Project Report for Master of Engineering

Simulation & Formation Control for Leader-Follower Wheeled Mobile Robots Based on Embedded Control Technique

University of Victoria

Department of Mechanical Engineering

Prepared by:

Chathusanka Jayasingha Appuhamilage | V00998312

Supervisor:

Dr. Daniela Constantinescu

Abstract

Formation control has garnered significant attention from researchers in recent times. This heightened interest can be attributed to its applicability in a wide range of tasks, including but not limited to search and rescue operations, agricultural coverage jobs, and area patrols. This surge in attention is primarily attributed to its ability to enhance efficiency, reliability, and the capacity to accomplish complex tasks effectively within these domains. Formation control for ground vehicles has particularly been useful for application such as cargo transportation, cooperative manipulation, and surveillance and exploration and many more [2]. Combining sensors on robots in a formation enables them to scan larger areas more quickly by directing sensors in different directions, surpassing the smaller area of coverage of a single robot can achieve during the same time. This enhances the efficiency of the search and exploration process compared to individual robot operations. For this project, a formation control strategy based on the embedded control technique for a leader-follower system was used. The proposed technique divides the formation control problem into two subtasks, which is different from the traditional design philosophy of basing the formation controller design directly on the formation tracking errors [3]. Two subtasks are virtual signal generator and trajectory tracking controller. The virtual signal generator achieves the desired formation control goal and outputs a reference signal to the trajectory tracking controller of the follower [3]. The embedded control technique reduces the complexity of directly implementing formation tracking errors while providing several other benefits as well. A predesigned mobile robot model in CoppeliaSim was used as the leader and follower. The robots were programmed to implement the controller and simulate the system. Finally, a number of tests were performed adjusting gains and observing the performance of the robots to identify the optimal performance gain values. Gains were adjusted with the goal of minimizing the error and reducing the motion jerk of the follower.

Content

List of Figures

1. Introduction

The focus on this project is on the leader-follower formulation method for 3-wheel WMRs (Wheeled mobile robot). The wheeled robots used in this simulation are differential wheeled robots; meaning the desired movement of the robot is achieved by only controlling the speed of two separately driven wheels. The third wheel is a free moving caster wheel, which provides more stability for the robot by preventing it from tilting. In this project, one robot is appointed as the leader and the other robot is appointed as the follower. CoppeliaSim, an industry used robot simulator, is used to simulate the real time behaviour of the system. Thanks to its built-in physics simulation libraries, a more realistic simulation can be achieved using appropriate friction values that closely resemble a real life scenario with friction as opposed to pure mathematical simulation which doesn't account for external factors such as friction, weight of the robots etc.

The Formation controller is designed to make each follower follow the movement of its leader and keep the formation in the proper shape relative to the leader. The two main classical structures for the leader-follower controllers are separation-bearing control (l-Φ) and separation-separation control (l-l). In separation-bearing mode the desired relative distance l and the angle Φ between the leader and the follower are considered while the separation-separation mode tries to maintain two relative x and y distances (l_1 and l_2) from leader to follower [3]. For this project, the separation-bearing control (l-Φ) method will be used.

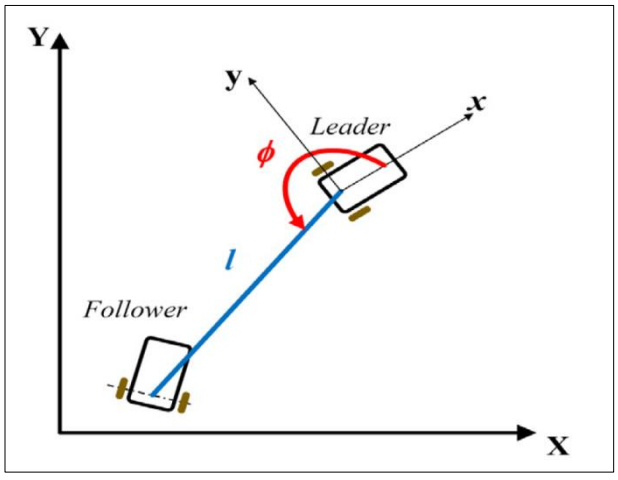

Figure 1: separation-bearing control (l-Φ) [14]

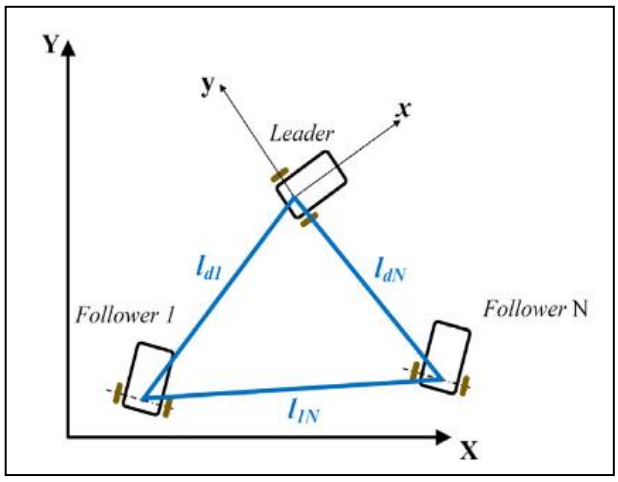

Figure 2: separation-separation control (l-l) [14]

These two classical structures have been the basis for recent formation controls that use a combination of nonlinear strategies such as sliding mode control, adaptive back-stepping control, model predictive control, and neural network-based control [3].

Most of those above mentioned formation controls use a conventional formation control design philosophy in which the formation tracking errors of WMRs are directly used for the control design. A simple schematic of how signals are sent is shown in Figure 3 below.

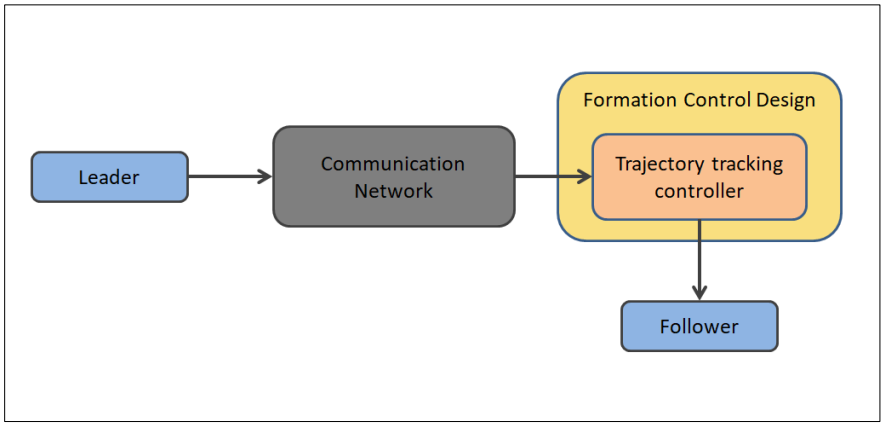

Figure 3: Conventional control design schematic

The communication network in Figure 3 indicates the communication method used to broadcast information from leader to follower. It is common for some WMRs to have predesigned control systems in order to achieve their own independent tasks [3]. These different types of predesigned controllers can be used for different control purposes. Examples of such predesigned controllers can be pointed out as path tracking controllers, trajectory tracking controllers, posture stabilization etc. [4][3]. Trajectory tracking is about keeping a system stable while following a reference that changes over time. If the aim is to stick to a fixed geometric reference without time considerations, it's called Path Following [15]. The drawback of conventional design philosophy is, when the formation goal (path tracking, trajectory tracking etc), these predesigned WMRs' controllers need to be redesigned or replaced, which is very difficult and not ideal considering these predesigned WMRs are tightly enclosed and their controllers cannot be set at will [3].

To bridge this limitation, paper [3] suggests an alternative design to the conventional formation control design philosophy. Paper [3] calls this technique the embedded control technique. The embedded control technique splits the control task into two sub tasks; namely, the virtual signal generator and the agents' tracking controller design [3].

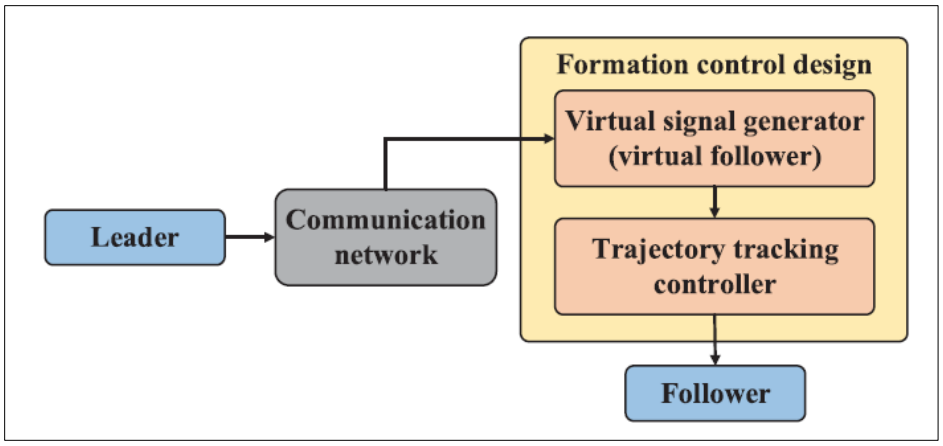

Figure 4: Embedded formation control design schematic [3]

The virtual signal generator (or virtual follower) generates the desired goal based on the current state of the leader. In other words, the virtual follower has a WMR kinematic model which always achieves the desired formation goal. Thus it always maintains the required distance and the bearing angle with the leader [3]. This generator output is embedded into the feedback loop of the follower as follower's reference trajectory. A general tracking controller is then designed for the follower to track the virtual follower's reference output. This two-step embedded control technique design enables the leader-follower WMR system to achieve formation control [3]. While it's possible to design a tracking controller directly for the follower without employing a virtual follower, introducing a virtual follower can simplify the complexities associated with directly implementing a trajectory tracking controller for the follower. The advantages of incorporating a virtual follower are explained below in the advantages section.

