# Multi-Agent Source Seeking in Unknown Environments

## A hybrid adaptive feedback approach for unicycles

M. J. van der Linden

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

# Multi-Agent Source Seeking in Unknown Environments

## A hybrid adaptive feedback approach for unicycles

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

M. J. van der Linden

December 29, 2020

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology


Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Multi-Agent Source Seeking in Unknown Environments

by

M. J. van der Linden

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: December 29, 2020

Supervisor(s):

ir. S. Krilašević

dr. ir. S. Grammatico

Reader(s):

dr. ir. L. Ferranti

# Abstract

The application of autonomous robotic vehicles to explore unknown environments is a growing field of interest in the research community. This thesis report studies the practical implementation of employing three autonomous vehicles to navigate towards a signal emitting source in an unknown environment. The group of autonomous vehicles is assigned to drive towards the source in a coordinated formation shape whilst avoiding an obstacle. Therefore, a control algorithm is developed that is required to be suitable for a non-holonomic unicycle. As an experimental platform for the algorithm, the Husarion ROSbot is used. To evaluate the performance of the algorithm, simulations are run in a Gazebo simulator environment. To achieve the stated research objective, the problem is broken down into three sub-objectives: source seeking, obstacle avoidance, and formation control. First, two types of source seeking approaches for unicycles are implemented to investigate how each approach affects the source seeking performance. Thereafter, an obstacle avoidance algorithm is developed to combine with both source seeking approaches. [19] is used as a framework to create a hybrid adaptive feedback (HAF) law to avoid an obstacle, with the location and orientation of the obstacle w.r.t. the source is not assumed to be known *a priori*. Subsequently, two follower vehicles are introduced to study how to enable a group of vehicles to drive in a coordinated formation to the source while avoiding an obstacle. Finally, the algorithm developed in this thesis is empirically shown in simulations not to suffer from convergence to a detrimental line $M$ that would occur with an artificial potential function approach, as described by [24]. The follower vehicles employ a decentralized leader-follower strategy to maintain formation shape and successfully avoid an obstacle.

# Table of Contents

# List of Figures

# List of Tables

# Acknowledgements

This report will describe my final thesis work to receive my Master of Science degree in Systems & Control Engineering at the Delft University of Technology.

First of all, I would like to thank my supervisor ir. Suad Krilašević for his assistance throughout my thesis. I learned a lot from his feedback and interesting insights during our meetings. Also, I would like to thank dr. ir. Sergio Grammatico, for his guidance and feedback during my thesis.

Lastly, I would like to thank my friends and family for their support. Without you, the final result not have been the same.

Delft, University of Technology
December 29, 2020

M. J. van der Linden

# Chapter 1

# Introduction

This chapter will introduce the topic of this thesis research into *multi-agent source seeking in unknown environments*. First, the motivation for conducting this thesis research is explained. Combined with a discussion on research related to this thesis. Thereafter, the research objectives for this thesis are presented. Followed by outlining the approach to achieve the research objectives and a discussion on the challenges that arise. Lastly, a brief overview is given on the structure of this thesis report.

## Motivation

The deployment of autonomous robotic vehicles to explore unknown environments has been a growing field of research, as stated by [3], [16], and [32]. The use of autonomous robots for scientific or disaster response missions is especially promising when sending a human operator is undesirable due to the mission being too dangerous or burdensome [13]. A possible task for a robotic vehicle, as shown in Figure 1-1, in such a mission could be to locate a source that emits a signal that attenuates over distance w.r.t. the source. These signals could be chemical, biological, or thermal in nature and generated by physical processes, e.g., such as gas leakage. For these unknown environments, position information is often unavailable due to the absence of GPS. Control algorithms capable of dealing with these types of source seeking problems have been applied to a wide variety of engineering applications [32].

Also, unknown environments could be cluttered with obstacles, e.g., furniture or equipment, if the robotic vehicle is exploring an industrial plant. Therefore, including the robot's ability to avoid obstacles to the control algorithm is interesting because it would expand the applicability of the robot also to perform source seeking in cluttered environments, as noted by [9].

Besides navigating one robotic vehicle in a cluttered environment towards a source, designing an algorithm to enable a group of autonomous robots to perform these tasks could increase the mission's value. Because using a group of robotic vehicles in a coordinated formation to navigate towards a source, more complex tasks could be solved. Thereby acquiring an


**Figure 1-1:** The husky platform participating in DARPA's SubTerranean Challenge. The contest demonstrates the requirement of improving safety and efficiency for underground search-and-rescue missions, from [23]

improved understanding of the unknown environment, as stated by [14]. E.g., other vehicles in the formation could be tasked to obtain measurement samples of the environment whilst one vehicle is navigating towards the source.

## Research Objectives

Based on these observations, it is valuable to develop an algorithm capable of being deployed on a group of robotic vehicles and successfully navigate in such an environment towards an unknown source. Therefore, the main research objective of this M.Sc. thesis will be the following:

**Main Research Objective**

Develop and implement a control algorithm capable of steering a group of three autonomous unicycle vehicles towards an unknown source while avoiding an obstacle that interferes with its path and without inter-vehicle collisions. The combined control algorithm should be suitable to deploy on the ROSbot autonomous robot platform.

To satisfy the main research objective, the problem is broken down into multiple sub-objectives. For each of these sub-objectives, a different algorithm is studied to achieve the objective and if it is suitable to incorporate with the other sub-objectives. The sub-objectives are the following:

1. Implement a source seeking algorithm capable of finding an unknown source for a unicycle vehicle with control input limitations.
2. Develop an obstacle avoidance method suitable to combine with a source seeking unicycle, without *a priori* knowledge on the obstacle's position and orientation w.r.t. the source.
3. Implement a decentralized formation control architecture for unicycles to maintain formation shape, prevent inter-vehicle collisions, and avoid obstacles.

## Research Approach

Each of the above-listed sub-objectives poses a different challenge that this thesis research will try to overcome. The source-seeking problem assumes that the robotic vehicle can only acquire the signal emitting source's scalar signal strength locally. Because generally, it is not possible to measure the gradient of a signal field, and thus the gradient needs to be estimated based on local measurements only. To seek the source based on local scalar signals, an extremum seeking method is employed. By using extremum seeking, the gradient is estimated by moving the vehicle through space. However, because of the kinematic constraints on the autonomous ROSbot platform, only algorithms suitable for unicycles are studied. Designing a source seeking algorithm for a unicycle is challenging because it cannot directly be moved sideways and only be controlled by the angular and forward velocity, as stated by [33].

One challenge that arises when using source seeking control to explore unknown environments is the addition of obstacle avoidance to the control objectives. A common method to guide a robotic vehicle around obstacles is to augment the signal the source seeking control law is using to find the source by a potential function. However, as stated by [5], [19], and [25], obstacles that interfere with the vehicle's trajectory when using a smooth source seeking control law preclude robust stabilization of the algorithm. Because of the topological obstructions induced by the obstacle, it could lead the source seeking vehicle to prematurely assume it has reached the source from a certain set of initial conditions or hit the obstacle. In [19], a hybrid adaptive feedback (HAF) law is presented that claims to overcome this problem. Therefore, it will be used as a framework for the obstacle avoidance approach in this thesis research.

The last sub-objective is to navigate a group of vehicles in a coordinated way to the unknown source. Because the addition of robotic vehicles could allow for more complex tasks while exploring the unknown environment. When creating such a multi-robot formation algorithm, the control architecture poses a significant challenge. In this thesis, a decentralized leader-follower architecture is chosen similar to the approach presented in [30]. The approach allows each follower robot to compute their desired control input solely on their locally obtained measurements and does not require communication between vehicles.

As a framework, this thesis research uses the work presented in [19] to study the practical implementation of a HAF law on a group of unicycle vehicles performing source seeking. Other than in [19], where a HAF algorithm is employed on a two-dimensional point mass, or in [15] where the HAF algorithm is used on a UAV modeled with single integrator dynamics. No research has yet been published on the applicability of a HAF approach to non-holonomic unicycles. Because a significant share of robotic vehicles are modeled as non-holonomic unicycle, the HAF approach could offer an interesting solution to the problem described in [25]. Furthermore, studying how to implement an algorithm suitable for navigating multiple vehicles in a coordinated formation towards a source would add value since the authors in [19] assume that the leader vehicle in the formation avoids the obstacle by a distance larger than the formation size. Hence, the follower vehicles do not need to apply control effort for obstacle avoidance but only need to maintain formation shape. However, if the algorithm were to be deployed on an actual robot, it would be beneficial if the formation shape is flexible. The leader vehicle could then avoid the obstacle by a smaller distance resulting in a shorter path towards the source. Therefore, this thesis research would make a valuable contribution based on reviewed literature following the main research objective.

## Thesis Structure

This thesis report is structured as follows. First, the ROSbot platform is introduced in Chapter 2, on which the control algorithms are implemented and tested. Thereafter, in Chapter 3, the source seeking approach will be explained. The obstacle avoidance method to be incorporated with the source seeking algorithm is elaborated on in Chapter 4. Following Chapter 5, the decentralized formation control algorithm is described, which includes source seeking and obstacle avoidance algorithm. In Chapters 3, 4, and 5 each time the newly included algorithm is tested in simulation, and their respective results are shown and discussed. Finally, in Chapter 6, conclusions are drawn on the overall performance of the developed control algorithm for this thesis, research and recommendations are given for further research.

# Chapter 2

# ROSbot Platform

As an experimental platform for the control algorithms in Chapters 3, 4, and 5, the Husarion ROSbot is used [10]. First, in section 2-1, a general introduction will be given to the Husarion ROSbot. Thereafter, in section 2-2, the robot's kinematic model is introduced that the control algorithm will have to consider. See section 2-3 for the hardware specifications and section 2-4 for an elaboration on the software the ROSbot uses. The experiments performed for this thesis research are conducted in a Gazebo simulator environment, further described in section 2-4-1.

## 2-1 Robot Description

The Husarion ROSbot is a wheeled mobile robot, as shown in Figure 2-2, which is utilized at the laboratory of the Delft Center for Systems and Control (DCSC) at TU Delft as an experimental platform. The ROSbot is an autonomous robot platform based on a Husarion CORE2-ROS controller, which is suitable to develop custom algorithms for designing, implementing, testing, and validating autonomous driving tasks, behaviors, and maneuvers as stated by [4]. Unfortunately, due to the global pandemic, no physical experiments will be performed in the laboratory. Only simulations are conducted using the Gazebo simulator environment elaborated on in section 2-4-1. In this thesis research, three ROSbots will be utilized to conduct experiments on the developed algorithm.

## 2-2 Kinematic Model

Each ROSbot possesses four wheels, where each wheel is equipped with a separate drive. However, to simplify kinematic computations, the ROSbot is modeled as a two-wheeled robot. Therefore the ROSbot can be modeled as a unicycle with two virtual wheels on an axis through the robot's geometric center, see Figure 2-1. The robot's heading is controlled by changing the relative rate of rotation of its wheels and does not need extra steering. Therefore, the

ROSbot is considered a skid-steer drive robotic vehicle.

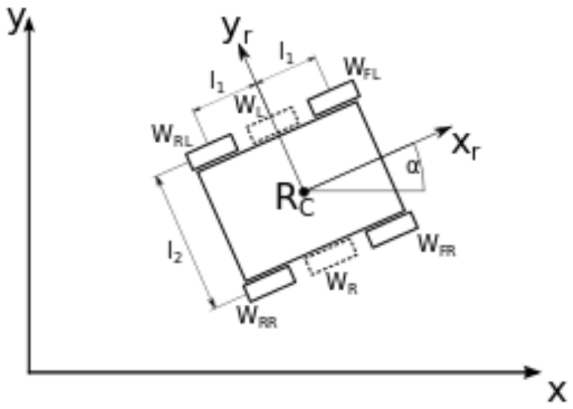



**Figure 2-1:** The simplified kinematic model of the ROSbot with two virtual wheels, from [10].

The ROSbot can only drive in the $x - y$ plane and has kinematic constraints with 3 degrees of freedom (DOF). The robot cannot move sideways. Therefore not all DOF are controllable. Thus, the ROSbot vehicle is considered to be a non-holonomic system. The forward velocity and angular velocity can be manipulated to control the direction of the ROSbot in the $x - y$ plane.

