots. The step to perform multi-robot merged formation in Webots is to build models, such as scene models, robot models, and sensor models, and then compile a control program to execute the algorithm of merger formation. In the test, multiple nodes are used to model cooperative mobile robots and playground scenarios.

A. Webots Scene Model

Webots GUI (Graphic User Interface) consists of 4 main windows. The Scene tree interface is a hierarchical illustration of the current world (left). The 3D window interface lets in interplay with the 3D simulation (middle). The Text editor interface allows customers to edit and supply code compilation (right). Finally, the Console interfaces show each compilation and controller output (bottom). The Webots GUI is shown in Fig. 5.

The Scene Tree. A hierarchical representation defines a world. A world file contains the contents of a simulation. This world file contains the simulation's information, such as where objects are, how they look, how they interact with one another, the color of the sky, where the gravity vectors are, and so on.

Scene Modeling. Webots offers two ways to do scene modeling. First, create a scene from scratch. Second, use a sample world. It prefers to use a sample world close to your application when the sample exists. Because this research focused on omnidirectional robots, accordingly ”omni_wheels” sample world was used. To access it, go to the File menu first, then click ”Open Sample World”. It will showed-up five options, choose ”samples”. Then open the ”howto” option. The last action is to choose ”omni_wheels” and select ”omni_wheels.wbt”.

This example world requires some modifications to make it suited for the experiments, such as the arena’s pattern should be changed to draw the robot trajectory using a pen. To realize it, select ”RectangleArena” solid in the scene tree view, then delete the ”floorAppearance Parquetry”. To make this floor work with a pen, add a node in ”floorAppearance”, choose ”Base nodes”, then select ”PBRAppearance”. For the last setting, the ”metalness” becomes 0 in ”floorAppearance”.

The last step in modifying the arena is dimension—this research adjusted the arena dimension to 11 x 11 m. The ”floorSize” changed to 11 for both x and y. This option is in the same solid ”RectangleArena”. Next, set the ”floorTileSize” with 11 for both axes. The details setup of the rectangle arena is presented in Fig. 6 (a).

The Solid node. Buildings, junctions, trees, and other objects can be added to the environment to make it more realistic. The Solid node is introduced in this section. The physics engine of Webots is intended to emulate rigid bodies. Decomposing diverse elements into indivisible rigid bodies is critical to developing a simulation. Fig. 7 depicts the most basic Webots model of a rigid body, which has a graphical representation (Shape), a physical bound (boundingObject), and exists in a dynamic environment (Physics). The first step in defining a rigid body is to build a Solid node. All of these sub-nodes are optional, except for the boundingObject, which the physics field requires.

The obstacles must be added to the environment to make the simulation more realistic. Some simple cylinders simulate obstacles. First, ”Add New” on the scene tree view; then, in the ”Add a node” window, choose ”PROTO nodes (Webots Project)”. Second, choose ”objects”, then ”obstacles”, and then click ”OilBarrel (Solid)”.

B. Webots Robot Model

The parent node is the robot node, with numerous child nodes such as the box, sphere, servo, group, transform, and so on. In the sample world, the ”DEF OMNI_WHEELS Robot” was already created from the beginning. This node is important in the cooperative omnidirectional mobile robots model because it provides the driving source. To get two Omni-robot, copy and rename them to ”DEF OMNI_WHEELS1 Robot” and ”DEF OMNI_WHEELS2 Robot”. The scene tree in Fig. 6 (b) displays all the nodes utilized to model the simulation environment.

Next, the GPS sensor must be added to the robot model to detect their coordinates which is essential to conduct the

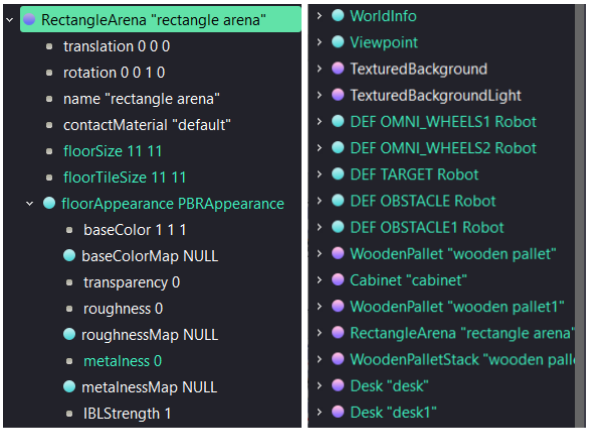

Fig. 6. The scene tree (a) details the setup of the rectangle arena, (b) two robots model and items of furniture