The embedded formation design controller designed in this project is a dynamic controller as opposed to a static controller. The goal of a static controller is to bring the robots in the system to a certain formation, and once the robots achieve the formation the robots come to a stop. In a dynamic controller, such as the one designed in this project, the control goal is to maintain the formation while the leader is on the move. Therefore, all mathematical derivations in this project are based around the assumption that the leader is always moving.

The embedded control technique offers 3 main advantages over the conventional design philosophy method.

1) *Simplified Modular Design*: The embedded control technique separates the leader-follower WMR formation control design into two distinct steps; the virtual signal generator design and the trajectory tracking controller design. Directly implementing formation controllers with the formation tracking errors on WMRs' models causes difficulties and complexity which can be simplified using the above embedded control technique [3].

2) *Plug and Play Function*: In conventional design philosophy, formation controllers need to be redesigned to replace the original trajectory tracking controllers of WMRs whenever the formation goal changes. This is difficult and not convenient. The modular control scheme of the embedded control technique provides a plug and play function which provides a more convenient

way. Changing its reference trajectory by adjusting the corresponding signal generator output, a WMR can now join a new formation task without having to change the original tracking controller [3].

3) *Application-Oriented Feature*: The embedded modular control approach is able to lessen the challenges of applications brought on by the encapsulated properties of WMRs because this strategy only uses kinematics of WMR which offers a great practical universality. The encapsulated properties are bundled and encapsulated within the WMR. Example of some encapsulated properties would be that the dynamic parameters of WMRs are unknown and support only the pose and velocity inputs [3]. Kinematics is concerned with describing motion in terms of position, velocity, and acceleration without considering the forces involved, while dynamics involves the study of the forces, torques, and energy that cause and affect motion. These encapsulated properties could also include types of sensors equipped on the WMR, details about the controls system and the software used for navigation and control, communication protocol etc.

2. Literature Review

This project aims to address the leader-follower formation control problem and simulate it in CoppeliaSim, a robot simulator used in industry. Formation control is a particular aspect of multi-robot coordination that has drawn a lot of interest in recent years [2]. Formation controlling can be simply explained as maintaining the relative position of and the orientation of a group of robots while enabling the group to move as a whole. Cooperative formation controls offer better efficiency, fault tolerance, reliability, stronger robustness, better economy and complex task achieving capabilities that a single mobile robot lacks [2][3]. Numerous researches have been carried out for formation control of different types of vehicles such as ground vehicles, unmanned aerial vehicles, aircrafts, surface and underwater autonomous vehicles [1]. Autonomous mobile vehicles offer a wide range of applications such as searching, surveillance and exploration applications to cargo transportation, and cooperative manipulation [2]. Formation also plays an important role in scenarios like military applications where sensors are limited. Individual team members can focus their sensors on a certain area of the surroundings while their partners cover the rest thanks to formations [10]. For instance, fighter pilots in the Air Force prioritize their visual and radar search duties based on where they are in a formation. In DARPA's Demo II project also, robotic scout vehicles in a formation direct their sensors in various directions to achieve a complete coverage. The formation approach can be used in numerous other fields, including security patrols, agricultural covering jobs, and search and rescue operations [10].

In literature, three main methods have been used for the robot formation control problem: behaviour-based, virtual structure and leader following [1]. In the behaviour-based approach, each robot is assigned with some low-level actions (sub tasks), which are needed to be executed individually to achieve the group behaviour (or mission) [1][2]. The importance of each sub task is considered to determine the resulting action of each robot [1]. These group behaviours are inspired by common behaviours that can be seen in nature such as flock of birds, schools of fish, random walks of ants and school of fish [2][10]. Studies of flocking and schooling demonstrate that these behaviors develop as a result of a desire to remain in the group while also maintaining a separation distance from other group members [10]. Animals benefit from these behaviors in various ways. For example, animals in a herd benefit each other by decreasing predator encounters as the chance to detect a predator is increased when animals combine their sensors to survey the surrounding. Combined sensors also benefit the animals in the herd as the efficiency for food search is increased [10]. Researchers in robotics and the artificial life community have drawn inspiration from these biological studies to develop formation behaviors for both robots and simulated agents since groups of artificial agents could similarly benefit from formation strategies. Robotic formation generation methods can be characterized by their sensing needs, behavioral integration technique, and preplanning stance [10]. Behavior-based systems are able to navigate to waypoints, avoid collisions, and maintain formation simultaneously due to the concurrent integration of multiple goal oriented behaviors [10]. The main downside to this approach, however, is that it is challenging to guarantee the convergence of the formation to the desired group behaviour due to the complexity of mathematical formulation [1].

In the virtual structure approach, the group formation is considered as one virtual rigid structure. Thus the behaviour of the group can be seen as something similar to that of a physical object [1]. In this case the desired trajectories are assigned to the group as a whole as opposed to individual robots receiving them

and the coordination is accomplished through shared knowledge of virtual structure's states [1][11]. With this method, the general solution to the problem of moving in formation relies on the notion that each individual robot in a virtual structure will move in the direction of the force when a virtual force field is applied to the virtual structure. The relative positions of the mobile robots are then used to determine the position and orientation of the virtual structure. Therefore, it is possible to say that the virtual structure moves to fit the robots' present positions and the robots move to remain in the virtual structure [12]. The virtual structure is made up of a random number of points, and for simplicity this number can be taken as the number of robots in the formation. The initial step is to align the virtual structure to the robots, and this can be done by constructing a fixed one-to-one mapping between the points on the virtual structure and each robot [12]. When the robotic system is initialized, a fixed mapping from the robots to the virtual structure is established. By minimizing errors of the virtual structure points with the corresponding positions of the robots, the alignment of the virtual structure with robot positions is then executed [12]. Now that the robots are fitted into the virtual structure, moving the virtual structure is the next step. The virtual structure can be moved by simply adding a displacement to the translation and rotation of which the magnitude is determined by nature of the task and the capabilities of the robots [12]. Caution must be taken to avoid moving the structure beyond the capabilities of the robots in order to avoid errors [12]. The final step is to compute individual robot velocities necessary to move the robots onto the displaced virtual structure and adjust wheel velocities to make them follow the calculated trajectories [12]. These steps are then followed again in a loop. The advantage of the virtual structure method is being able to predict the exact behaviour of the formation, although a larger inter-robot communication bandwidth is usually required [1].

In the leader-follower method, the appointed leader robot (or robots) decides the trajectory of the whole group, and the rest of the robots, followers, are instructed to follow the leader maintaining the desired relative distance and the orientation to the leader while keeping the group formation unchanged [1][2]. Several variations in this method can be found in the literature. Paper [1] proposes a leader-follower alternate setup where the desired angle between the leader and the follower is measured in the reference frame of the follower as opposed to the commonly used leader reference frame. Compared to the typical leader-follower techniques defined on the leader reference frame, this approach ensures lesser control effort and smoother trajectories for the follower [1]. Paper [13] proposes a synchronization control approach to trajectory tracking leader- follower formation problem. In a traditional leader-follower formation controller, it can only achieve a desired formation by driving the trajectory tracking errors to zero. Each robot's control loop only receives local feedback from the leader, and the followers aim to complete the necessary tracking task without interacting with any other robots. In the proposed synchronization method, followers also receive feedback from other followers who are in trajectory tracking. In other words, in addition to the position errors' convergence to zero, the manner in which they do so will also be taken into account. Thus, synchronization controllers have the ability to control how the robots get into the desired position as well. This method is ideal for problems with trajectory tracking control of several robots while following desired time-varying formations [13].

Leader-follower architecture has a simple control structure that is easy to scale and implement. Due to less computational load, it is a suitable method for real-time implementations [3]. However, this method heavily relies on the leader to achieve the goal, and if the leader malfunctions, the whole system will stop working. Hence, this approach is not desirable in situation where adverse conditions exist [1].

3. Kinematic Model of Differential Wheel Robots

Shown below in Figure 3 is a schematic diagram for variables related to a WMR.

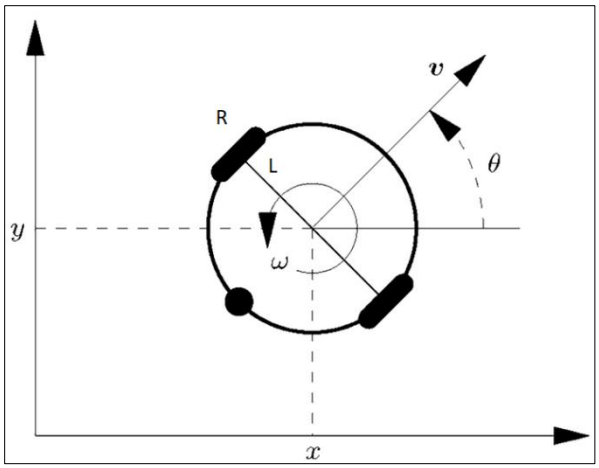

Figure 5: WMR parameters [6]

In Figure 3, *L* is the wheelbase (the distance between the two wheels) and *R* is the radius of the wheel. The variable *v* is the linear velocity and ω is the angular velocity of the robot. The linear velocity is always in the direction of robot's steering.
The angular velocities of the left and right wheels are ω_l and ω_r. These are the only user-controllable variables for the WMR so ω_l and ω_r are considered as inputs for the WMR.
The goal is to come up with equations for ω_l and ω_r in terms of known constants and *v* and ω. Then the required inputs ω_l and ω_r can be calculated in order to achieve a given velocity (*v*) and an angular velocity (ω) for the robot.