## 2-3 Hardware Specifications

For the development of the control algorithm for this thesis report, the robot's sensors should be taken into account to know what information the robot can acquire on its condition and environment. The algorithm must also be developed to run on the ROSbot's single-board computer (SBC). The ROSbot platform is equipped with the following hardware, listed in Table 2-2 and shown in Figure 2-2. The dimensions of the ROSbot with a camera and Light Detection And Ranging (LiDAR) sensor mounted are $200 \times 235 \times 220$ mm $[L \times W \times H]$, as stated in [10]. The RGBD camera and LiDAR are linked to the SBC, whereas the other sensors listed in Table 2-2 are connected to the CORE2-ROS controller. Since the developed algorithm in this research only utilizes the LiDAR, inertial measurement unit (IMU), and the quadratic encoders on the DC motors, the working principles of these components will briefly be explained. The LiDAR mounted on the ROSbot is used to detect nearby obstacles and fellow ROSbots of the formation using the laser triangulation principle to scan its environment and generate 2D point cloud data, which is further explained in section 4-2-1. The detection range of the LiDAR has a minimum distance of 15 $cm$ and a maximum of 8 $m$. The IMU sensor acquires the roll, pitch, yaw, and acceleration of the ROSbot. To measure each wheel's angular rotation, the quadratic encoders are used, which have a resolution of 48 $ppr$ (or 7.5 $degrees$).

**Table 2-1:** Table with all Hardware components on the Husarion ROSbot platform

| Amount | Component | Description |
|---|---|---|
| 1 | ASUS Tinker Board (SBC) | with Husarion CORE2-ROS version |
| 1 | RPLIDAR A2 360° | Laser Scanner |
| 1 | MPU 9250 Inertial Sensor (IMU) | accelerometer + gyro |
| 1 | RGBD camera Orbbec Astra | camera |
| 4 | VL53L0X Time-of-Flight | (Laser) Distance Sensors |
| 4 | DC motors | with quadratic encoders |

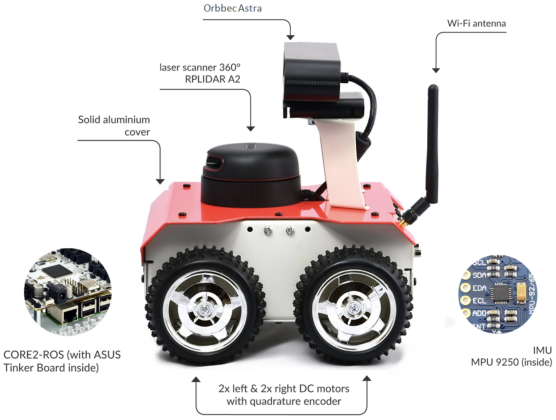



**Figure 2-2:** Hardware components of the Husarion ROSbot, from [10]

## 2-4 Software

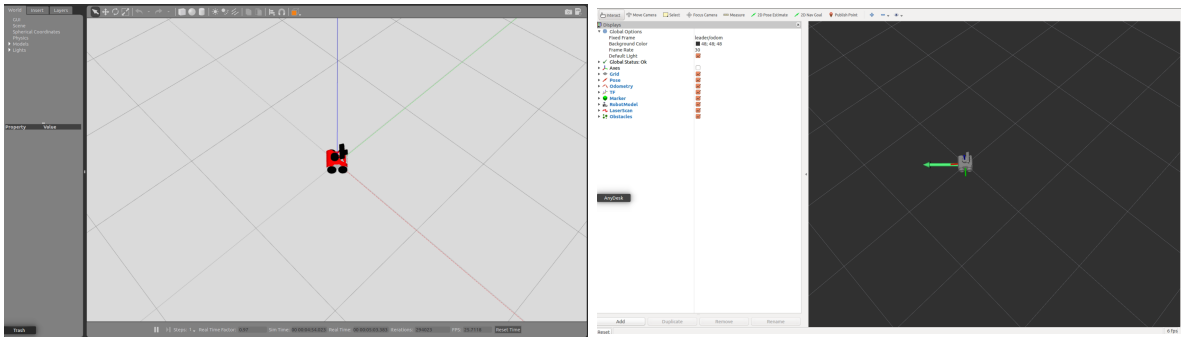
The software used for the ROSbot platform can be categorized into two sections, the low-level firmware and the operating system (OS) based on Ubuntu 16.04, where the low-level firmware runs on the real-time Husarion CORE2-ROS controller and the OS on the SBC that works with Robot Operating System (ROS). The four DC motors powering each wheel are controlled by the CORE2-ROS controller using libraries developed by Husarion. The algorithm developed in this thesis will run as higher-level code on the SBC. The control algorithm can "set" values on the low-firmware for the CORE2-ROS controller to read, e.g., "set" forward velocity to 0.3 $m/s$.

The control algorithm for this thesis will be programmed using the structured communication method of ROS. ROS is an open-source operating system; however, it is different from most operating systems because it does not specify process management and scheduling. Instead, ROS produces a structured communication method between a heterogeneous computer cluster's host operating system, as described by [22].

### 2-4-1 Gazebo Simulator

As a simulator environment for the control algorithms in this thesis, Gazebo is used. Gazebo is an open-source robotics simulator environment suitable for ROS. To simulate the ROSbot in Gazebo, gazebo model supplied by Husarion is used [11]. The gazebo model uses the Open

Dynamics Engine as a physics engine to simulate the robot dynamics. The sensors listed in Table 2-1 can be used in simulations, and noise can be added to the generated sensor measurements to create realistic simulations. RViz is used as a graphical interface tool to visualize the information acquired by the sensors in simulation. In Figure 2-3, the simulator environment is shown, wherein the Figure 2-3a, the gazebo simulator is displayed, and the corresponding RViz interface in Figure 2-3b. The simulations are all run on a regular laptop equipped with an Intel Core i7-8550U, 4GHz processor, and 8 GB of RAM.



**(a)** Gazebo environment **(b)** RViz graphical interface

**Figure 2-3:** Simulator environment.

# Chapter 3

# Discrete Unicycle Source Seeking

As described in the introduction, see Chapter 1, one of the main objectives of this thesis research is to develop an algorithm for navigating an autonomous robotic vehicle towards an unknown source. An extremum seeking control approach is chosen to deal with this problem, which is an adaptive control method. Unlike classical adaptive control methods, extremum seeking does not manage stabilization of known reference trajectories or set-points. Extremum seeking differs from classical adaptive control methods in being a model-free approach, thereby allowing it to tackle problems with unreliable or complex models [1]. One common application of extremum seeking is navigating robotic vehicles towards the extremum of an objective function, as has been published in [13], [19], [33]. Extremum seeking applied to these types of navigation problems is also considered in the literature as source seeking [32]. First, in section 3-1, the basic idea of source seeking is introduced. Thereafter, in section 3-1-1 it is explained how the concept of extremum seeking is applied to be suitable for navigating the ROSbot towards a source. Subsequently, the source-seeking algorithm's implementation approach is explained in section 3-2. For simulation results of the developed source seeking algorithm, the reader is referred to section 3-3. Finally, in section 3-4 the performance of the implemented source seeking algorithms are discussed.

## 3-1 Problem Statement

The main goal of a source seeking algorithm, which as defined by [8], is to determine and to maintain the extremum output value $y$ of a single unknown objective function $f(x, u)$ like shown for the system given in Eq. (3-1.1).

$$\begin{aligned} \dot{x} &= f(x, u) \\ y &= h(x) \end{aligned} \tag{3-1.1}$$

Where $x \in \mathbb{R}^n$ is the measurable state, $u \in \mathbb{R}$ is the system input and $y \in \mathbb{R}$ is the output. Throughout this thesis research, the problem is considered to be seeking the source of the

scalar signal. The signal strength is assumed to be decaying away from the source and the functions $f : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ and $h : \mathbb{R}^n \to \mathbb{R}$ are assumed to be smooth. However, the shape of the signal field is not known to the source seeking vehicle. The vehicle can only obtain the signal's scalar value by measuring it at the vehicle's current location. Therefore, only model-free gradient estimation source seeking methods are suitable for this thesis research.

From the variety of model-free source seeking methods, the sinusoidal perturbation based source seeking approach is chosen. Because it can easily be adapted to suit a multi-dimensional input system, like the ROSbot platform see section 2-2. By adding a sine wave to each input channel, the sinusoidal perturbation approach estimates the gradient of the signal field by moving the vehicle through space. In section 3-1-1, a more detailed explanation on how the source seeking controller is employed on the ROSbot vehicle will be given.

### 3-1-1 Non Holonomic Source Seeking

Employing a unicycle vehicle, like the Husarion ROSbot, to move in a particular manner to allow for gradient estimation is much more difficult to achieve than with a point mass due to the kinematic constraints on the vehicle, as stated by [33]. However, it is still possible to create a stable source seeking algorithm to navigate the robot towards the source, as will be shown in this section.

Control algorithms suitable for a unicycle vehicle using sinusoidal perturbation based source seeking can be categorized into three strategies. The first one is a forward velocity tuning approach presented by [33]. The algorithm of [33] applies extremum seeking to tune the vehicle's forward velocity while keeping the angular velocity constant. The approach of [33] produces a periodic forward-backward motion while the unicycle is navigating towards the source. The second approach, published by [3], uses extremum seeking to tune the vehicle's angular velocity and keeps the forward velocity constant. By keeping the forward velocity constant, the algorithm becomes suitable for unicycles where forward-backward motion is not feasible, e.g., fixed-wing aircraft. The strategy of [3] generates a trajectory of the unicycle, which sinusoidally converges towards the source and settles to a ring around the source. The third method presented by [7], employs extremum seeking to tune both the angular velocity ($\Omega$) directly and the forward velocity ($v$) indirectly. The velocity inputs are tuned by applying sinusoidal perturbation-based sources seeking directly on the angular speed and adding simple derivative-like feedback to a constant forward velocity. Using this approach, according to the authors in [7], the best features from [3] and [33] are combined. In [6], this resulted in the vehicle to slow down when approaching the source and converge to a closer proximity w.r.t. the source without diminishing the convergence speed. Therefore, the unicycle source seeking method of [7] is further studied in this thesis research because the strategy appears to exhibit the best performance for mobile unicycle robots like the Husarion ROSbot.

As described in section 2-2, the Husarion ROSbot is modeled as a unicycle. The equations of motion corresponding to the vehicle's geometric center are shown in Eq. (3-1.2).

$$\dot{r}_c = v e^{j\theta} \tag{3-1.2a}$$

$$\dot{\theta} = \Omega \tag{3-1.2b}$$

The ROSbot's geometric center is expressed as a complex variable $r_c$ in $2D$, $v$ and $\Omega$ are the forward and angular velocity inputs and $\theta$ is the ROSbot's orientation. In Figure 3-1, the vehicle's center and sensor are displayed with the vehicle's position, heading, forward and angular velocities. The vehicle's sensor is used to measure the scalar signal strength. The sensor's location can either be placed at a $R$ distance away from the vehicle's center or collocated with the center, as shown in Eq. (3-1.3).

$$r_s = r_c + Re^{j\theta} \tag{3-1.3}$$

The sensor will be placed at a $R > 0$ distance to improve the source-seeking algorithm's convergence rate, as noted in [3]. The authors in [3] demonstrate that by placing the sensor at a $R > 0$ compared to $R = 0$ distance from the vehicle's center, the trajectory towards the source is improved. Because mounting the sensor off-center on the vehicle will cause the sensor to "sweep" the signal field. Thereby requiring less movement of the vehicle itself to approximate the signal gradient, resulting in improved performance. When the sensor is collocated with the vehicle's center, the vehicle must perform much sharper turns. Hence, using the whole vehicle to measure the signal field and not just the vehicle's outer end.

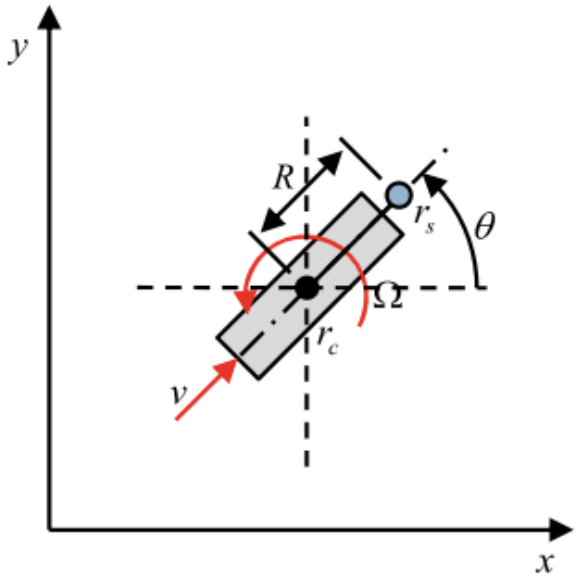



**Figure 3-1:** The model of the ROSbot's center dynamics and non-collocated sensor, from [7].

The signal field that the source is producing is assumed to have the following properties. The distribution of signal field is presumed to be an unknown nonlinear map $J = f(r(x, y))$, with an isolated local maximum at $f^* = f(r^*)$ and $r^*$ being the location of the source. The objective of the source seeking algorithm is to autonomously find the local maximum at $r^*$, where the signal field is assumed to satisfy Assumption 3-1.1. The authors in [19] assume the same condition as stated in Assumption 3-1.1 for the signal field to study perturbation based source seeking. The sensor measures the signal strength, so $f(r_s)$.