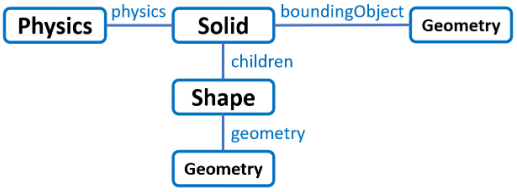

Fig. 7. A solid node and its sub-nodes

Lyapunov control computations. To add GPS to each robot, select the model of the robot, then ”Add a node” as children of each robot, and choose ”Base nodes”. The last thing is selecting the ”GPS”. The coordinates contain three variables that indicate q_1, q_2, and q_3-axes. This paper use 2D coordinates means q_1 and q_2-axes only. To have those two robots communicate with each other, nodes ”Emitter” and ”Receiver” should be added. Similarly, when the GPS sensor is added above, in ”Base nodes” choose ”Emitter”. Do the same procedure when adding the ”Receiver” node—nothing particular setting on those two nodes. A ”Pen” node should be added to analyze the robots’ motion. Like other nodes above, choose ”Base node”, then select ”Pen”. Some configurations are arranged, like ”inkColor” to set up a different color for each robot’s trajectories. For example, red color for robot 1, and blue color for robot 2. For the lines drawn on the floor to be sharp and thin, the values of ”inkDensity” and ”leadSize” are set to 1 and 0.0001, respectively.

C. LiDAR Sensor Model

Besides position sensor such as GPS, to simulate the actual world, the nodes of distance sensor such as LiDAR is applied to the simulation. The model in Webots should use commercial product specifications to make the simulation close to the real application. The popular commercial LiDAR sensor, such as LiDAR A1M8. Proto node ”RobotisLds-01” provided by Webots is more similar to the other LiDAR model. To put the node at the robot’s top, the translation in q_3-axes changed

to 0.135 m, then rotated for 3.14 radian to make the sensor face forward. Here with some arrangement to this node such as "fieldOfView" 6.28, "numberOfLayer" 1, "maxRange" 12, and "horizontalResolution" 360. Until this step, the simulation environments are completely set up. Fig. 9 shows the environment used in this research.

D. Webots Controller

A Webots controller file is created based on equations that describe omnidirectional robot movement. The program allows the definition of a robot's behavior through the use of a file named "Controller." The Webots controller file is developed in C language code in this work, which consists of the setup of nodes, motors, time control variables, and user-defined variables. Fig. 8 shows a flow diagram of cooperative omnidirectional mobile robots Webots controller file.

IV. Experimental Results and Analysis

A. Omni-wheel Robot Inverse Kinematics

The trajectory plan of each robot is calculated using the Lyapunov Function method. The current coordinates of each robot, target coordinates, and the coordinates of several obstacles, are the data needed to perform trajectory calculations using the Lyapunov function method. The result of this method is a change in the coordinates of the robot $\dot{\mathbf{q}}$. The Omni-wheel robot's inverse kinematics equation is used to get specified velocities to reach desired coordinates.

From Eq. 1, suppose the matrix A is as follows to simplify inverse kinematics calculation.

$$[A] = \begin{bmatrix} \mathrm{c}(\theta_1+\varphi) & \mathrm{c}(\theta_2+\varphi) & \mathrm{c}(\theta_3+\varphi) \\ \mathrm{s}(\theta_1+\varphi) & \mathrm{s}(\theta_2+\varphi) & \mathrm{s}(\theta_3+\varphi) \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

$$[A] = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} & -\frac{1}{2} & 1 \\ \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad (11)$$

Next, the inverse kinematics equation is as follows.

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}^{-1} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \quad (12)$$

$$[A]^{-1} = \frac{\begin{pmatrix} ei-fh & hc-ib & bf-ce \\ gf-di & ai-gc & dc-af \\ dh-ge & gb-ah & ae-db \end{pmatrix}}{aei+bfg+cdh-ceg-afh-bdi} \quad (13)$$

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} -0.33 & 0.58 & 0.33 \\ -0.33 & -0.58 & 0.33 \\ 0.67 & 0 & 0.33 \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} \quad (14)$$

Using Eq. 14 and Eq. 9, the velocity of each motor is defined successfully.