The kinematic model of the robot is [7]

$$\dot{x} = v\cos\theta \quad (1)$$
$$\dot{y} = v\sin\theta \quad (2)$$
$$\dot{\theta} = \omega \quad (3)$$

The differential drive wheel model for the robot is [5]

$$\dot{x} = \frac{R}{2}(\omega_r + \omega_l)\cos\theta \quad (4)$$
$$\dot{y} = \frac{R}{2}(\omega_r + \omega_l)\sin\theta \quad (5)$$
$$\dot{\theta} = \frac{R}{L}(\omega_r - \omega_l) \quad (6)$$

From (1) and (4),

$$v\cos\theta = \frac{R}{2}(\omega_r + \omega_l)\cos\theta$$

$$\frac{2v}{R} = \omega_r + \omega_l \qquad (7)$$

From (3) and (6),

$$\omega = \frac{R}{L}(\omega_r - \omega_l)$$
$$\frac{\omega L}{R} = \omega_r - \omega_l \qquad (8)$$

Now (7) and (8) can be solved for ω_r and ω_l.

$$\omega_r = \frac{2v+\omega L}{2R} \qquad (9)$$

$$\omega_l = \frac{2v-\omega L}{2R} \qquad (10)$$

Variables v and ω can have both positive and negative values. Variables *L* and *R* are known, measured dimensions of the robot. Once required ω_r and ω_l are calculated, they can be directly input to the WMR to control the wheel velocities and thus its motion.

4. Mathematical Formulations

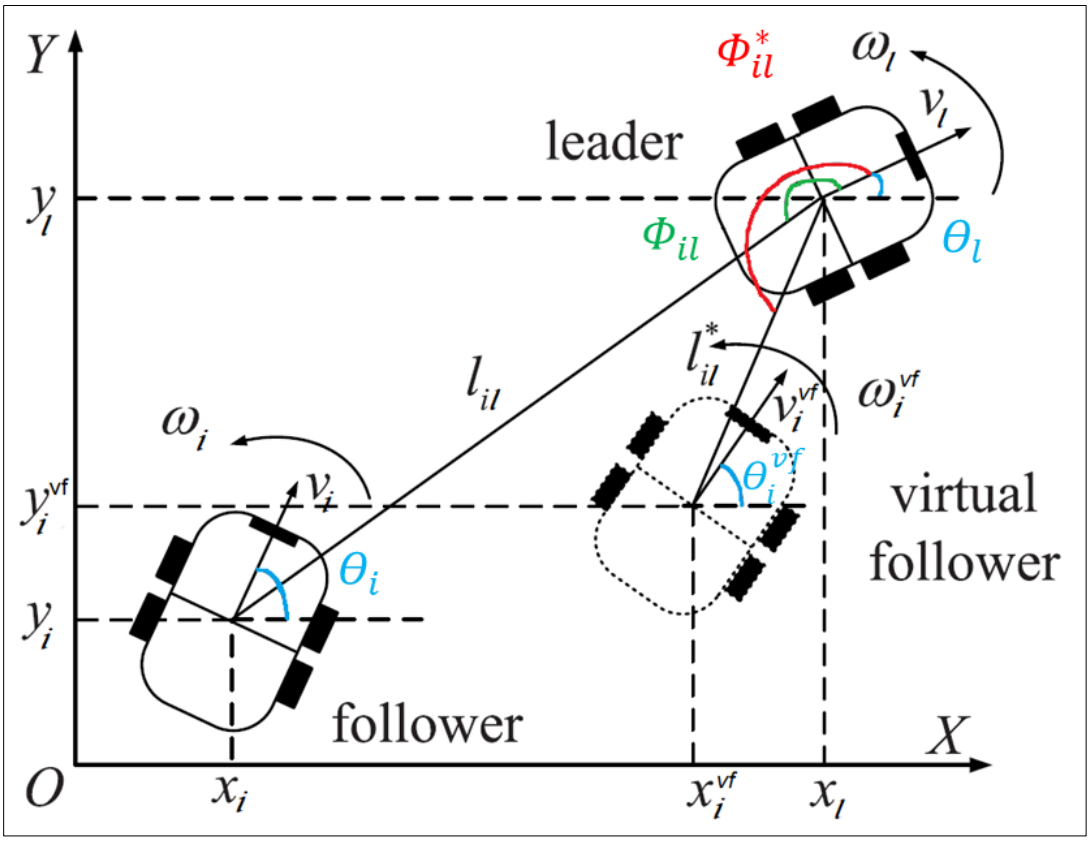

Figure 6: Formation related variables [3]

Figure 4 shows the schematic diagram of the formation of a leader-follower system. This diagram includes one follower, one leader and one virtual follower corresponding to the follower.

The first step is to decide the formation that needs to be achieved by defining the distance l and the angle Φ between the leader and the follower. In this project, the notation l_{il} will be utilized to denote separation, while Φ_{il} will represent the bearing angle between the leader and the follower. The desired distance and the bearing angle between the leader and the follower are defined as l_{il}^* and $\Phi_{il}^* \in [0,2\pi)$ respectively. The equations presented in this section are the same as those utilized in paper [3].

The kinematic model of the leader is

$$\dot{x}_l = v_l \cos\theta_l$$
$$\dot{y}_l = v_l \sin\theta_l$$
$$\dot{\theta}_l = \omega_l \qquad (11)$$

The separation and the bearing angle can be calculated as follows.

$$l_{il} = \sqrt{(x_l - x_i)^2 + (y_l - y_i)^2}$$

$$\Phi_{il} = mod(\ arctan2(y_l - y_i, x_l - x_i) - \theta_l + \pi, 2\pi\ ) \qquad (12)$$

The term "*mod*" in equation (12) is a function that returns the remainder after the first number is divided by the second number inside the parentheses (the two numbers are separated by a comma). The positions, velocities and accelerations of the leader and the follower can be measured in real time through the software.

Now the formation tracking errors can be defined as:

$$e_{l_{il}} = \ l_{il}^* - l_{il}$$
$$e_{\Phi_{il}} = \ \Phi_{il}^* - \Phi_{il} \qquad (13)$$

The goal of the project is to design a controller for the follower to maintain the desired l_{il}^* and Φ_{il}^*values. In other words, the formation tracking errors should be driven as close as to zero to maintain the formation of the leader and the follower [3].

4.1 Virtual Signal Generator Design

The virtual signal generator is designed to always keep the desired relative distance and the bearing angle with the leader. The virtual signal generator can be considered as a virtual follower with respect to the leader. Furthermore, the same kinematic model used for the leader in equation (11) can also be used for the virtual follower as well.

The following assumptions are considered for the signal generator design.

- $v_l(t)$ and $\omega_l(t)$ are differentiable. Velocities and accelerations of the leader are bounded for $t > 0$
- $0 < v_l^{min} \leq |v_l(t)| \leq v_l^{max}$ and $\omega_l(t) \leq \omega_l^{max}$
- If $\Phi_{il}^* = \pi/2$ (or 90^0), then $v_l \ \neq \ l_{il}^* \omega_l$
- If $\Phi_{il}^* = 3\pi/2$ (or 270^0), then $v_l \ \neq \ -l_{il}^* \omega_l$ (14)

Similar to the leader, the kinematic model of the virtual follower is,

$$\dot{x}_i^{vf} = \ v_i^{vf} \cos\theta_i^{vf}$$
$$\dot{y}_i^{vf} = \ v_i^{vf} \sin\theta_i^{vf}$$
$$\dot{\theta}_i^{vf} = \ \omega_i^{vf} \qquad (15)$$

Variables related to virtual follower are superscripted with a "*vf*". Here, x_i^{vf} and y_i^{vf} are *x* and *y* positions, $\theta_i^{vf}$is the orientation, v_i^{vf} is linear velocity and ω_i^{vf} is angular velocity of the virtual follower.

The desired distance and the bearing angle of the desired formation can be written as;

$$l_{il}^{*\,2} = (x_l - x_i^{vf})^2 + (y_l - y_i^{vf})^2$$
$$\Phi_{il}^* = mod(\ arctan2\big(y_l - y_i^{vf}, x_l - x_i^{vf}\big) - \theta_l + \pi, 2\pi\ ) \qquad (16)$$

With the assumptions made in (14) for the leader, it is possible to state that $v_i^{vf}(t)$ is differentiable. Also, $v_i^{vf}(t)$, $\dot{v}_i^{vf}(t)$, and $\omega_i^{vf}(t)$ are bounded for $t > 0$ with $0 < v_i^{vf^{min}} \leq |v_i^{vf}(t)| \leq v_i^{vf^{max}}$ and $\omega_i^{vf}(t) \leq \omega_i^{vf^{max}}$. The proof is given below.

It is possible to obtain the following using (16)

$$x_i^{vf} = x_l + l_{il}^* \cos(\theta_l + \Phi_{il}^*)$$
$$y_i^{vf} = y_l + l_{il}^* \sin(\theta_l + \Phi_{il}^*) \qquad (17)$$

By taking time derivatives of (17), the following equations can be derived

$$\dot{x}_i^{vf} = v_i^{vf} \cos\theta_i^{vf} = v_l \cos\theta_l - l_{il}^* \omega_l \sin(\theta_l + \Phi_{il}^*)$$
$$\dot{y}_i^{vf} = v_i^{vf} \sin\theta_i^{vf} = v_l \sin\theta_l + l_{il}^* \omega_l \cos(\theta_l + \Phi_{il}^*) \qquad (18)$$

It follows from (18) that:

$$v_i^{vf^2} = \dot{x}_i^{vf^2} + \dot{y}_i^{vf^2}$$
$$v_i^{vf^2} = v_l^2 + l_{il}^{*2}\omega_l^2 - 2l_{il}^* v_l \omega_l \sin\Phi_{il}^*$$
$$v_i^{vf^2} = v_l^2(1 - \sin^2\Phi_{il}^*) + (v_l \sin\Phi_{il}^* - l_{il}^*\omega_l)^2 > 0 \qquad (19)$$

Under the assumptions in (14), v_i^{vf}(t) is bounded for $t > 0$ with $0 < v_i^{vf^{min}} \leq |v_i^{vf}(t)| \leq v_i^{vf^{max}}$, where $v_i^{vf^{min}}$, and $v_i^{vf^{max}}$ are positive constants that are calculated using $v_l(t)$, $\omega_l(t)$, l_{il}^*, and Φ_{il}^*. Hence, at least one of the variables $\dot{x}_i^{vf}(t)$ and $\dot{y}_i^{vf}$ has a value that is not equal to zero [3]. In other words, $\dot{x}_i^{vf}(t)$ and $\dot{y}_i^{vf}$ cannot be both equal to zero simultaneously. Therefore, from equation (19), it is obtained that v_i^{vf}(t) is not equal to zero. Physically, this means that the velocity of the virtual follower can never be zero. This is to be expected as the controller that is designed in this project is a dynamic controller.