**Assumption 3-1.1.** *The function $J(x, y) : \mathbb{R}^2 \to \mathbb{R}$ is smooth with a strict global maximum $[x^*, y^*]^T \in \mathbb{R}^2$. Furthermore, for every $\gamma \in \mathbb{R}$ the set $\{(x, y) : J(x, y) \geq \gamma\}$ is compact, where there are no $\nabla J(x, y) = 0$ points except $[x^*, y^*]^T$.*

Besides different control algorithm strategies for unicycle source seeking, two types of approaches to sensor designs have been published in the literature. In [7], a fixed sensor design is used where the vehicle's heading is always aligned with the sensor as illustrated in Figure 3-1. Another sensor design is developed by [3], where the sensor's movement is not coupled with the vehicle's motion, as shown in Figure 3-3. Resulting in an oscillating sensor to probe the signal field. Both sensor design approaches are considered in this thesis research to study their performance w.r.t. each other and how to apply both approaches on the ROSbot platform.

### 3-1-2 Fixed Sensor

The source seeking controller developed by [7] employs the control laws for the input velocities shown in Eq. (3-1.4). Where extremum seeking is directly applied on the angular velocity input, which is expressed in Eq. (3-1.4a) and derivative-like feedback on the forward speed in Eq. (3-1.4b). The signal strength sensor measurements are fed into a high-pass filter to remove the "DC-component" from the output of the filter. The filtered output is expressed by $\xi$ in Eq. (3-1.4).

$$\Omega = \alpha\omega\cos(\omega t) + c\xi\sin(\omega t) \tag{3-1.4a}$$

$$v = V_c + b\xi \tag{3-1.4b}$$

Eq. (3-1.4a) is constructed like a basic sinusoidal perturbation source seeking law. Where $\alpha\omega\cos(\omega t)$ is describing the perturbation added to excite the system and $c\xi\sin(\omega t)$ to estimate the angular gradient of the signal field. The logic behind the control law for the forward velocity input in Eq. (3-1.4b) is to increase the forward speed when the vehicle is driving closer to the source and orientated w.r.t. straight towards the source. This will result in the signal strength to increase and thereby $\xi > 0$. Alternatively, if the vehicle drives away from the source, which causes the signal strength to decrease and $\xi < 0$. This would let the vehicle slow down its forward velocity. The source seeking controller scheme is depicted in Figure 3-2.

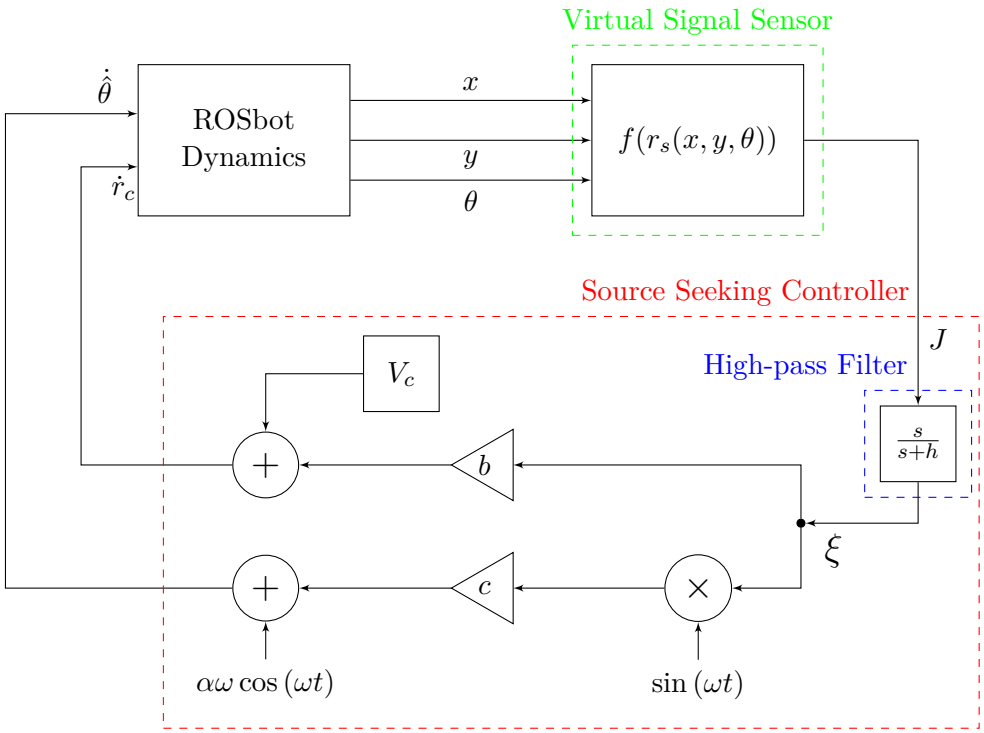



**Figure 3-2:** Continuous source seeking scheme

The source emits a nonlinear signal field, which is assumed to be quadratic, as given in Eq. (3-1.5). The variables $q_x$ and $q_y$ are unknown positive constants, and $r^*$ is the unknown maximizer with $f^* = f(r^*)$ being the unknown maximum as previously described.

$$J = f(r(x,y)) = f^* - q_x(x-x^*)^2 - q_y(y-y^*)^2 \tag{3-1.5}$$

Hence, when maximizing the objective function in Eq. (3-1.5) only the relative distance of the vehicle to the source is considered and not its orientation $\theta$ w.r.t. the source. The performance of the source seeking scheme in Figure 3-2 can be influenced by tuning the $\alpha$, $c$, $b$, $R$, $\omega$ and $V_c$ parameters.

### 3-1-3 Oscillating Sensor

An alternative approach for probing the signal field w.r.t. the configuration shown in Figure 3-1, is to decouple the sensor movement from the vehicle's movement. In [3], the authors present this as a solution for applications when periodic perturbations of the vehicle's movement to measure the signal field is not feasible or desired. Following the work described by [3], the sensor dynamics are modeled as expressed in Eq. (3-1.6) and illustrated in Figure 3-3.

$$r_s = r_c + Re^{j(\theta+\theta_s)} \tag{3-1.6}$$

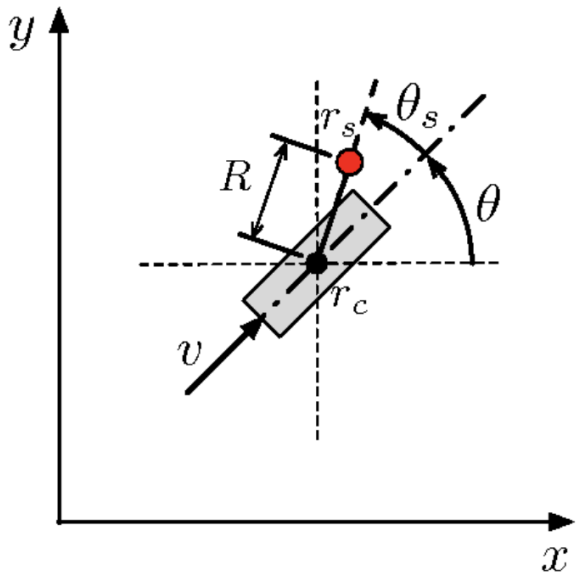



**Figure 3-3:** The model of the ROSbot's center dynamics with an oscillating sensor, from [3].

So the virtual sensor is capable of oscillating along the vehicle's geometric center $r_c$ and the sensor position w.r.t. the global $x$-$y$ coordinate frame becomes a function of the orientation of the vehicle $\theta$ and the angle between the centerline of the vehicle and sensor $\theta_s$. The angle $\theta_s$ is determined by the sinusoidal perturbation in Eq. (3-1.7).

$$\theta_s = \alpha \sin \omega t \tag{3-1.7}$$

Using an oscillating sensor for probing the signal field, the control law for the vehicle's orientation simplifies to Eq. (3-1.8).

$$\dot{\theta} = c\xi \sin \omega t \tag{3-1.8}$$

Hence, by utilizing the oscillating sensor approach, the perturbation term $\alpha\omega \cos \omega t$ is not affecting the vehicle's dynamics anymore.

## 3-2 Implementation

One of the objectives of this thesis research is to implement the unicycle source seeking method on the ROSbot platform, as stated in Chapter 1. First, the continuous source seeking scheme in Figure 3-2 needs to be transformed into a discrete scheme as depicted in Figure 3-4 for the fixed signal strength measuring sensor. A Zero-Order Hold (ZOH) discretization method is applied to convert the digital velocity input signals to an analog signal. For the discrete controller, a high sampling rate $T$ is chosen to minimize the effect of time-delays on the system. Therefore, in all simulations, a sampling rate of 40 Hz is used to publish the computed velocity inputs to the ROSbot.

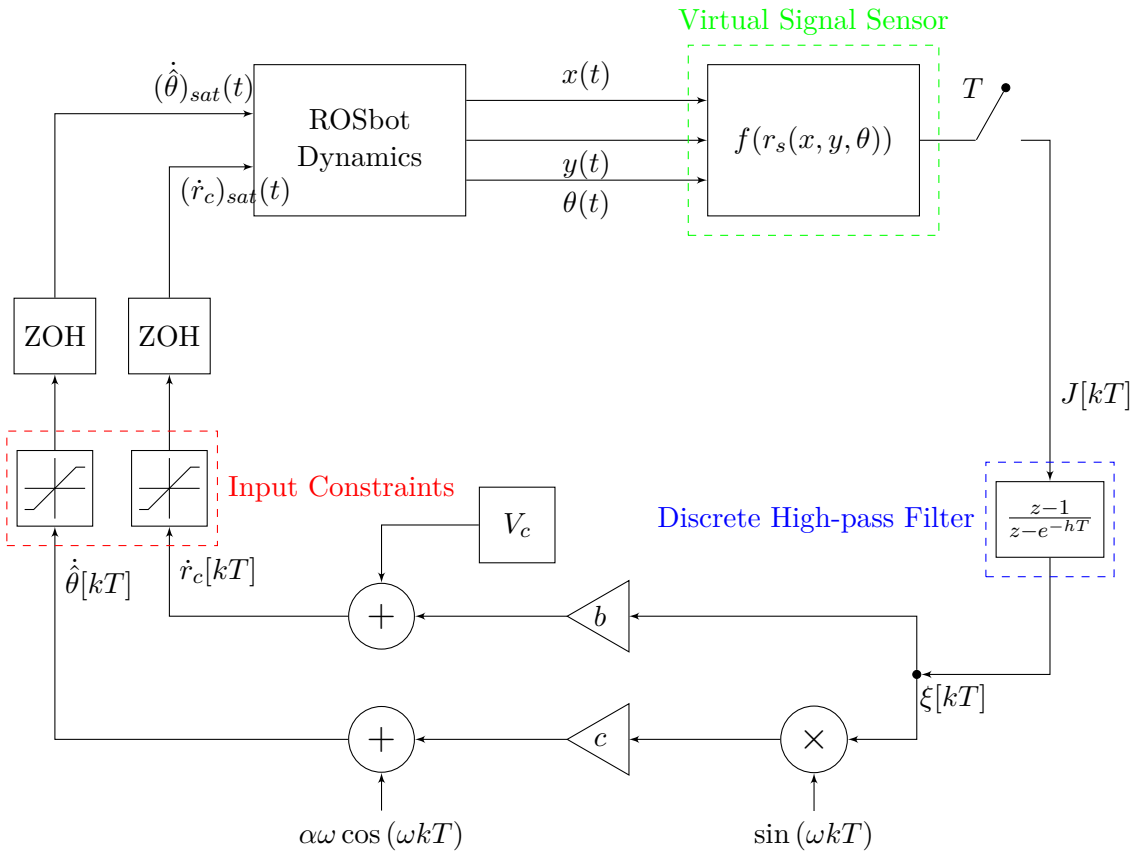



**Figure 3-4:** Discrete source seeking scheme

Instead of the continuous-time high-pass filter from the scheme in Figure 3-2, a discrete stable high-pass filter transfer function is applied $\frac{z-1}{z-e^{-hT}}$. The high-pass filter however still needs to be transformed into a difference equation so that it is suitable for the ROS programming environment, as expressed in Eq. (3-2.1).

$$H(z) = \frac{z-1}{z-e^{-hT}} \rightarrow y(n) = -e^{-hT}y(n-1) + x(n) - x(n-1) \tag{3-2.1}$$

When designing the source seeking algorithm, it is important to note that the cut-off frequency $h$ of the high-pass filter in Eq. (3-2.1) should be tuned to sufficiently capture the transient response of the changing signal strength. Besides, discretization of the source seeking scheme also constraints need be placed on the unicycle's velocity inputs. Since the source seeking algorithm should be suitable to be employed on the ROSbot. The velocity inputs need to be limited by the ROSbot's maximum translational velocity of 1.0 m/s and rotational velocity of 7.33 rad/s, as noted in [10].