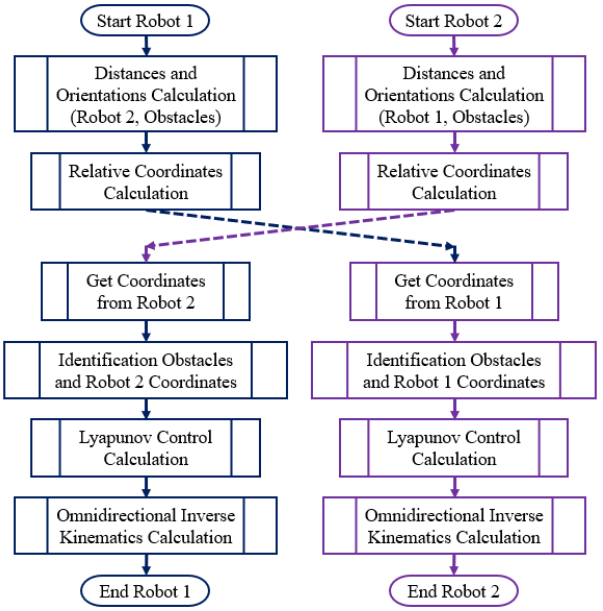

Fig. 8. Flowchart of the decentralized controller of robot 1 and robot 2

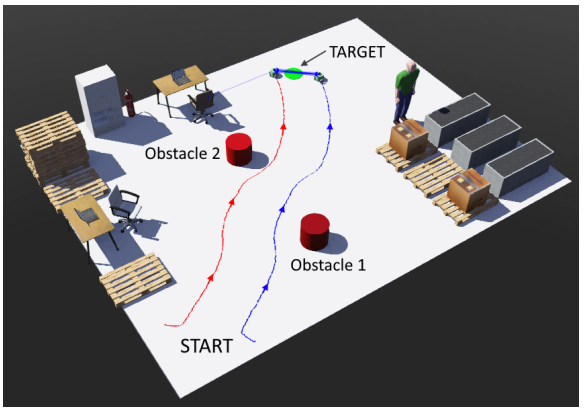

Fig. 9. The Webots simulation environment

B. Webots Simulation

Webots robot simulation software is used to validate the robot system. Each robot has a controller programmed in C language. There are two robot controls without supervisor control. The Webots simulation environment is set up in such a way as to be able to test the reliability of the proposed method. In the start position, the robots are placed far apart at a distance of more than the desired formation length. The target, two obstacles' positions, and the arena configuration are fixed. The virtual arena poses the same physical properties as a real arena in terms of friction coefficients, gravity, and collision forces. Fig. 9 presents the multi-robot implementation and the working environment in the Webots simulator.

Fig. 10 shows that the Euclidean distance between robots at the beginning is 1.67 m. By designing two gain functions for each robot, the robot moves closer and maintains a distance of

1 m. The values of the attraction to the center of the formation and inter-robots collision avoidance functions, $\gamma_i = 27$, and $\beta_{ij} = 1.52$, respectively. The gain of the obstacle avoidance function (λ_{ik}) is set to only 0.8, and the gain towards the target (α) is 40 making the robot formation able to avoid obstacles during the trip to the desired target.

The gain of the obstacle avoidance function is relatively low, and only one robot can sense the potential field of the obstacle. The simulation is arranged to have each robot immediately sense the potential field of the obstacle. When encountering an obstacle, the distance between the two robots in formation narrows slightly and then returns to a length of 1 m. Fig. 11 shows the speed response of two robots resulting from the inverse kinematics calculation of an Omni-wheel robot. Eq.10 proves the system's (3) stability mathematically, and testing the system's stability through the experiment is successfully shown in Fig. 11, where the speed of 3 wheels goes to zero.

The response of each Lyapunov function can be seen in Fig. 12. The initial attraction to the target function has a considerable value. It decreases gradually as the robot formation approaches the target position and reaches zero value after reaching the target coordinates. In contrast to the attraction function, the repulsive obstacle avoidance function reduces the value of the function to zero when close to an obstacle. Then its value increases after returning away from the obstacle. While the other two functions, such as attraction to the center of the formation, and repulsive inter-robot collision, the value of the function is minimal because the distance between the robots is relatively tiny.

V. Conclusions and Future Works

The case study based on the Lyapunov control focuses on finding the possibility of a distributed cooperative mobile robot using a LiDAR sensor to recognize the 'friend' robot and the obstacle. This paper shows that each robot can recognize well between its neighbor or obstacles and maintain its formation of 1 m during the trip with obstacle presence. The simulations were performed in Webots to verify the proposed algorithms. The mathematical analysis and the experiment prove the system's stability by seeing the robots' velocity. For future works, an analysis of gain variations will be carried out to determine the effects of each gain when the gain attraction to the target function is enlarged and the gain to maintain the formation is reduced or vice versa.