Variables v_l and ω_l become differentiable with the first two assumptions made in (14). By taking the time derivative of (18);

$$\ddot{x}_i^{vf} = \dot{v}_l \cos\theta_l - v_l\omega_l \sin\theta_l - l_{il}^*\omega_l^2 \cos(\theta_l + \Phi_{il}^*) - l_{il}^*\dot{\omega}_l \sin(\theta_l + \Phi_{il}^*)$$
$$\ddot{y}_i^{vf} = \dot{v}_l \sin\theta_l + v_l\omega_l \cos\theta_l - l_{il}^*\omega_l^2 \sin(\theta_l + \Phi_{il}^*) + l_{il}^*\dot{\omega}_l \cos(\theta_l + \Phi_{il}^*) \qquad (20)$$

The same variables can also be derived by taking the time derivative of (15)

$$\ddot{x}_i^{vf} = \dot{v}_i^{vf} \cos\theta_i^{vf} - v_i^{vf}\omega_i^{vf} \sin\theta_i^{vf}$$
$$\ddot{y}_i^{vf} = \dot{v}_i^{vf} \sin\theta_i^{vf} + v_i^{vf}\omega_i^{vf} \cos\theta_i^{vf} \qquad (21)$$

Now (18) and (21) together leads to:

$$\theta_i^{vf} = arctan2\left(\dot{x}_i^{vf}, \dot{y}_i^{vf}\right) + 2n\pi$$
$$v_i^{vf} = \dot{x}_i^{vf}\cos\theta_i^{vf} + y_i^{vf}\sin\theta_i^{vf}$$
$$\dot{v}_i^{vf} = \ddot{x}_i^{vf}\cos\theta_i^{vf} + \ddot{y}_i^{vf}\sin\theta_i^{vf}$$

$$\omega_i^{vf} = \frac{\ddot{y}_i^{vf}\cos\theta_i^{vf} - \ddot{x}_i^{vf}\sin\theta_i^{vf}}{v_i^{vf}} \qquad (22)$$

Variable *n* in (22) indicates that θ_i^{vf} is continuous and differentiable. Variables θ_l and θ_i are also continuous and differentiable.

Assumptions in (14) introduce some limitations to this controller. With the first two assumptions, it indicates the goal of this controller is to achieve dynamic formation tracking rather than static formation generation where WMRs move until they achieve the desired formation and stop once they do [3]. In other words, the velocity of the leader can never be zero, and the controller is designed to maintain the formation as long as the leader keeps moving. Due to the last two assumptions, this dynamic controller is not capable of handling the scenario when the desired angle Φ_{il}^* is equal to $\frac{\pi}{2}$ or $\frac{3\pi}{2}$ [3]. These last two assumptions are necessary to make equation (19) valid if the desired angle Φ_{il}^* is equal to $\frac{\pi}{2}$ or $\frac{3\pi}{2}$. Otherwise, y_i^{vf} can become zero and this will make ω_i^{vf} goes to infinity in equation (22) as the denominator of equation (22) is y_i^{vf}.

The trajectory tracking errors $x_{e_{ivf}}$, $y_{e_{ivf}}$, and $\theta_{e_{ivf}}$ can be defined as follows

$$\begin{bmatrix} x_{e_{ivf}} \\ y_{e_{ivf}} \\ \theta_{e_{ivf}} \end{bmatrix} = \begin{bmatrix} \cos\theta_i & \sin\theta_i & 0 \\ -\sin\theta_i & \cos\theta_i & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_i^{vf} - x_i \\ y_i^{vf} - y_i \\ \theta_i^{vf} - \theta_i \end{bmatrix} \qquad (23)$$

Using the virtual signal generator, the leader-follower WMR formation control problem is transformed into a trajectory tracking problem for the follower. The goal of the trajectory tracking controller is to make the real follower track the virtual follower accurately while the trajectory tracking errors $x_{e_{ivf}}$, $y_{e_{ivf}}$, and $\theta_{e_{ivf}}$ converge to zero asymptotically. As long as these conditions are met, the follower is able to maintain the desired distance l_{il}^* and the bearing angle Φ_{il}^* relative to the leader. Thus the formation tracking problem for the leader-follower WMR is solved [3].

4.2 Trajectory Tracking Controller Design

With the results obtained in (22) and (23), the trajectory tracking controller for the follower can be designed by [3]:

$$v_{icon} = k_1 x_{e_{ivf}} + v_i^{vf}\cos\theta_{e_{ivf}}$$

$$\omega_{icon} = \omega_i^{vf} + k_2\theta_{e_{ivf}} + k_3 y_{e_{ivf}} v_i^{vf} \int_0^1 \cos(s\theta_{e_{ivf}})ds \quad (24)$$

Here, k_1, k_2, k_3 are positive gains that can be adjusted accordingly. The goal is to adjust k values until a balance is achieved between the error and motion jerk. The challenge of jerkiness persists even at slower speeds, primarily due to considerations related to the initial position. Even if the leader robot is commanded to move at a reduced speed, a significant starting relative distance between the leader and the follower can lead to issues during the formation phase. In such cases, the follower may attempt to reach higher speeds than the current speed of the leader, resulting in undesirable jerks in the motion of the follower. Higher k values tend to reduce error, but also make the follower's motion jerky and less stable. More details on how k values affect WMRs will be explained in results. Variables v_{icon} and ω_{icon} are the control inputs for the follower. The two inputs can be plugged into (9) and (10) to control the follower.

4.3 Trajectory Tracking Controller Design with saturated velocities

In practical scenarios, WMRs have limitations such as maximum velocity they can achieve. When linear and angular velocities of a WMR are saturated, the input velocities v_{icon} and ω_{icon} also become saturated. Saturated velocities refer to a situation where the velocity of a robot has reached its maximum allowable or specified limit. The paper [3] suggests a solution to avoid this velocity saturation. The control input for the follower $v_{icon}(t)$, and ω_{icon}(t) should satisfy

$|v_{icon}(t)| \leq v_i^{max}$, $\quad |\omega_{icon}(t)| \leq \omega_i^{max}$ for all $t > 0$ [3]

Variable v_i^{max} is the maximum linear velocity and ω_i^{max} is the maximum angular velocity the follower can achieve.
Furthermore, in order to ensure the formation realization, the velocities of the leader v_l, ω_l, and the desired formation parameters l_{il}^* and Φ_{il}^* must be chosen to satisfy

$v_i^{max} > v_i^{vf^{max}}$ and $\omega_i^{max} > \omega_i^{vf^{max}}$. [3]

Now the constrained trajectory tracking controller for the follower with saturated velocities can be designed as follows [3]

$$v_{icon} = \frac{k_1 x_{e_{ivf}}}{\sqrt{x_{e_{ivf}}^2 + y_{e_{ivf}}^2 + 1}} + v_i^{vf} \cos\theta_{e_{ivf}} \quad (25)$$

$$\omega_{icon} = \omega_i^{vf} + \frac{k_2\theta_{e_{ivf}}}{\sqrt{\theta_{e_{ivf}}^2 + 1}} + \frac{k_3 y_{e_{ivf}} v_i^{vf} \int_0^1 \cos(s\theta_{e_{ivf}})ds}{\sqrt{x_{e_{ivf}}^2 + y_{e_{ivf}}^2 + 1}} \quad (26)$$

Here, k_1, k_2 , and k_3 are positive gains that can be adjusted and v_{icon} and ω_{icon} are the control inputs for the follower.

5. CoppeliaSim Implementation

With the equations derived for the control inputs for the follower, controller can now be implemented and simulated in CoppeliaSim.

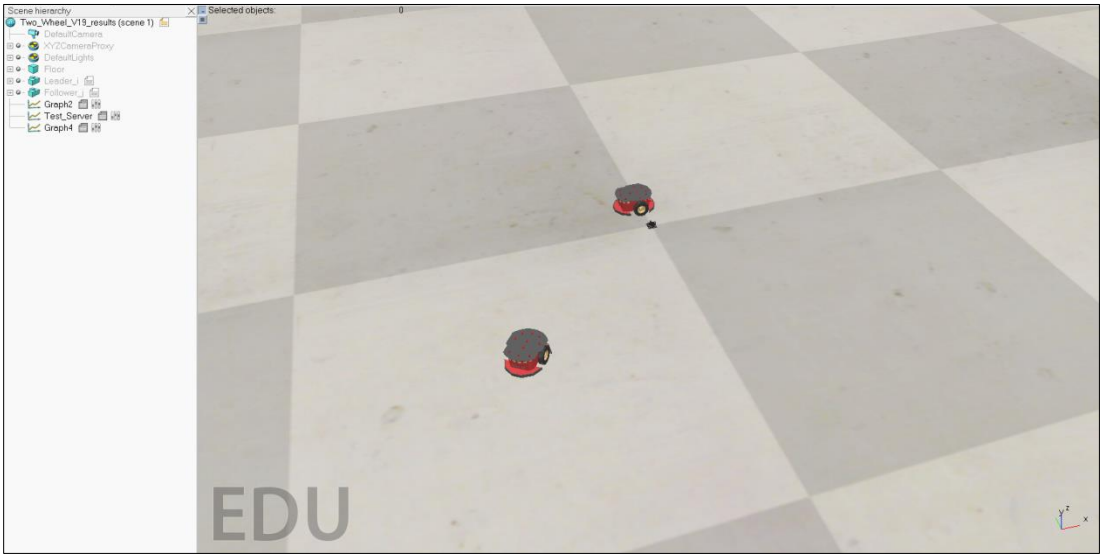

Figure 7: Simulation in CoppeliaSim

CoppeliaSim has several built in mobile robots. Out of those, Pioneer p3dx, a close resemblance of the actual Pioneer 3-DX robot [8], is chosen for this project as it is the only two wheel differential drive robot model available, and it is well suited for the purpose of this simulation. The default program that came for this robot was completely removed, and the robot was programmed from scratch as it offered more freedom to shape how the robot behaved.