To measure the scalar value of the signal strength, a virtual sensor is created. An artificial sensor is programmed into ROS as a separate node that generates signal strength values, shown in green in Figure 3-4. The signal values are generated by keeping track of where the geometric center of the ROSbot is located and its angle $\theta$ w.r.t. the source. Then, based on the orientation and geometric center of the ROSbot, the sensor location is computed

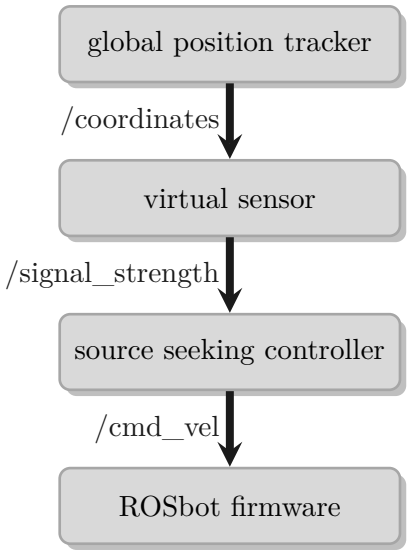



**Figure 3-5:** Communication scheme between the ROS nodes used for source seeking.

as expressed in Eq. (3-1.3) for the fixed sensor and Eq. (3-1.6) for the oscillating sensor approach. Subsequently, the virtual sensor generates using its location a signal strength value $J$ according to a predefined quadratic cost function, see Eq. (3-1.5).

In Figure 3-5, a scheme is shown how the algorithm is designed to communicate with the different ROS nodes needed for the unicycle source seeking method. The location and orientation of the ROSbot w.r.t. the global coordinate frame is tracked by the global position tracker node, which publishes the ROSbot's coordinates and heading using the */coordinates*-topic. Next, the virtual sensor node subscribes to the */coordinates*-topic and computes a signal strength value using the defined objective function Eq. (3-1.5) and sensor location Eq. (3-1.3). The signal strength value is published by the virtual sensor node to the */signal_strength*-topic. Then, the source seeking controller node containing the implemented source seeking algorithm is subscribed to the */signal_strength*-topic to perform source seeking and publishes appropriate velocity control inputs to the */cmd_vel*-topic of the ROSbot firmware. The global position tracker and virtual sensor ROS nodes in Figure 3-5, are publishing their respective topics at a loop-rate of 100 Hz, which is significantly higher than the source seeking controller's sampling rate $T$. To prevent low measurement frequency of the signal strength inhibiting source seeking by the ROSbot.

## 3-3 Simulation & Results

The unicycle source seeking algorithm is tested by implementing it in a Gazebo simulator environment. Where the signal emitting source is placed at the center of the environment. Hence, the parameters determining the objective function's unknown maximum in Eq. (3-1.5) are set to be $x^* = 0$ and $y^* = 0$. The shape defining parameters of Eq. (3-1.5) are $f^* = 1$, $q_x = 1$ and $q_y = 1$, all of the these parameters are not known *a priori* to the source seeking vehicle.

In Figure 3-6, a simulation set-up is shown in Gazebo where the initial position of the ROSbot

is located at $x : 3[m]$, $y : 3[m]$ and the initial orientation pointed towards the source. Both the fixed and oscillating virtual sensor approaches are evaluated to study how the source seeking parameters affect the performance.

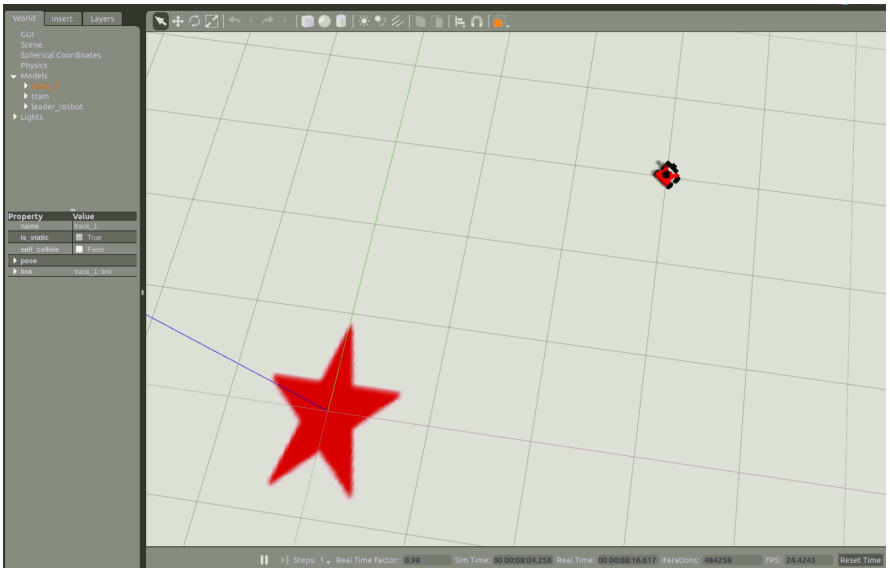


**Figure 3-6:** Figure showing the Gazebo simulation environment, with the ROSbot located in the top-right corner at $x : 3[m]$, $y : 3[m]$ and the source at the red star at $x : 0[m]$, $y : 0[m]$

To evaluate the performance of the source seeking controller, the following characteristics are determined to be of importance:

- Time to reach the source
- Path length to reach the source
- Consistency of the paths towards the source
- Convergence around the source

The parameters listed above are computed as follows, the time to reach the source is computed by measuring how long, on average, the ROSbot's geometric center takes to reach the source within 0.1m. The path length to reach the source is calculated by determining the average traveled distance the ROSbot's geometric to reach within 0.1m. The consistency of the generated paths is measured by taking the maximum standard deviation of the trajectories the ROSbot traveled to reach the source within 0.1m. Lastly, convergence around the source is defined by the maximum distance the ROSbot overshoots when it reaches the source within 0.1m. All parameter configurations are run 10 times to determine how they influence the performance based on the properties listed above.

### 3-3-1 Fixed Sensor Source Seeking

First, the fixed sensor approach described in section 3-1-1 is studied by choosing different parameter values. The influence of the perturbation frequency $\omega$ is tested on the source

seeking ROSbot, by varying $\omega$ while keeping the other parameters constant, as shown in Figure A-1, Figure A-2 and Figure A-3

Besides, the perturbation frequency $\omega$ also the parameter $\alpha$ can be utilized to change the perturbation term to excite the system. In Figure A-7 the $\alpha$ constant is doubled and in Figure A-8 reduced by 50% w.r.t. Figure A-2 to evaluate the influence of $\alpha$ on source seeking.

Also, the influence of the gain $c$ on the gradient estimate and $h$ parameter for the high-pass filter is studied. To determine the affects of increasing the gain on the gradient estimate, $c$ is in Figure A-4 increased 2 times w.r.t. Figure A-2. Next, the cut-off frequency $h$ is tuned to improve the trajectory of the ROSbot in Figure A-6, Figure A-2 and Figure A-5.

### 3-3-2 Oscillating Sensor Source Seeking

After conducting simulation experiments using the fixed sensor approach, the oscillating sensor model explained in section 3-1-3 is simulated to evaluate the influence of decoupling the sensor movement from the vehicle's movement. A much higher perturbation frequency $\omega$ for probing the signal field can be applied using an oscillating sensor approach since the sensor movement is decoupled from the vehicle.

So first, the affect of having a higher perturbation frequency $\omega$ as a result of decoupling the sensor movement is investigated in Figure A-12, Figure A-13 and Figure A-14. Thereafter, in Figure A-15 w.r.t. Figure A-13 the cut-off frequency $h$ is increased to study how would alter the performance.

## 3-4 Discussion

Based on simulations performed in section 3-3, the results are discussed in this section on how the trajectory of the ROSbot can be influenced by tuning the source seeking parameters $\alpha$, $\omega$, $b$, $c$, $V_c$ and $h$, for both the fixed and oscillating sensor approach. First, the factors that need to be considered when the forward velocity is tuned are explained based on both sensor types' simulations. Thereafter, an elaboration is given on how the parameters influence the source seeking controller's angular gradient estimation. Lastly, conclusions are drawn on the performance of the source seeking controller for both sensor type approaches.

#### Forward Velocity Tuning

As explained in section 3-1-1, the $V_c$ and $b$ parameters affect the vehicle's forward velocity. The $b$ parameter is employed as a dampening factor on the forward speed. It needs to be tuned so that the high-pass filtered signal $\xi$ is reduced sufficiently to slow the vehicle's forward velocity down when it approaches the source while not amplifying the forward velocity too much when it is navigating towards the source. For the fixed sensor approach this is illustrated when comparing Figure A-2 to Figure A-9, where the $b$ parameter is increased too much resulting in a deterioration of the generated paths towards the source. However, $b$ can not be set too small w.r.t. $V_c$. Because if parameter $b$ is set too small, the constant added forward speed term $V_c$ will be too dominant, which prevents the vehicle converging to proximity around

the source since the dampening forward velocity is insufficient as is seen when comparing Figure A-2 to Figure A-10. For the oscillating sensor method, the derivate term $b$ has to be chosen smaller than the fixed sensor. The filtered signal $\xi$ contains more of the steady-state due to the perturbation frequency for source seeking being much higher than the fixed sensor. Therefore, a higher constant forward velocity $V_c$ compared to the fixed sensor approach needs to be selected for the oscillating sensor.

### Angular Gradient Estimation

The $\alpha$, $\omega$, and $c$ parameters influence the actual perturbation-based source seeking, which is used to control the angular velocity to steer the ROSbot towards the source. The vehicle's sensor dynamics determine how the vehicle is capable of measuring the signal field, which is significantly different for the fixed and oscillating sensor approach, as described in section 3-1-3. Where $\omega$ dictates the perturbation frequency of the source seeking controller. In case a fixed sensor approach is employed on the ROSbot, the frequency $\omega$ is much lower than the oscillating sensor approach as depicted in the simulation results in Appendix A-1 and A-2 because the ROSbot's vehicle dynamics restrict the frequency at which the perturbation term $\alpha\omega\cos(\omega kT)$ can excite the angular velocity to probe the signal field. Whereas, for the oscillating sensor approach, the perturbation term $\alpha\sin(\omega kT)$ is not inhibited by the ROSbot's dynamics. Therefore, allowing it to operate at a much higher perturbation frequency $\omega$. The resulting trajectories of the ROSbot's geometric center towards the source also reflect this difference. For the fixed sensor approach in Appendix A-1, the trajectories are sinusoidally looping towards the source. The oscillating sensor method in Appendix A-2 the trajectories follow the gradient of the signal field.

For both source seeking approaches, increasing the perturbation frequency $\omega$ will decrease the time the source seeking ROSbot takes to reach the source. Comparing Figure A-2 to Figure A-3 where $\omega$ is doubled from 0.5 $rad{*}s^{-1}$ to 1 $rad{*}s^{-1}$ using a fixed sensor approach the average time the ROSbot takes to reach the source decreases, however the path reliability decreases as well which can be seen from the increased standard deviation of the generated paths. Similarly for the oscillating sensor approach, when $\omega$ is increased from 10 $rad * s^{-1}$ in Figure A-14 to 20 $rad * s^{-1}$ in Figure A-13 the average time to reach the source is reduced. Increasing the perturbation frequency $\omega$ for the oscillating sensor approach though also results in a lesser convergence around the source as can be seen when comparing Figure A-14 to Figure A-12.

The $\alpha$ parameter can be used to control the amplitude of the perturbation term $\alpha\omega\cos(\omega kT)$ for the fixed sensor or $\alpha\sin(\omega kT)$ for the oscillating sensor approach. The amplitude of the perturbation term must be set so that the system's excitation is adequately large enough. In the case of the fixed sensor approach, it would entail that the ROSbot will keep turning and thereby measuring signal strengths in its proximity without inhibiting the estimation of the angular gradient. Alternatively, when using the oscillating sensor method, assuring that the sensor covers enough of the signal field w.r.t. the ROSbot's heading to sufficiently estimate the gradient of the signal field.

If the parameter configurations for the fixed sensor approach are tuned, more emphasis can be set on the system's excitation by choosing a high $\alpha$, as shown in Figure A-7. Thereby, the trajectory of the ROSbot will be less determined by the angular gradient estimate and more by the perturbation term $\alpha\omega\cos(\omega kT)$ to dictate the angular velocity input. So if the

ROSbot steers towards the source, the forward velocity will increase, and when the ROSbot turns away, the forward velocity is reduced.

The signal field's angular gradient is approximated by multiplying $\sin(\omega kT)$ with the high-pass filtered signal $\xi$. To amplify the estimated angular gradient $\xi \sin(\omega kT)$, the gain $c$ is placed on this signal. The gain $c$ should be significant enough to increase the estimated angular gradient so that the vehicle is orientated along the gradient of the signal field. Parameters $\alpha$ and $c$ are tuned so that both the system's excitation and angular gradient estimation are appropriately weighed. In Figure A-8, simulations are shown for the fixed sensor approach to demonstrate how the path consistency is degraded when too much weight is placed on the estimated angular gradient by a low $\alpha$, but improving the time, the ROSbot takes to reach the source. In case the gain $c$ is increased for the oscillating sensor design, the convergence around the source improves but the time towards the source decays, as can be seen in Figure 3-8 where $c$ is doubled w.r.t. Figure A-13.