Acknowledgment

The Research Center and Community Engagement, University of Surabaya, Indonesia, supports this research financially.

References

[1] H. W. Agung and I. Nilkhamhang, "Hierarchical decentralized lqr control for formation-keeping of cooperative mobile robots in material transport tasks," *Engineering Journal*, vol. 25, no. 12, pp. 37–50, 2021.

[2] A. Khamis, A. Hussein, and A. Elmogy, *Multi-robot Task Allocation: A Review of the State-of-the-Art*, 05 2015, vol. 604, pp. 31–51.

[3] J. K. Verma and V. Ranga, "Multi-robot coordination analysis, taxonomy, challenges and future scope," *Journal of intelligent & robotic systems*, vol. 102, no. 1, pp. 1–36, 2021.

[4] R. Almadhoun, T. Taha, L. Seneviratne, and Y. Zweiri, "A survey on multi-robot coverage path planning for model reconstruction and mapping," *SN Applied Sciences*, vol. 1, no. 8, pp. 1–24, 2019.

[5] A. Farinelli, E. Zanotto, E. Pagello *et al.*, "Advanced approaches for multi-robot coordination in logistic scenarios," *Robotics and Autonomous Systems*, vol. 90, pp. 34–44, 2017.

[6] A. Farley, J. Wang, and J. A. Marshall, "How to pick a mobile robot simulator: A quantitative comparison of coppeliasim, gazebo, morse and webots with a focus on accuracy of motion," *Simulation Modelling Practice and Theory*, vol. 120, p. 102629, 2022.

[7] S. A. Kumar, J. Vanualailai, and B. Sharma, "Lyapunov-Based Control for a Swarm of Planar Nonholonomic Vehicles," *Mathematics in Computer Science*, vol. 9, no. 4, pp. 461–475, 2015. [Online]. Available: http://link.springer.com/10.1007/s11786-015-0243-z

[8] W. R2022b, "http://www.cyberbotics.com," open-source Mobile Robot Simulation Software since version R2019a. [Online]. Available: http://www.cyberbotics.com

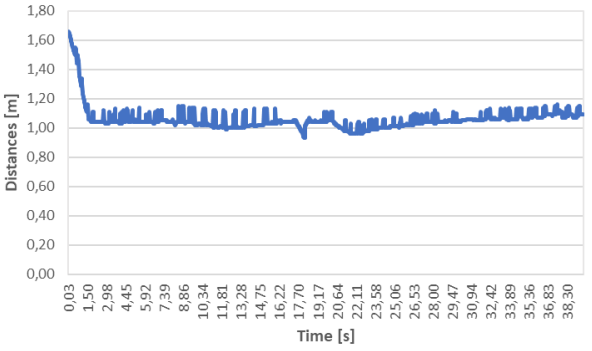

Fig. 10. The euclidean distances between robots

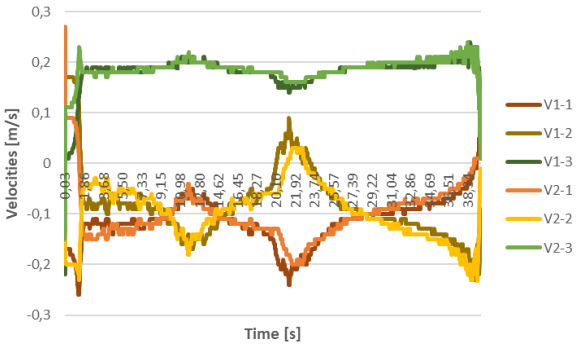

Fig. 11. The Omni-wheels' velocities of the robots

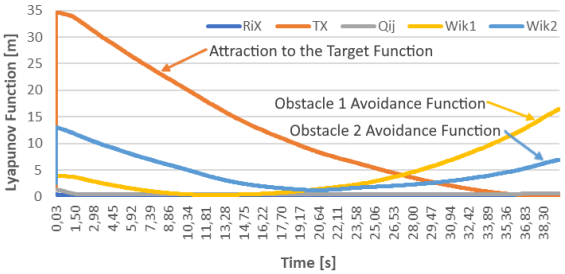

Fig. 12. The Lyapunov function states