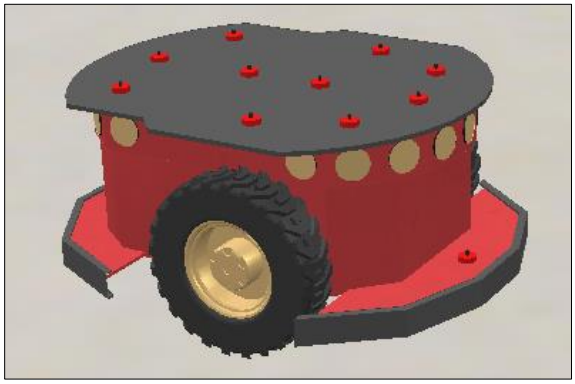

Figure 8: Pioneer p3dx model

A separate function was written to move the robot. The function accepts the required linear and angular velocity that needs to be achieved as inputs for the function and calculates the angular velocities for the left and right wheels (ω_l and ω_r) that need to be provided to the robot using the equation (9) and (10).

With the function that moves the robot in place, a way to provide inputs for the leader should be explored now; in other words, a way to control the leader. For this, CoppeliaSim's custom UI plugins can be used. These plugins allow the user to create a custom UI window and modify it according to user specific needs. A variety of widgets were available, but "spinbox" widgets stood out to be promising for controlling the leader as it offered the option to add elements that can be increased or decreased by the user [9]. Each element can be assigned for a variable that is used to control the leader such as linear velocity and angular velocity. The linear velocity is measured in meters per second (m/s), and the angular velocity is measured in degrees per second (deg/s). Both linear velocity and angular velocity are assigned a step size of 0.1, indicating that only one decimal point is considered relevant for these two values.

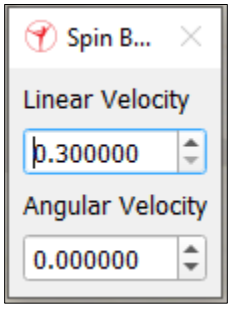

Figure 9: UI with "spinbox"

Moreover, the "spinbox" widget permits decimal value increments for its elements. It also allows user to either enter a value in the text field or increase or decrease the value using the arrows. Opting for arrows as the control method proves effective, as it enables the robot to execute fluid transitions in both speed and direction. The ability to make fine, decimal-based adjustments also further enhances the robot's control precision.

Whenever the value of linear velocity and angular velocity is changed in the UI by the user, CoppeliaSim runs a separate function, which is then used to update the new velocity variable values and pass the new values to the function that moves the robot.

CoppeliaSim doesn't have a built in function to calculate the acceleration of an object. Therefore this has to be calculated manually. Velocity of the leader in equation (19) was used to calculate the acceleration. The function "getSimulationTimeStep" retrieves the time step specific to each iteration of the loop. Subsequently, this time step was employed to compute the velocity alteration within that interval, consequently yielding the acceleration.

Ensuring the continuity of angles proved to be a pivotal aspect of the simulation. To achieve this, a comparison was made between the current and previous angles, with a focus on their positions within the first and fourth quadrants. The focus on degree values in first and fourth quadrant is due to the fact that the angle reset only occurs during jumps between these quadrants according to the way the angles are programmed. For instance, in cases of non-continuous angles, a scenario arises where the angle increases continually and, when reaching 359, resets to 0 during the transition from the fourth quadrant to the first quadrant. On the contrary, in cases of continuous angles, the subsequent value after 359 would not reset to 0; instead, it would persistently increase, becoming 360, 361, 362, and so forth. To facilitate this

comparison, the "math.mod" function was employed to standardize the angles within a 0-360 degree range. In cases where the angles spanned different quadrants (specifically, the first and fourth quadrants), the simulation determined the number of full turns the previous angle had completed using the "math.floor" function. This count of full turns was then adjusted based on which quadrant each angle occupied. Finally, the current angle's contribution to the total degrees rotated thus far was added to ensure the continuity of angles throughout the simulation.

The ideal k_1, k_2, and k_3 values were chosen after running multiple tests experimenting with different values. The higher k values generally made swift adjustments to correct the error. However, due to the quick motion adjustments to the follower, the follower tended to flip over much more easily and more often and the motion tended to be not smooth. Therefore, k values were chosen to achieve a smoother motion for the follower while the error correction also happened with a reasonable accuracy. The definition of reasonable accuracy can vary depending on the specific scenario. For this project, an accuracy of ± 0.05 radians for angle error and ± 0.05 *m* distance error was considered reasonable. The optimum k values that worked best for this simulation were k_1=3, k_2=4, k_3=7.

6. Results

All components in place, the system can now be simulated to assess whether the follower can maintain the defined formation with the leader.

Gain k_1 only affects the velocity (v_{icon}) as shown in equation (25), and k_2 and k_3 affect the angular velocity (ω_{icon}) of the follower as shown in equation (26). First, several k_1 values were tested to see which values work best for the follower. During the testing both straight manoeuvring and making turns were considered. All tests shown below were also conducted for varied starting positions to ensure the consistency of results. The leader's velocity correction increases as the value of k_1 rises. At higher k_1 values, the follower accelerated and decelerated quickly making the follower's motion jerky and eventually making it flipping over. The highest k_1 value with an acceptable smooth motion was achieved when k_1=3. Acceptable smooth motion was outlined by any motion that avoided the occurrence of sudden, visible jerks. Therefore, most of the next tests were performed keeping k_1 value at 3 as a constant and changing k_2 and k_3 to optimize the response further. However, this definition is contingent on the specific situation. In instances where the robot is required to swiftly traverse from point A to point B in the shortest possible time, the introduction of some degree of jerk in the motion may be deemed acceptable, considering that the primary objective is to minimize the overall time of travel.

It's also viable to initiate the tuning process with a different k value, such as k2 or k3, and subsequently adjust the remaining two k values to optimize the system's response. Importantly, the optimal k values would persist despite the order of adjustment. This is attributed to the independence of the effects of k1 and the other two k values. Whether k1 is set initially or after establishing a value for k2 or k3, the adjustments still exert a comparable impact on the robot's velocity, ensuring that the overall pattern of the graph remains unchanged.

Shown below are some test results from several takes. The first test is for k_1=2, k_2=1, k_3=1. In the angle graph (Figure 10), the angles are measured in degrees (vertical axis) and the time is measured in seconds (Horizontal axis). In formation tracking error graphs, $e_{\Phi_{il}}$ is measured in radians (the red line), and $e_{l_{il}}$ is measured in meters (the blue line).

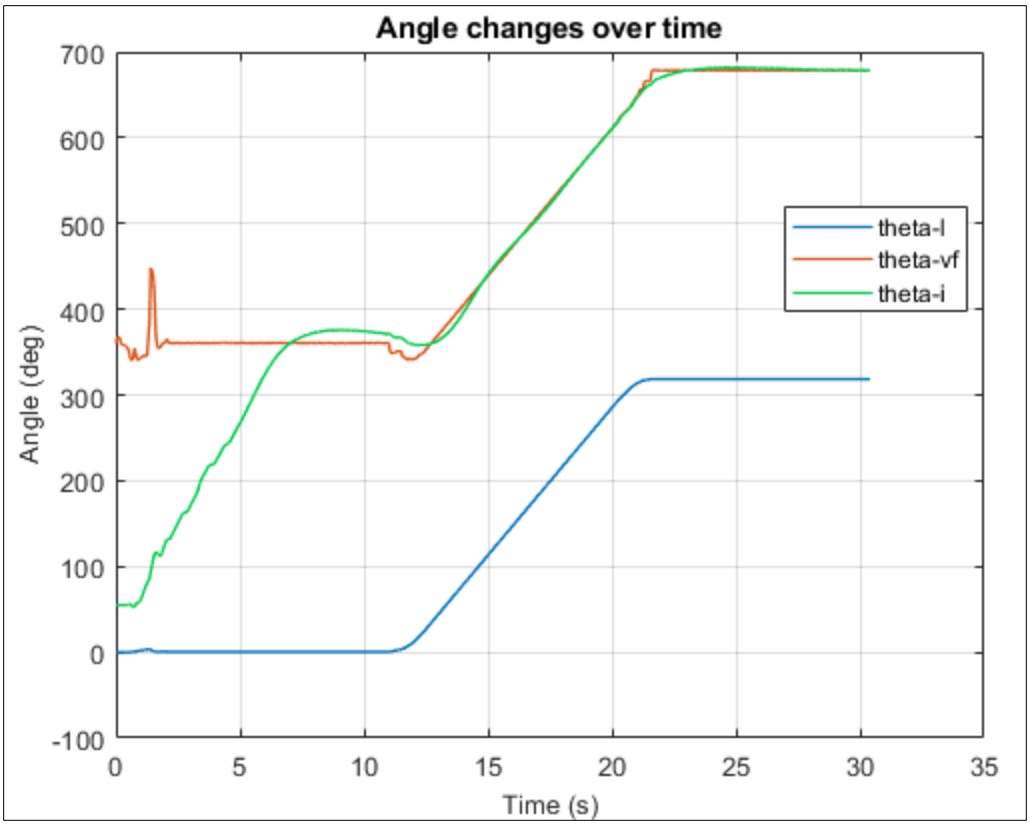

Figure 10: $\boldsymbol{\theta_l}$, $\boldsymbol{\theta_i^{vf}}$, $\boldsymbol{\theta_i}$ angle changes over time for test 1: All angles are measured in degrees

Figure 10 shows how angles θ_l, θ_i^{vf}, θ_i change over time. The blue line represents θ_l, the red line represents θ_i^{vf}, and the green line represents θ_i angles. Once the follower gets into the desired formation, the orientation of the follower (θ_i) follows the orientation of the virtual follower (θ_i^{vf}) closely. The virtual follower is required for the entire duration of operation as the controller for the follower is entirely designed based on the kinematics of the virtual follower. This is a good sign as the follower closely follows the orientation of the virtual follower to achieve the defined formation. However, it is not possible to conclude the formation is achieved since Figure 10 does not provide any data regarding the relative distance between the leader and the follower.