The $h$ parameter of the high-pass filter is chosen to correctly capture the transient response of the measured scalar signal strength used to estimate the angular gradient. Whenever a too high $h$ is chosen, too much of the transient is filtered. Vice versa, when a low $h$ is selected, the steady-state component of the filtered signal $\xi$ restricts the performance of the source seeking controller. For the fixed sensor approach in Figure A-6, Figure A-2 and Figure A-5 the $h$ parameters is changed to find the appropriate value while keeping the other parameters constant. A trade-off needs to be made where increasing $h$ results in a more consistent but slower path towards the source. Similarly, in Figure A-15 the cut-off frequency is increased w.r.t. Figure A-13 to demonstrate performance difference for the oscillating sensor method. The increase in $h$ also results in the ROSbot reaching the source faster but having a worse convergence around the source.

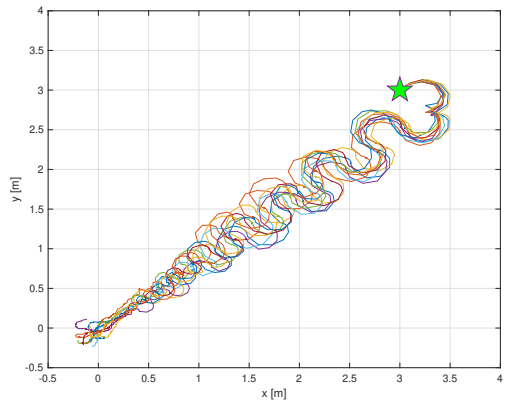
Besides, the source seeking parameters $\alpha$, $\omega$, $b$, $c$, $V_c$ and $h$ also the distance $R$ at which the sensor is placed influences the performance of the controller substantially. As can be seen when comparing Figure 3-7 to Figure A-11, where the fixed sensor is placed closer to the geometric center of the ROSbot. Consequently, path consistency decays because more vehicle movement is needed to probe the signal field. Thus, increasing the distance of $R$ of the sensor improves the source-seeking algorithm's performance, but the distance $R$ is set to $0.1m$. Thereby not having to increase the dimensions of the ROSbot as defined in section 2-3.

## Conclusion

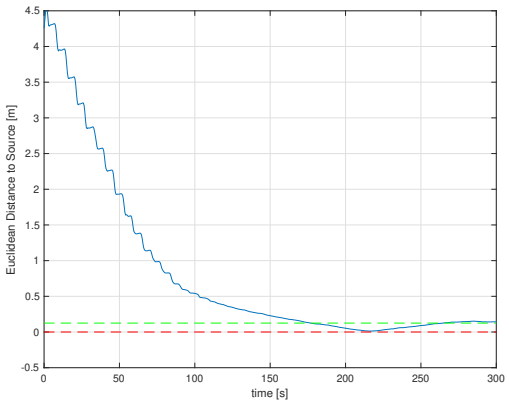
As can be seen from the simulations in Appendix A-1 and A-2, the fixed sensor and oscillating sensor approach produce significantly different trajectories when used for source seeking. Using a fixed sensor approach has benefit w.r.t. to the oscillating sensor that the fixed sensor is relatively simple to mount on a robotic vehicle and does not require extra actuators to operate.

On the other hand, the oscillating sensor method is capable of following the gradient of the signal field much more accurately compared to the fixed sensor approach, which could be advantageous when producing a consistent and straight path towards the source is crucial. These properties of both source seeking approaches should guide the decision of the engineer whether to choose a fixed or oscillating sensor, also considering the mission requirements and

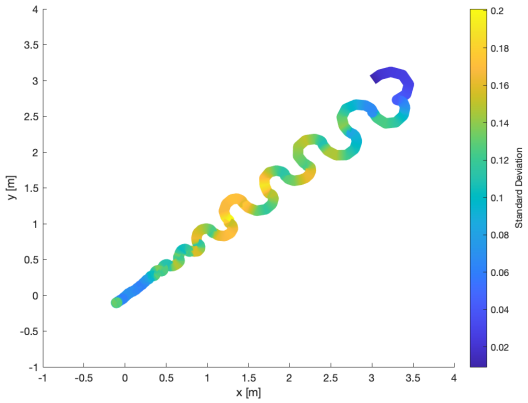
available resources. The simulations using the best performing parameter configurations for both sensor design approach are shown in Figure 3-7 and Figure 3-8.




**(a)** All trajectories




**(b)** Distance to source




**(c)** Mean trajectory with standard deviation

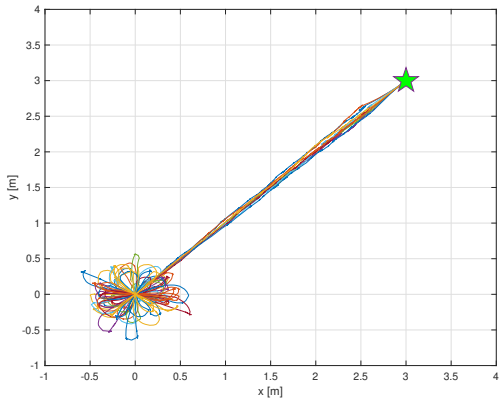
| Parameter | Value | Unit (SI) |
|---|---|---|
| $\omega$ | 0.5 | $rad$ |
| $h$ | 1 | unitless |
| $c$ | 80 | unitless |
| $\alpha$ | 1.8 | unitless |
| $R$ | 0.1 | $m$ |
| $b$ | 0.5 | unitless |
| $V_c$ | 0.005 | $m/s$ |

**(d)** Tuning parameters

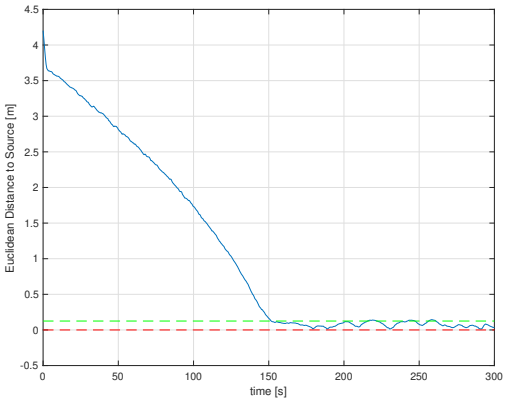
| Characteristic | Value | Unit (SI) |
|---|---|---|
| Time to source | 176.8 | $s$ |
| Path length | 4.13 | $m$ |
| Path consistency | 0.201 | unitless |
| Convergence | 0.152 | $m$ |

**(e)** Performance characteristics

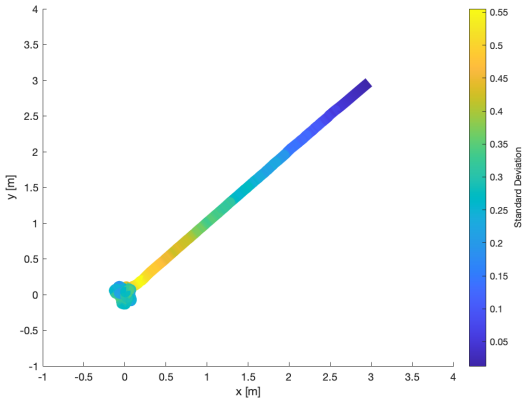
**Figure 3-7:** Simulation of 10 trajectories of the source seeking controller using a fixed sensor approach, where the ROSbot's geometric center is analyzed. The source located at $x : 0[m]$, $y : 0[m]$.




**(a)** All trajectories




**(b)** Distance to source




**(c)** Mean trajectory with standard deviation

| Parameter | Value | Unit (SI) |
|---|---|---|
| $\omega$ | 20 | $rad/s$ |
| $h$ | 1 | unitless |
| $c$ | 190 | unitless |
| $\alpha$ | 1.8 | unitless |
| $R$ | 0.1 | $m$ |
| $b$ | -0.25 | unitless |
| $V_c$ | 0.05 | $m/s$ |

**(d)** Tuning parameters

| Characteristic | Value | Unit (SI) |
|---|---|---|
| Time to source | 151.8 | $s$ |
| Path length | 4.09 | $m$ |
| Path consistency | 0.555 | unitless |
| Convergence | 0.144 | $m$ |

**(e)** Performance characteristics

**Figure 3-8:** Simulation of 10 trajectories of the source seeking controller using an oscillating sensor approach, where the ROSbot's geometric center is analyzed. The source located at $x : 0[m]$, $y : 0[m]$.

# Chapter 4

# Obstacle Avoidance

In this chapter, the method for creating an obstacle avoidance algorithm suitable to incorporate with the source seeking controller from Chapter 3 will be elaborated on. Two different types of algorithms for obstacle avoidance are studied in this Chapter, namely, an artificial potential function (APF) and a hybrid adaptive feedback (HAF) approach. Obstacle avoidance in this thesis research is considered to be, as defined by [18], to let an autonomous vehicle steer to a trajectory that overcomes expected or unexpected obstacles.

In section 4-1, the APF algorithm is described, which steers the vehicle away from the obstacle by creating a repulsive potential function around the obstacle. However, the authors in [24] describe a pathological situation where due to topological obstructions induced by the obstacle, the APF approach can crash or get stuck behind the obstacle as further explained in section 4-1-1. The HAF law developed by [19], is designed to overcome this problem. In section 4-2, the HAF approach of [19] is elaborated and further improved upon. Since the HAF method employed in [19] uses two assumptions that make it less useful to deploy on an autonomous vehicle for a scientific mission as described in Chapter 1. The first assumption is that the obstacle's location is assumed to be known *a priori*, in section 4-2-1 a solution is presented that overcomes requiring this assumption. The second assumption is that the obstacle's orientation w.r.t. the source is assumed to be known *a priori*. If the obstacle's orientation w.r.t the source is assumed to be known *a priori*, it defeats the purpose of employing a source seeking controller. Because this assumption would imply that the gradient of the signal field would also have to be assumed to be known *a priori* when the vehicle is avoiding an obstacle. Therefore, in section 4-2 an algorithm is developed to improve upon the HAF law from [19] that does not require this assumption. Thereafter, in section 4-3 both the APF and HAF algorithms are compared in simulation. Using simulations, the conditions under which the APF method would fail and the HAF algorithm would deliver on its promises are studied. Finally, in section 4-4 conclusions are drawn on the improved HAF algorithm compared to the APF approach.

The signal field used to evaluate both obstacle avoidance methods is assumed to follow Assumption 3-1.1, which is similar as used by [19]. The signal field emitted by the source is

also not assumed to be affected by the obstacle. Both types of sensor designs described in Chapter 3 will be used to study the obstacle avoidance methods.

## 4-1 Artificial Potential Function

An APF based obstacle avoidance approach works by creating a potential function that includes the scalar signal of the source and all the interaction rules of the ROSbot with the environment, e.g., an obstacle [29]. The authors in [32] for example, develop an algorithm that, by minimizing a potential function, achieves source seeking, formation control, obstacle avoidance, and collision avoidance. How such a potential function is constructed is shown in Eq. (4-1.1a), where $\rho$ is the range for the repulsion potential and $d_q(x, y)$ is the Euclidean distance between the vehicle and obstacle as expressed in Eq. (4-1.1b). The sensors equipped on the ROSbot, see Table 2-1, are utilized to measure the relative distance to the obstacle, e.g., the laser scanner for computing the repulsive potential.

$$U(z) = \begin{cases} -z^2 \log\left(\frac{1}{z}\right), & \text{if } z \in [0, \rho] \quad \text{where } z = d(x, y) \\ 0, & \text{if } z > \rho, \end{cases} \tag{4-1.1a}$$

$$d(x, y) = \sqrt{(x - x_{ob})^2 + (y - y_{ob})^2} \tag{4-1.1b}$$

The obstacle avoidance potential function in Eq. (4-1.1a) can simply be added to the measured scalar signal strength by the sensor, $f(r_s(x, y))$ from Eq. (3-1.5), to combine source seeking and obstacle avoidance, see Eq. (4-1.2).

$$J(x, y) = f(r_s(x, y)) + U(d(x, y)) \tag{4-1.2}$$

### 4-1-1 Local Minimum Problem

One significant downside of implementing conventional APF methods for obstacle avoidance from Eq. (4-1.1) and Eq. (4-1.2) is the occurrence of local minima. Local minima develop when the sum of all forces is zero at a certain location, which results in the vehicle to stop driving and becoming stuck at that point as shown in Figure 4-1.