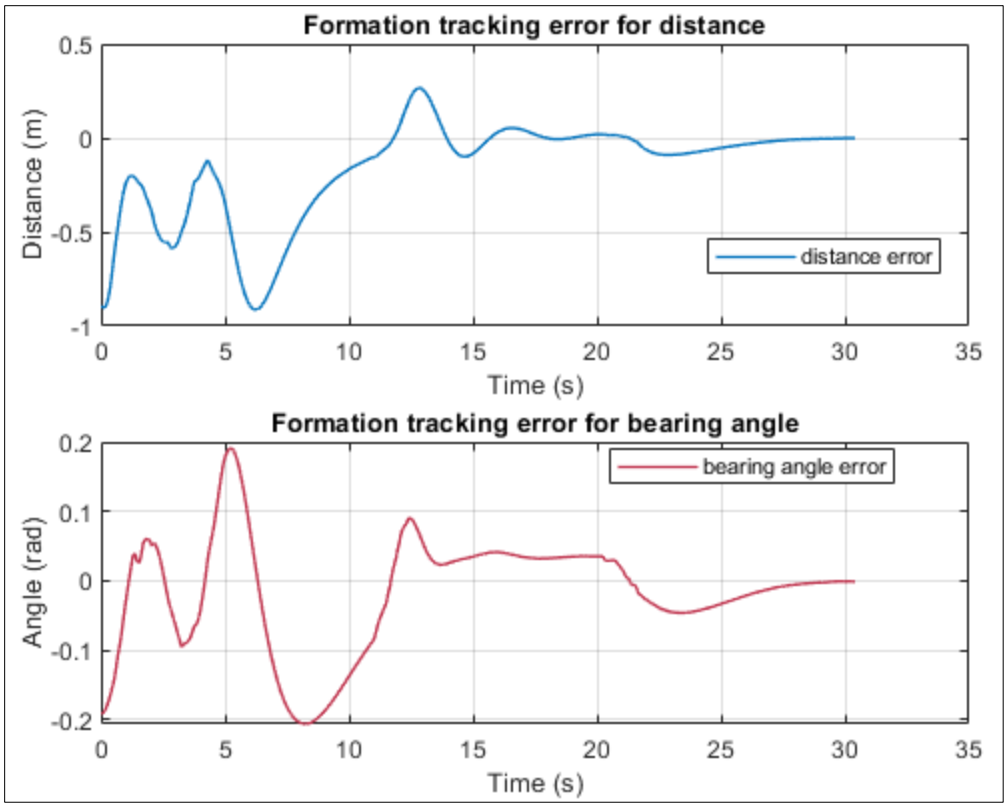

Figure 11: Formation tracking error changes over time for test 1: $\boldsymbol{e_{l_{il}}}$ is measured in meters (the blue line); $\boldsymbol{e_{\Phi_{il}}}$ is measured in radians (the red line)

In order to make sure the formation is preserved, formation tracking errors needs to be checked and they should be close to zero. Figure 11 displays the formation tracking errors $e_{l_{il}}$(distance error) and $e_{\Phi_{il}}$(bearing angle error) that were discussed in equation (13) in blue and red respectively. Figure 11 shows that the controller tries to drive both errors towards zero with time. This is apparent around 7-11 second section where θ_l is unchanged and again around 23-30 seconds. Additionally, immediately following a change in θ_l, there is a temporary increase in errors. However, the controller subsequently corrects these errors, driving them back towards zero. In general, the errors consistently remain below 0.25 throughout the entirety of this test, demonstrating a consistently small error range (except for the period from start to getting into formation where the system obviously starts with a huge error depending on the starting position). This is a favorable outcome for the initial test. The goal now is to take the control response in Figure 10 as a base and try to improve the response.

Two more tests (test 2 and 3) were done slightly increasing the k values for approximately the same θ_l profile in Figure 10. The k values tested for the test 2 were k_1=3, k_2=2, k_3=1 and for test 3 were k_1=3, k_2=4, k_3=1.

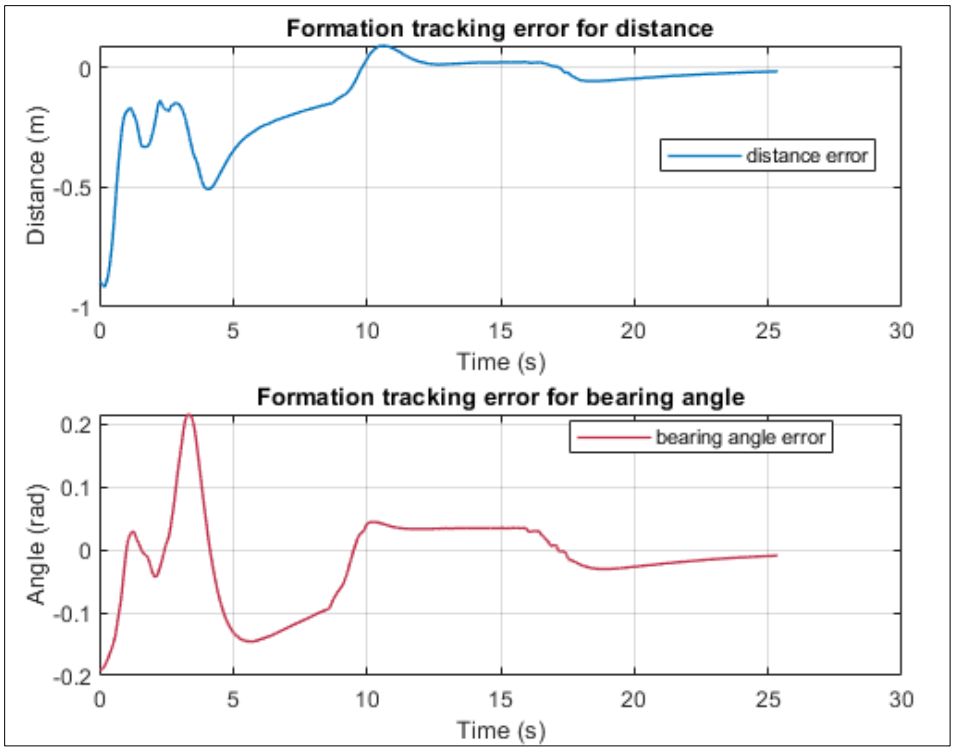

Figure 12: Formation tracking errors for test 2: $\boldsymbol{e_{l_{il}}}$ is measured in meters (the blue line); $\boldsymbol{e_{\Phi_{il}}}$ is measured in radians (the red line)

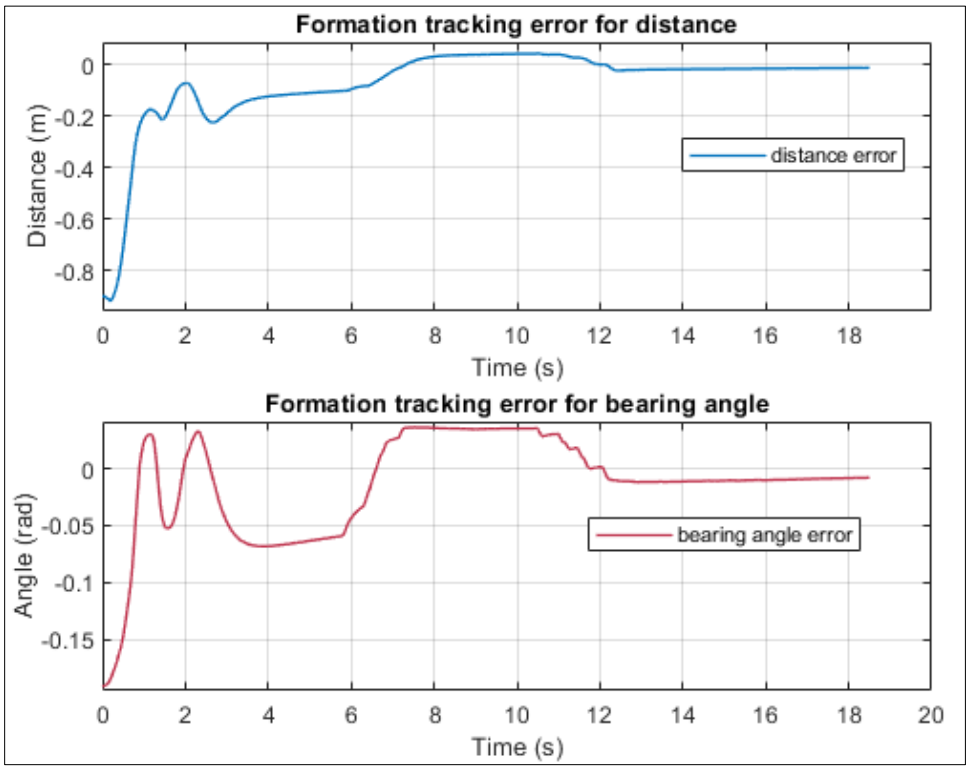

Figure 13: Formation tracking errors for test 3: $\boldsymbol{e_{l_{il}}}$ is measured in meters (the blue line); $\boldsymbol{e_{\Phi_{il}}}$ is measured in radians (the red line)

By comparing Figures 11 to 13, it is possible to notice that higher k_2 values decreases the oscillations, straights out the signal and improve the response. With the decrease in oscillations, the maximum error that happens due to oscillations is also reduced. With the first θ_l change that occurred around 11s in test 1, the maximum error recorded for $e_{l_{il}}$ was around 0.25 *m* at around 12.8s. The error was substantial, considering the desired distance was 2 *m*, and improvement was necessary. This number was reduced to approximately 0.1 *m* in test 2 and to 0.04 *m* in test 3.

The value of k_3 was then adjusted to see how it affects the response. Test 4 was done with k values k_1=3, k_2=4, k_3=4.

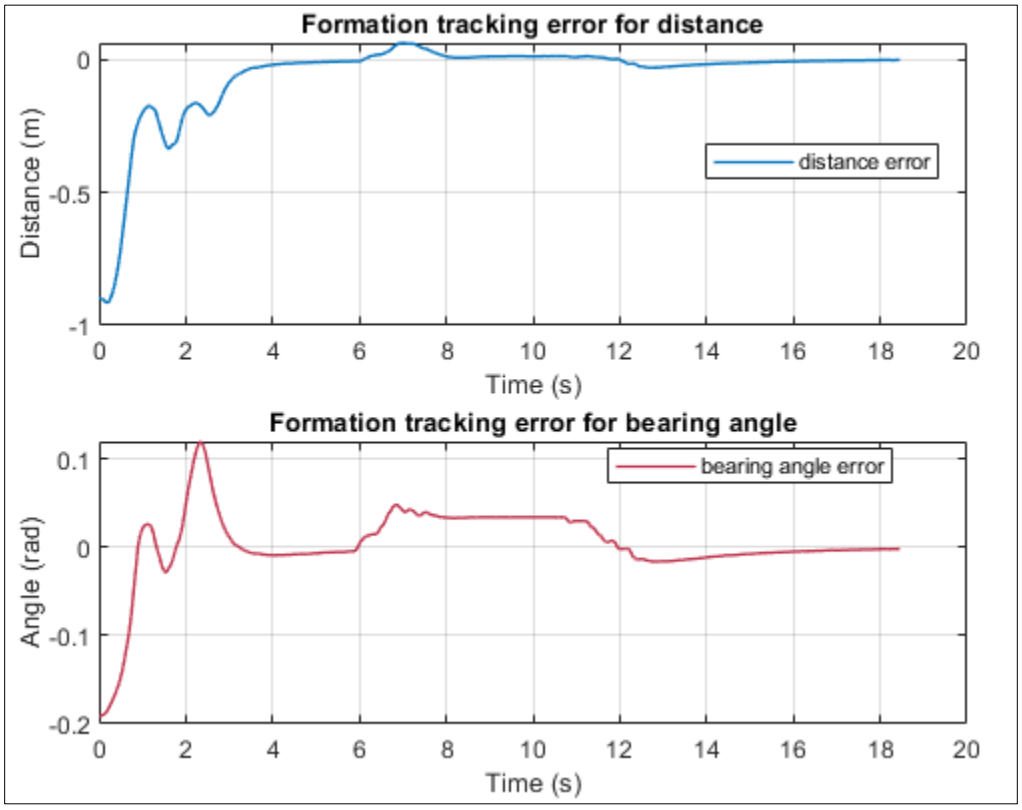

Figure 14: Formation tracking errors for test 4: $\boldsymbol{e_{l_{il}}}$ is measured in meters (the blue line); $\boldsymbol{e_{\Phi_{il}}}$ is measured in radians (the red line)

With the higher value of k_3, Figure 14 demonstrates a considerable improvement in error correction for $e_{l_{il}}$ compared to Figure 13. During 8s to 11s time duration in Figure 13, $e_{l_{il}}$ value is around 0.04 *m*, which is decreased to around 0.01 *m* during the same θ_l change as shown in Figure 14. This value becomes very close to zero in test 5, for which the results are shown in Figure 15. Test 5 was done for k_1=3, k_2=4, k_3=8 by further increasing the k_3.

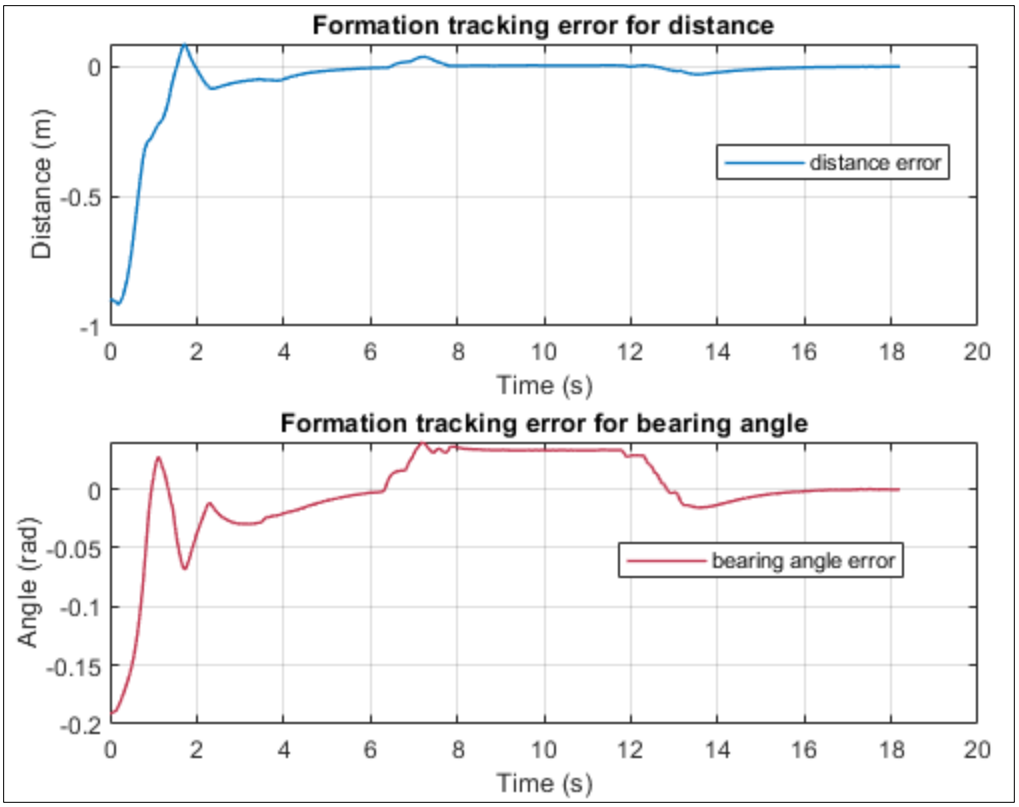

Figure 15: Formation tracking errors for test 5: $\boldsymbol{e_{l_{il}}}$ is measured in meters (the blue line); $\boldsymbol{e_{\Phi_{il}}}$ is measured in radians (the red line)

Upon comparing the results, it is evident that increasing the value of k_3 leads to improved error correction, driving the error signals closer to zero. It is possible to keep increasing the k values to decrease the error further; however, the higher error correction speed makes the follower's motion jerky and less stable leaving more room for the follower to flip over. Therefore, a balance should be achieved between the smooth motion and error correction when choosing k values. Figure 16 indicates a scenario where k values are very high (k_1=3, k_2=12, k_3=12) and the follower is not able to physically keep up with the quick error corrections provided to the follower. At this point the motion of follower becomes unstable, and the follower is not able to move properly to maintain the formation.

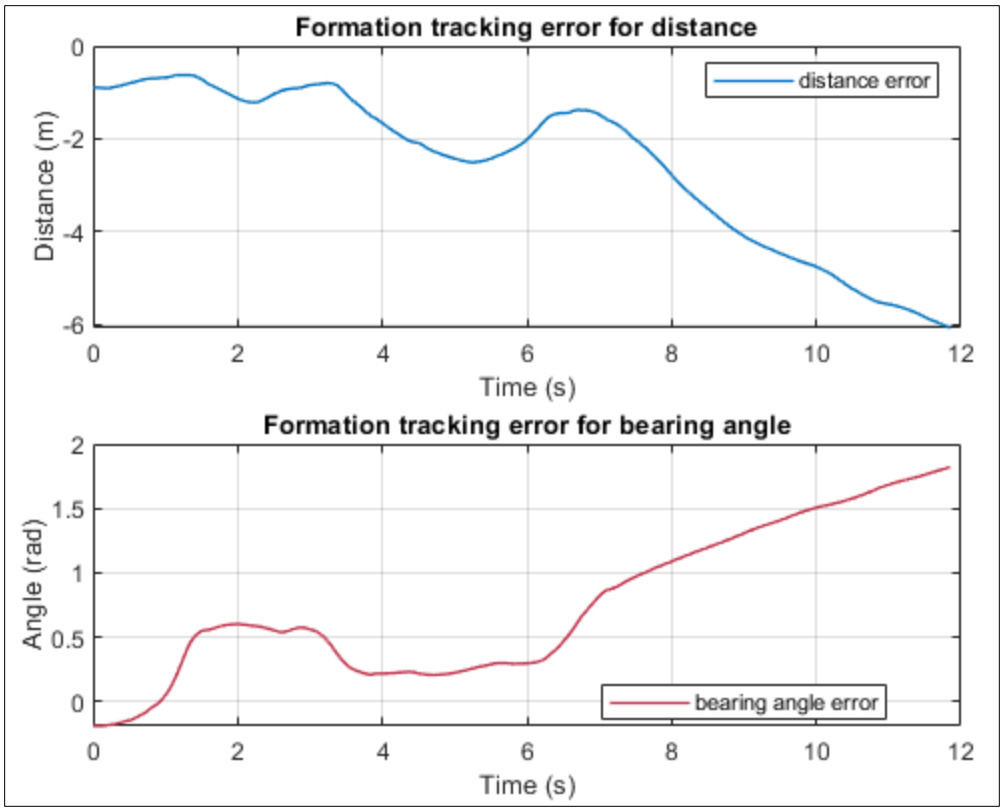

Figure 16: Unstable movement of the follower: $\boldsymbol{e}_{l_{il}}$ is measured in meters (the blue line); $\boldsymbol{e}_{\boldsymbol{\Phi}_{il}}$ is measured in radians (the red line)

After conducting numerous tests, it is concluded that the optimal k values that provide smooth maneuvering for the follower while also achieving an acceptable reduction in control error are: k_1=3, k_2=4, and k_3=7. The optimal response for these k values is shown in Figure 17 below.