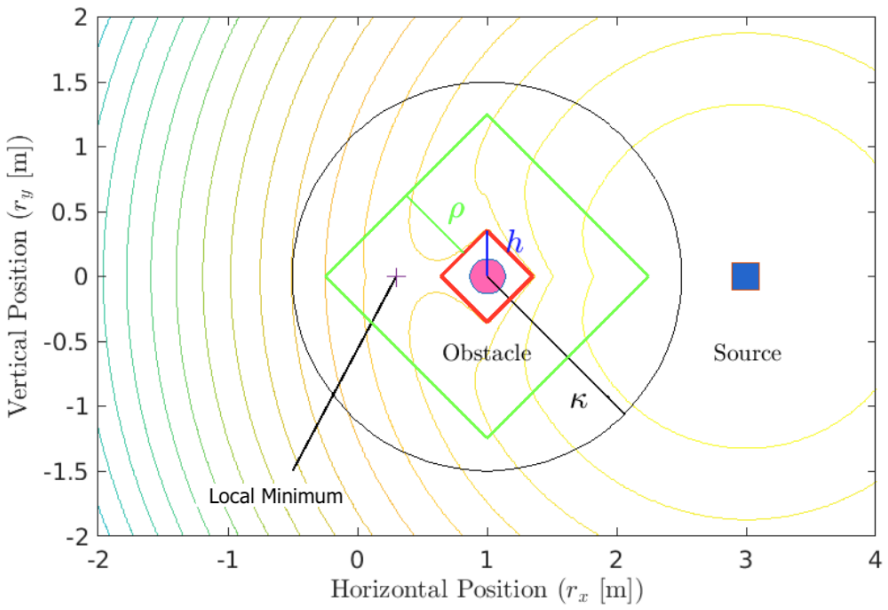
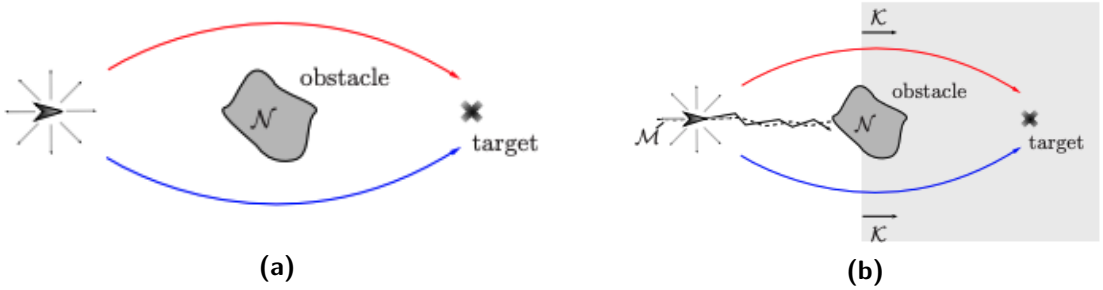



**Figure 4-1:** Figure showing the occurrence of a local minimum when using the conventional APF approach. Where a virtual box (red) is created around the obstacle with tunable height $h$, $\kappa$ is the detection range of the obstacle and $\rho$ being the region that is influenced by the repulsion potential, from [16].

Alternatively, like in Figure 4-2 as identified by the authors in [24], when the forward velocity of the vehicle is kept constant, and only the angular velocity is controlled for by the algorithm, the autonomous vehicle should avoid the obstacle by either choosing the red or blue trajectory in Figure 4-2a. However, in the case of measurement noise, the vehicle could keep switching between both trajectories and crash into the obstacle, as shown in Figure 4-2b.




**(a)** **(b)**

**Figure 4-2:** Global steering to a target with obstacle avoidance, from [24]. (a) Global steering of an autonomous vehicle to a target with obstacle avoidance. The task is to control the vehicle so that it avoids the obstacle N and approaches the target $x$. (b) From initial conditions above the (dashed) line $M$, the trajectories approach the set $K$, and from there they approach the target from above the obstacle, while from initial conditions below the (dashed) line, the trajectories approach the set $K$ and then the target from below the obstacle. In the presence of measurement noise, a trajectory could stay in a neighborhood of the (dashed) line, potentially causing the vehicle to crash into the obstacle

## 4-2 Hybrid Control

To prevent the situation depicted in Figure 4-2b from happening, an algorithm should be developed that assures that the vehicle reaches set $K$. Therefore, [19] produced a solution to tackle the local minimum problem whilst guaranteeing that the feedback law's stability properties are preserved for arbitrary small adversarial noise signals. The authors in [19] apply research published in [25], by dividing the state space of the velocity adaptive control law for the vehicle and adding a switch state $q \in \{1, 2\}$. Which transforms Eq. (4-1.2) into a mode-dependent localization function $J_q(x, y)$ shown in Eq. (4-2.1), where the output of Eq. (4-2.1) is fed to a hybrid and model-free feedback law.

$$J_q(x, y) = -J(x, y) + B(d_q(x, y), J(x, y)) \tag{4-2.1}$$

Where differently from Eq. (4-1.1b), $d_q(x, y) = |[x, y]^T|^2_{\mathbb{R}^2 \setminus \mathbb{O}_q}$ is a function that maps the position of the vehicle $[x, y]^T \in \mathbb{R}^2$ to the squared value its distance to the set $\mathbb{R}^2 \setminus \mathbb{O}_q$, see Figure 4-3. Hence, in case of two switch states $q \in \{1, 2\}$, for each switch state a specific localization function is computed, e.g. $J_1(x, y)$ and $J_2(x, y)$. The resulting $d_q(x, y)$ is an argument for the barrier function $B(\cdot)$, which is defined in Eq. 4-2.2 for a logarithmic barrier function. The computed barrier values are multiplied by $(|J| + 1)$ to ensure that the barrier value is dominant w.r.t. the measured scalar signal strength. To preserve continuity of the mode depended signal, the barrier function should reduce to zero when the squared value distance equals $\rho$ as stated by [16].

$$B(z, J) = \begin{cases} (|J| + 1)(z - \rho)^2 \log\left(\frac{\rho}{z}\right), & \text{if } z \in [0, \rho] \\ 0, & \text{if } z > \rho, \end{cases} \tag{4-2.2}$$

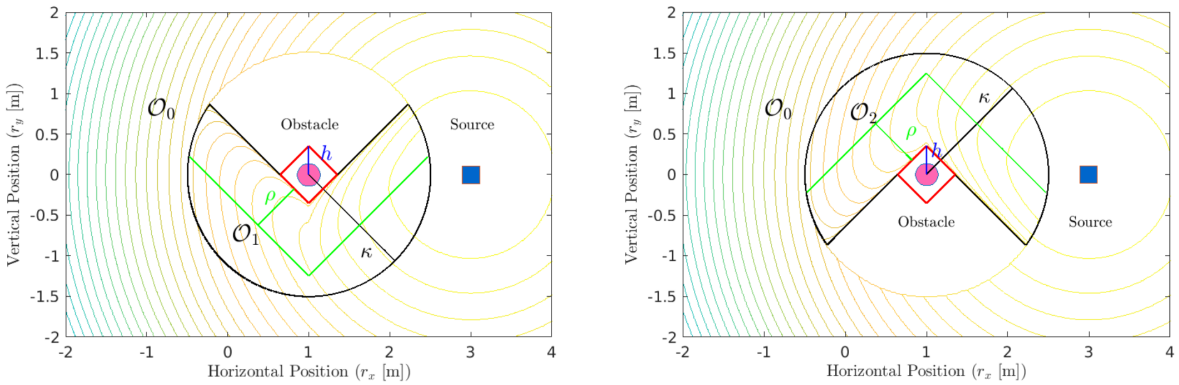
Figure 4-3 is from research published by [16], which illustrates how the dividing lines for the state space are drawn for the hybrid feedback law. In Figure 4-3, $\kappa$ is the perimeter around the obstacle where the obstacle avoidance method is used. So if the autonomous vehicle is within $\kappa$ distance w.r.t. the obstacle two mode-dependent localization values are computed what mode is chosen for $J_q(r_s(x, y))$ as is further detailed in Algorithm 1.

**Algorithm 1** Mode update law for $q^+$

**Input:** $z$, $z_1$, $z_2$, $q$
**Output:** $q^+$

```
function UpdateStateMode(z, z_1, z_2, q)
    if √z ≤ κ then
        if r_i ∉ O_1 then
            q^+ = 2
        else if r_i ∉ O_2 then
            q^+ = 1
        else if q = 1 then
            if J_1 ≥ (μ − λ)J_2 then
                q^+ = 2
            else
                q^+ = 1
            end if
        else if q = 2 then
            if J_2 ≥ (μ − λ)J_1 then
                q^+ = 1
            else
                q^+ = 2
            end if
        else
            if z_1 > z_2 then
                q^+ = 1
            else
                q^+ = 2
            end if
        end if
    else
        q^+ = 0
    end if
end function
```

Algorithm 1 from [19], is implemented for switching between two modes ($q \in \{1, 2\}$) to avoid the vehicle from following the $M$-trajectory in Figure 4-2b. Where $\mu > 1$ is used to prevent recurrent jumps between the two modes, while the $\lambda \in (0, \mu - 1)$ parameter is applied to guarantee that $O_1$ and $O_2$ overlap for establishing robustness.




**(a)** Partitioned space for modes 0 and 1

**(b)** Partitioned space for modes 0 and 2

**Figure 4-3:** Partitioned state space into two modes $q \in \{1, 2\}$. Where a virtual box (red) is created around the obstacle with tunable height $h$, the edges of the box are extended to intersect the detection perimeter $\kappa$. The edges determine the mode-dependent spaces with $\rho$ being the limit for the barrier function, from [16].

### 4-2-1 Obstacle Location

To determine where obstacles are located in a sparsely cluttered environment, the Light Detection And Ranging (LiDAR) sensor on the ROSbot platform is used; see Table 2-1. The principle of a LiDAR sensor is based on laser triangulation ranging, which computes relative distances to objects by emitting a modulated infrared laser signal towards. Objects in the ROSbot's environment then reflect the laser signal. The distance of the LiDAR w.r.t. objects is obtained by capturing the reflected signal and measuring the time of flight, as shown in Figure 4-4.

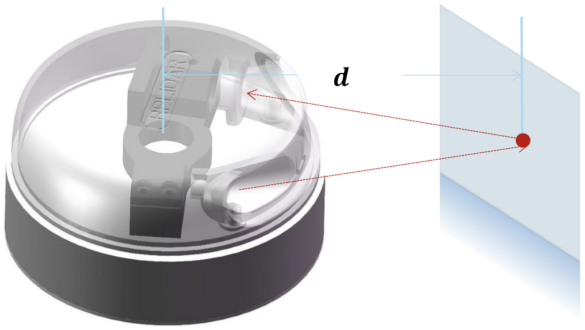



**Figure 4-4:** A schematic model of the LiDAR sensor on the ROSbot, where $d$ is the relative distance between object and LiDAR, from [26].

For detecting and tracking obstacles, the open-source *obstacle_detector*-package [20] based on the paper published by [21] is used. The package provides tools to process data obtained by $2D$ laser scanners for object detection. The obstacles that the ROSbot vehicle should detect and avoid are assumed to be of a cylinder shape. The obstacles are approximated from the laser scan data by a circular geometric model expressed in Eq. (4-2.3).

$$c \triangleq \{r, \vec{p}_0\} = \{r, (x_0, y_0)\} \tag{4-2.3}$$

Where $r$ is the radius of the circle and $\vec{p}_0 = (x_0, y_0)$ the coordinates of its center. To compute the radius and center of an obstacle the *obstacle_detector*, first measures the length of line segment $\vec{l}$, illustrated in Figure 4-5. Then, by using Eq. (4-2.4a) it is possible to calculate the radius of the circle used to approximate the obstacle. Subsequently, the center point $\vec{p}_0$ is computed with Eq. (4-2.4b) where $\vec{n}$ is the normal vector of the measured line segment $\vec{l}$ pointing to the scanner.

$$r = \frac{\sqrt{3}}{3}\vec{l} \tag{4-2.4a}$$

$$\vec{p}_0 = \frac{1}{2}(\vec{p}_1 + \vec{p}_2 - r\vec{n}) \tag{4-2.4b}$$

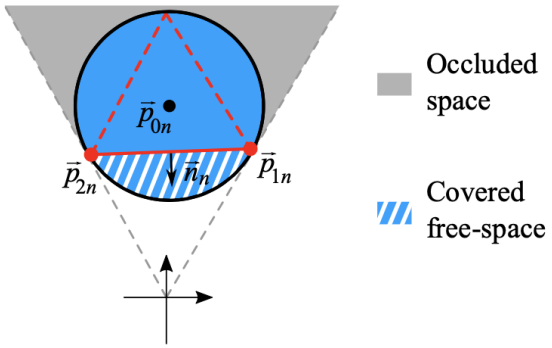


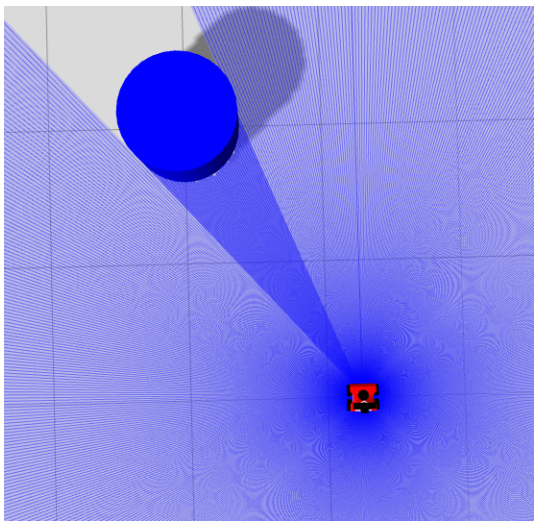

**Figure 4-5:** Extracting a circle from a detected segment $\vec{l}$ (bold red), from [21].

Figure 4-6a shows how the ROSbot would emit LiDAR rays in a simulation scenario in Gazebo with an obstacle in its proximity. Where, in Figure 4-6b the sensed LiDAR points are converted in RVIZ into objects using the *obstacle_detector*-package. To track an obstacle in case either the LiDAR and/or obstacle is moving the *obstacle_detector*-package employs an obstacle tracker algorithm which utilizes a Kalman filter [31] to track the obstacle's position.

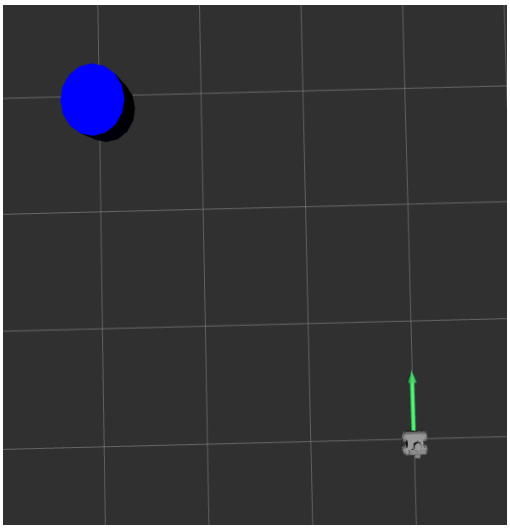
### 4-2-2 Drawing Partitioning Lines

Since the obstacle's location and its orientation w.r.t. the source is not assumed to be known *a priori*, an approach has to be developed to draw the partitioning lines of the state space as shown in Figure 4-3. Hence, a new algorithm is developed in this thesis research to construct the partitioning lines that repel the ROSbot away from the detrimental line $M$, further described in this section.

To start, the acquired LiDAR data on the obstacle is approximated using the circular geometric model described in section 4-2-1. The algorithm works as follows; first, a virtual box around the detected circular obstacle is drawn based on the circle's computed center point and radius. If the center point of the obstacle is within $\kappa$ distance of the vehicle and the vehicle is orientated straight towards the obstacle, the partitioning lines are drawn to intersect the $\kappa$ perimeter as shown in Figure 4-7.



**(a)** Simulation scenario in Gazebo, where the ROSbot emits LiDAR rays visible in blue



**(b)** Obtained LiDAR points of the obstacle are converted into an object (blue) by the *obstacle_detector*-package, shown in Rviz.

**Figure 4-6:** A simulation scenario showing how the *obstacle_detector*-package would convert LiDAR data into an detected object.

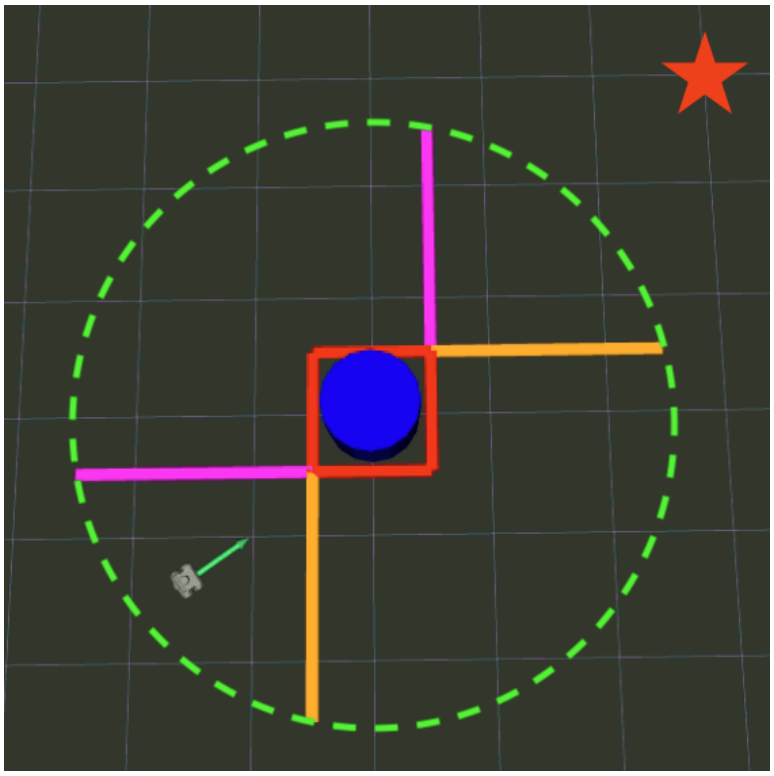
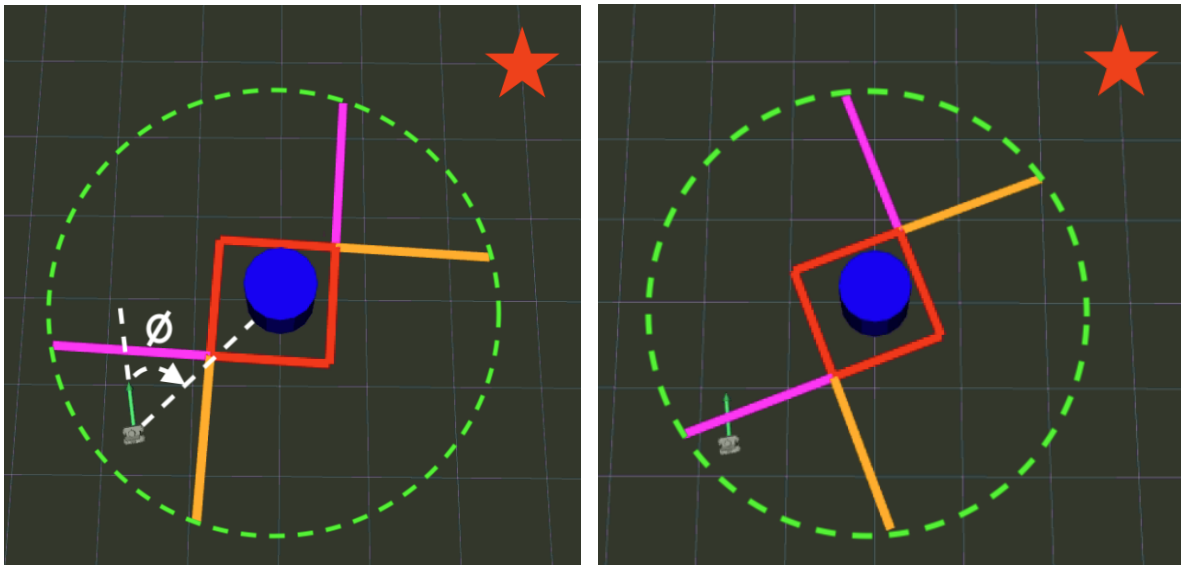


**Figure 4-7:** Figure showing how in Rviz the partitioning lines (in magenta and orange) should be drawn w.r.t. to the source which is located at the red star. The robot is located in the bottom left corner with the green arrow being its heading, and the green dashed circle the $\kappa$ perimeter. The obstacle is the blue cylinder, with the virtually created box being drawn in red.

The situation illustrated in Figure 4-7 depicts an "ideal" scenario where the ROSbot's heading is aligned with the obstacle and source. Then, the partitioning lines can be be drawn from the closest corner of the virtual box w.r.t. the ROSbot. The lines are drawn with a $\frac{1}{4}\pi$ angle from the line between the ROSbot and the closest virtual box corner until they intersect $\kappa$.

However, if the vehicle is not orientated straight towards the obstacle and the source, the partitioning lines' drawing becomes more difficult. As discussed above, the orientation of the obstacle w.r.t. the source in the global coordinate frame is not assumed to be known *a priori*. Therefore, first, the angle $\phi$ between the ROSbot and obstacle is computed. Thereby, it is assumed that the ROSbot is orientated to the source. The angle $\phi$ determines how much the partitioning lines should be rotated. So, in case the ROSbot is orientated at a $\phi$ angle away from the obstacle as illustrated in Figure 4-8a. The partitioning lines would need to be rotated by $\phi$, as shown in Figure 4-8.



**(a)** The partitioning lines are still drawn as depicted in Figure 4-7. However the ROSbot is now orientated at an $\phi$ angle away from the obstacle.

**(b)** The partitioning lines rotated by $\phi$ corresponding to how far the ROSbot is rotated away from the source.

**Figure 4-8:** Figure showing how in Rviz the partitioning lines (in magenta and orange) are drawn w.r.t. to the source (red star) when the ROSbot is orientated away from the source and obstacle (blue cylinder). The robot is located in the bottom left corner with the green arrow being its heading and $\phi$ describing the angle between the ROSbot's heading and the obstacle.

To determine the angle $\phi$, the $atan2(y, x)$-function is utilized which computes one unique arc tangent value from $x$ and $y$. The signs of both $x$ and $y$ are used to find the quadrant of the result, thereby choosing the right branch of the arc tangent of $\frac{y}{x}$. As a result, diametrically opposite directions can be distinguished, e.g., $atan2(1, 1)$ will equal $\frac{\pi}{4}$ and $atan2(-1, -1) = \frac{-3\pi}{4}$. In [19], it is assumed that the correct orientation of the partitioning lines w.r.t. the source is known to the vehicle. However, this assumption implies knowing the heading towards the source of the signal field *a priori* which defeats the need for a source seeking method that estimates the gradient.

As described in section 3-4, an inherent characteristic of unicycle perturbation based source seeking is that the trajectory traveled by the ROSbot towards the source is "sinusoidally

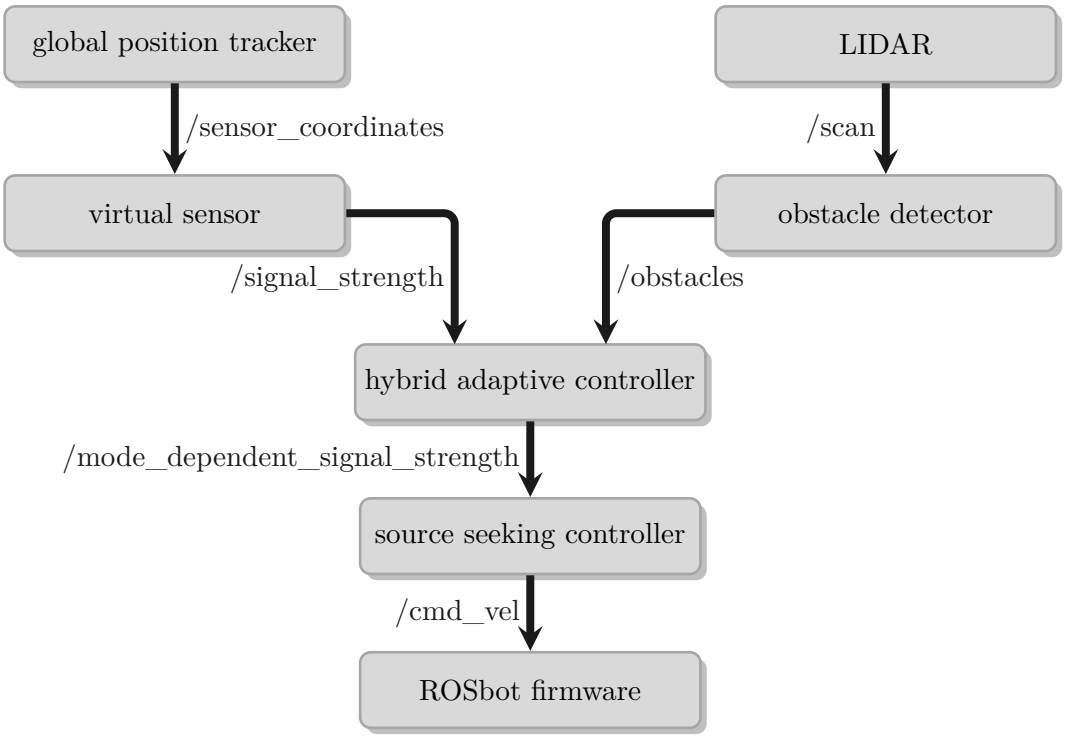



**Figure 4-9:** Communication scheme between ROS nodes to provide HAF obstacle avoidance based on LIDAR data.

looping." When the ROSbot is within $\kappa$ distance of the obstacle, and it makes a sharp turning movement while staying close to its location, the partitioning lines would therefore need to be rotated equally, as can be seen when comparing Figure 4-8a to Figure 4-8b. However, this behavior is not desirable because sudden turns are a property of the employed source seeking method and not due to significant gradient changes of the signal field. Therefore, a low-pass filter is applied on $\phi$ to prevent high-frequency changes in $\phi$ to influence the partitioning lines' rotation. As a low-pass filter, a moving average filter is used.