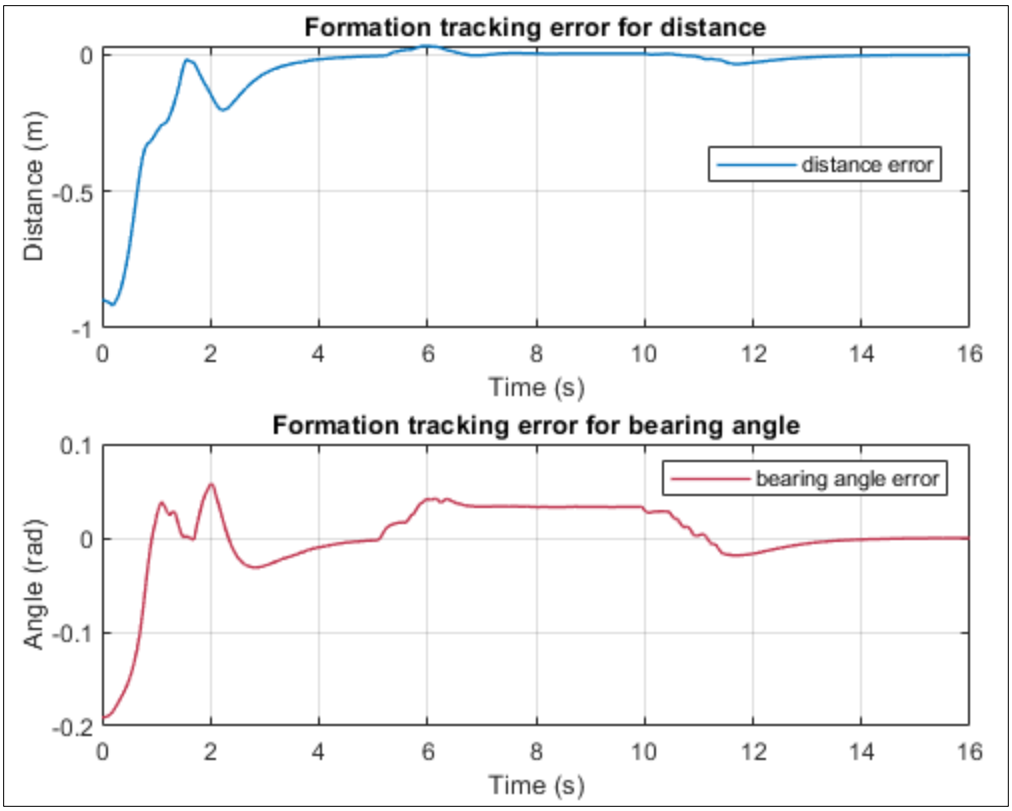

Figure 17: Optimal response: $e_{l_{il}}$ is measured in meters (the blue line); $e_{\Phi_{il}}$ is measured in radians (the red line)

It is technically possible to implement as many followers as necessary. For every follower, it is necessary to introduce a virtual follower configured with the desired formation corresponding to each individual follower. The only limitation to having many followers is the processing power of the simulation computer. As the number of followers increases, the demand for processing power also rises, leading to longer processing times. Additionally, as the number of followers within the system grows, the likelihood of collisions among them also rises. Hence, it becomes imperative to implement measures aimed at preventing robot collisions. Having achieved success in the design and simulation of a leader-follower formation controller, the next logical step is to seamlessly transition from the virtual environment to the real world. The embedded control technique, proven effective in simulations, is now poised to be applied to actual Wheeled Mobile Robots (WMRs). This marks a significant advancement, as the developed control systems can be implemented in real-world scenarios, enabling the design of leader-follower control systems tailored for a diverse range of tasks. This practical application promises to bring the theoretical achievements into tangible use, showcasing the adaptability and robustness of the control methodology in addressing various challenges and tasks encountered by WMRs in real-world environments.

7. Conclusions

7.1 Conclusion

The goal of this project was to design a controller for the leader-follower formation control problem and simulate it in CoppeliaSim. Two differential drive wheel robots were used for this simulation, one as the leader and the other as the follower. Embedded control technique was used for the controller design instead of a conventional formation control design philosophy in which the task is divided into two subtasks which are virtual signal generator and the agent tracking controller. This technique offers three main advantages over a conventional formation control design, namely simplified modular design, plug and play function, and application-oriented feature. Simplified modular design makes the implementation simpler compared to direct formation tracking method. Plug and play function provides a convenient way for a WMR to join a new task while the application-oriented feature lessens the challenges brought on by WMRs' encapsulated properties. The simulation was performed in CoppeliaSim, and the follower closely followed the leader successfully. Several tests were done adjusting k values to find the optimal k values. All in all, the simulation provided satisfactory results and the leader-follower formation controller was successfully programmed in CoppeliaSim. The embedded control method used in this project has some limitations as well. The primary limitation is that this controller is suitable only for scenarios where the leader is constantly in motion, and the equations become invalid when the leader comes to a stop. As long as the leader remains in motion, the followers will continue to move without coming to a stop. Another limitation is that this method does not accommodate two specific formations when the desired angle Φ_{il}^{*} equals $\pi/2$ and $3\pi/2$.

7.2 Future Works

For future work, several ideas can be implemented to improve the functionality of this controller. A haptic device can be used to control the leader robot through teleportation instead of the CoppeliaSim UI window that is currently being used for the project. Further research is needed on the functions available in CoppeliaSim for connecting it to a haptic device in order to establish a connection between the software and the haptic device. An additional improvement that can be made is the implementation of a collision avoidance algorithm to prevent robots from bumping into each other while forming the desired configuration. Such an algorithm can be very useful as the number of followers increases because the likelihood of collision increases with the number of robots in the system. CoppeliaSim provides a wide range of useful sensors that can be used for collision avoidance.

8. References

[1] - L. Consolini, F. Morbidi, D. Prattichizzo, and M. Tosques, “Leader–Follower Formation Control of Nonholonomic Mobile Robots With Input Constraints”,in *Automatica*, vol. 44, no. 5, pp. 1343–1349, 2008, doi: 10.1016/j.automatica.2007.09.019

[2] - J. Hirata-Acosta, J. Pliego-Jiménez, C. Cruz-Hernádez, and R. Martínez-Clark, “Leader-Follower Formation Control of Wheeled Mobile Robots without Attitude Measurements”,in *Applied Sciences*, vol. 11, no. 12, 2021, doi: 10.3390/app11125639.

[3] - W. Liu, X. Wang and S. Li, "Formation Control for Leader–Follower Wheeled Mobile Robots Based on Embedded Control Technique," in *IEEE Transactions on Control Systems Technology*, vol. 31, no. 1, pp. 265-280, Jan. 2023, doi: 10.1109/TCST.2022.3173887.

[4] - S. G. Tzafestas, “6 - Mobile Robot Control II: Affine Systems and Invariant Manifold Methods”, in *Introduction to Mobile Robot Control*, S. G. Tzafestas, Ed. Oxford: Elsevier, 2014, pp. 185–235, doi: 10.1016/B978-0-12-417049-0.00006-7

[5] - A.V. Chavan and Dr. J. L. Minase, “Design of a Differential Drive Mobile Robot Platform for Use in Constrained Environments”,in *International Journal of Innovations in Engineering Research and Technology*, vol. 2, no. 6, pp. 1–10, Mar. 2021.

[6] - G. Antonelli, S. Chiaverini and G. Fusco, "A Calibration Method for Odometry of Mobile Robots Based on The Least-Squares Technique: Theory and Experimental Validation," in *IEEE Transactions on Robotics*, vol. 21, no. 5, pp. 994-1004, Oct. 2005, doi: 10.1109/TRO.2005.851382.

[7] - A. Filipescu, V. Minzu, B. Dumitrascu, A. Filipescu and E. Minca, "Trajectory-Tracking and Discrete-Time Sliding-Mode Control of Wheeled Mobile Robots," *2011 IEEE International Conference on Information and Automation*, Shenzhen, China, 2011, pp. 27-32, doi: 10.1109/ICINFA.2011.5948958.

[8] - “Pioneer 3-DX,” generationrobots.com. https://www.generationrobots.com/media/Pioneer3DX-P3DX-RevA.pdf (accessed Dec. 13, 2023).

[9] - “CustomUI Plugin - UI XML Syntax,” coppeliarobotics.com. https://www.coppeliarobotics.com/helpFiles/en/simUI-widgets.htm (accessed Dec. 13, 2023).

[10] - T. Balch and R. C. Arkin, "Behavior-Based Formation Control for Multirobot Teams," in *IEEE Transactions on Robotics and Automation*, vol. 14, no. 6, pp. 926-939, Dec. 1998, doi: 10.1109/70.736776.

[11] - R. W. Beard, J. Lawton and F. Y. Hadaegh, "A Feedback Architecture for Formation Control," *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No.00CH36334)*, Chicago, IL, USA, 2000, pp. 4087-4091 vol.6, doi: 10.1109/ACC.2000.876990.

[12] - M. A. Lewis and K.-H. Tan, “High Precision Formation Control of Mobile Robots Using Virtual Structures”,in *Autonomous Robots*, vol. 4, no. 4, pp. 387–403, Oct. 1997, doi: 10.1023/A:1008814708459.

[13] - D. Sun, C. Wang, W. Shang and G. Feng, "A Synchronization Approach to Trajectory Tracking of Multiple Mobile Robots While Maintaining Time-Varying Formations," in *IEEE Transactions on Robotics*, vol. 25, no. 5, pp. 1074-1086, Oct. 2009, doi: 10.1109/TRO.2009.2027384.

[14] - M. A. Kamel, X. Yu, and Y. Zhang, “Formation Control and Coordination of Multiple Unmanned Ground Vehicles in Normal and Faulty Situations: A review”,in *Annual Reviews in Control*, vol. 49, pp. 128–144, 2020, doi: 10.1016/j.arcontrol.2020.02.001.

[15] - I. Sanchez, A. D’Jorge, A. Ferramosca, G. Raffo and A. H. Gonzlez, "Path Following and Trajectory Tracking Model Predictive Control using Artificial Variables for Constrained Vehicles," *2019 XVIII Workshop on Information Processing and Control (RPIC)*, Salvador, Brazil, 2019, pp. 198-203, doi: 10.1109/RPIC.2019.8882189.