In Figure 4-9, the communication scheme between ROS nodes is shown needed to implement the hybrid adaptive obstacle avoidance method. The hybrid adaptive controller receives information on the obstacle's estimated radius and relative position by subscribing to the */obstacles*-topic. The obstacle information is acquired by the obstacle detector node by subscribing to the */scan*-topic that contains the LiDAR data and publishing the estimated radius and relative position using the *obstacle_detector*-package to the */obstacles*-topic. Besides data on the obstacle information, the hybrid adaptive controller also receives the scalar signal strength using the */signal_strength*-topic published by the virtual sensor as described in section 3-2. Next, the hybrid adaptive controller computes using the logic shown in Algorithm 1, a signal strength that corresponds to the mode it is in and publishes the scalar signal on the *mode_dependent_signal_strength*-topic. Finally, the source seeking controller subscribes to the *mode_dependent_signal_strength*-topic and performs source seeking as elaborated on in section 3-2.

## 4-3 Simulation & Results

In this section, the APF and the improved HAF method will be studied by simulating the obstacle avoidance approaches using the Gazebo simulator environment. The objective of the simulations is to research under which conditions the APF approach would experience the situation described in section 4-1-1. Subsequently, the improved HAF algorithm developed in section 4-2 is analyzed under the same conditions in simulation as the APF approach. It is then researched whether the HAF algorithm can prevent the same detrimental situation from occurring.

All obstacle avoidance simulations are performed using the same two source seeking controllers as applied in section 3-4 and applying the objective function in Eq. (3-1.5) under Assumption 3-1.1. In Figure 4-10, the simulation set-up is shown for performing the obstacle avoidance experiments. Where similar to section 3-3 the source is located at $x : 0[m]$, $y : 0[m]$ (red star). The obstacle is placed in the simulation environment at $x : 3[m]$, $y : 3[m]$ (blue), which is assumed to be a cylinder with a radius of $0.5[m]$. The initial position of the ROSbot is set at $x : 6[m]$, $y : 6[m]$, and the initial heading pointed towards the source. Using this set-up, the obstacle is aligned with the source and ROSbot. Thereby, if the ROSbot would follow the gradient of the signal field produced by the objective function in Eq. (3-1.5), it would crash.

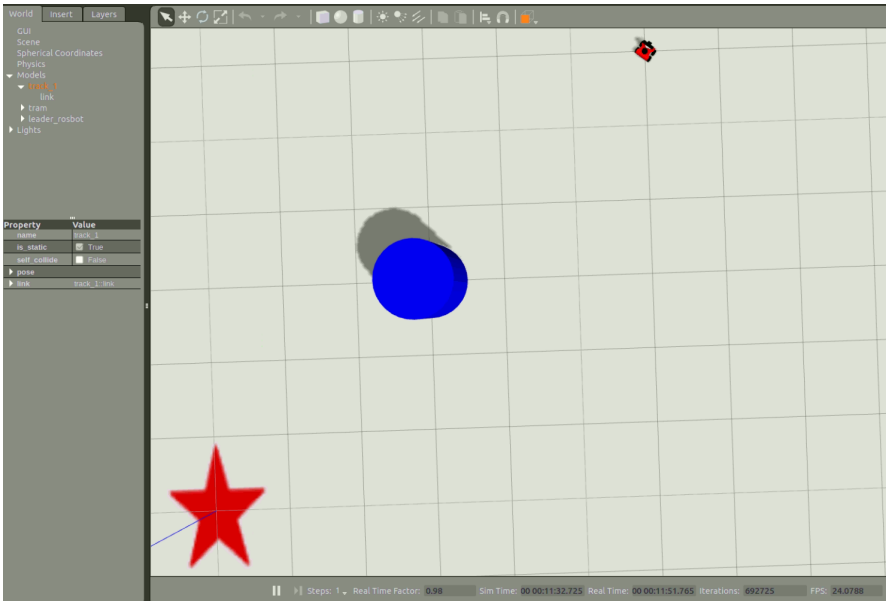



**Figure 4-10:** Figure showing the Gazebo simulation environment, with the ROSbot located in the top-right corner at $x : 6[m]$, $y : 6[m]$, the source at the red star at $x : 0[m]$, $y : 0[m]$ and obstacle at $x : 3[m]$, $y : 3[m]$ (blue) with a radius of $0.5$

First, it is confirmed that for both the fixed and oscillating sensor source seeking approach, the ROSbot would crash if no obstacle avoidance method would be employed, as shown in Figure B-1. Therefore, based on the simulations shown in Figure B-1, the need for an obstacle avoidance method to be included in the source seeking algorithm is evident.

Since, in section 4-2 a HAF obstacle avoidance method is developed on the premise of a detrimental line $M$, as illustrated in Figure 4-2b, in case an APF based obstacle avoidance

approach is employed. The conditions under which the line $M$ would be followed by the ROSbot are studied. If under these conditions in simulation the HAF can be shown to prevent the line $M$ from being followed, it would justify employing the HAF algorithm instead of the conventional APF approach.

First, for both types of source seeking controllers, the parameter configurations for which the ROSbot would follow line $M$ in a simulation are investigated when using an APF based obstacle avoidance. The APF approach is designed as follows, a virtual "safety" circle of radius $h_{apf}$ is created around the detected obstacle. The relative distance between the ROSbot and the virtual circle $\rho_{apf}$ is used as an argument for the barrier function applied by the APF. In Figure 4-11 a depiction is given of the designed artificial potential method.

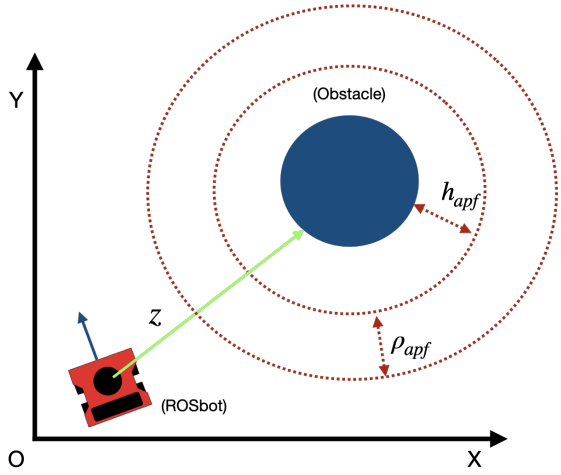


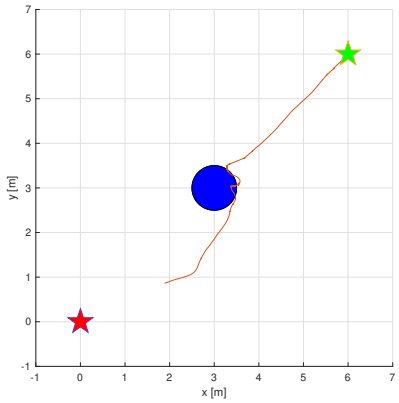

**Figure 4-11:** Scheme showing how the artificial potential function based obstacle avoidance method is implemented.

To determine the obstacle's radius and relative distance, the LiDAR on the ROSbot is used as described in section 4-2-1. Several parameters are of influence when acquiring a position and radius estimation of an obstacle. The first one being the noise affecting the simulated LiDAR rays in Gazebo and the sensor update rate of the LiDAR. The noise influencing the LiDAR rays is modeled as zero-mean white Gaussian with 0.01 standard deviation, which is the default setting provided by Husarion Gazebo model [11] for the ROSbot. Besides the noise on the LiDAR rays, also the estimation of the obstacle's position and radius by the *obstacle_detector*-package can be tuned to improve performance. The parameters for the *obstacle_detector*-package are set to values suggested by the authors in [21]. Where the measurement and process covariances $R$ and $Q$ are set to $R = 1$ and $Q = 0.01$. A process variance rate for the Kalman filter of 0.1 is chosen and a loop rate of 100 Hz to publish detected obstacles to the */obstacles*-topic which is the same as the publishing rate of the */signal_strength*-topic as described in section 3-2. The update rate for the LiDAR is set to 10 Hz, similar to the conducted experiment with a moving LiDAR in [21] and recommended by the LiDAR manufacturer [28].

An APF based obstacle avoidance method using the parameters listed in Figure 4-12c is applied on a ROSbot and simulated with the fixed virtual sensor approach. However, the scenario when the ROSbot would converge to line $M$ and get trapped behind an obstacle or

crash is difficult to reproduce in simulation. As Figure B-2 shows, where 20 simulation are performed. In all of the 20 simulation runs, the ROSbot is capable of avoiding the obstacle successfully.

For the oscillating sensor approach though the line $M$ is occurring in simulation as Figure B-3 shows when using an APF obstacle avoidance method. Besides, the APF approach illustrated in Figure 4-11 also a virtual box approach is simulated in Figure B-4. The virtual box is utilized to evaluate whether applying the APF method in an environment containing an obstacle with different geometry would also fail using the same barrier function parameters. Using the virtual box obstacle, the ROSbot is also not capable of sufficiently steering away from the obstacle and enters the "safety" box 2 out of the 20 simulation runs. In Figure 4-12, one simulation is shown where the APF would fail because of the situation described in section 4-1-1.




**(a)** Single trajectory of the ROSbot's geometric center, where ROSbot crashes into the obstacle while using an APF approach.

| Parameter | Value | Unit (SI) |
|---|---|---|
| $\omega$ | 20 | $rad$ |
| $h$ | 1 | unitless |
| $c$ | 80 | unitless |
| $\alpha$ | 1.8 | unitless |
| $R$ | 0.1 | $m$ |
| $b$ | 0 | unitless |
| $V_c$ | 0.1 | $m/s$ |

**(b)** Source seeking parameters

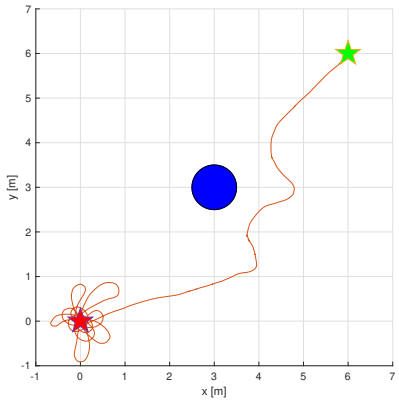
| Parameter | Value | Unit (SI) |
|---|---|---|
| $\rho_{apf}$ | 0.75 | $m$ |
| $h_{apf}$ | 0.5 | $m$ |

**(c)** Barrier function parameters

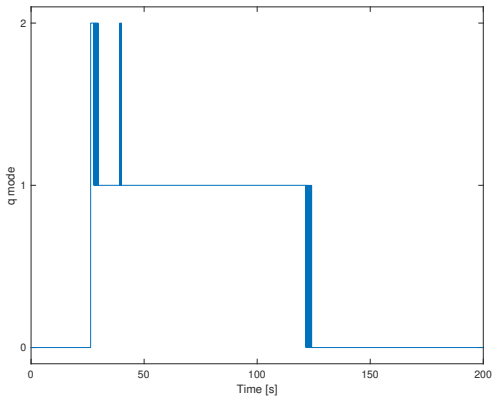
**Figure 4-12:** Simulation showing one trajectory using an oscillating sensor source seeking approach, when using the APF obstacle avoidance method the ROSbot is not capable of steering away from the detrimental line $M$.

As shown in Figure 4-12, the ROSbot crashes into the obstacle when using APF using the oscillating sensor approach. Therefore, having a hybrid adaptive obstacle avoidance method should, in theory, be beneficial to prevent such a crash. To evaluate whether the hybrid adaptive approach can steer the ROSbot away from line $M$, 20 simulations are performed using the same parameter configuration to define the repulsion barrier function as for the artificial potential approach. In Figure B-5, the simulated trajectories are shown of the ROSbot avoiding the obstacle while equipped with an oscillating sensor. One simulation run is depicted in Figure 4-13, wherein Figure 4-13b the updated hybrid switch mode $q$ is shown whilst the ROSbot is steering away from the obstacle.

Besides performing simulations, orientating the initial heading of the ROSbot straight towards the obstacle and aligning the initial position of the ROSbot w.r.t. the obstacle and source with an $\frac{\pi}{4}$ angle. The hybrid adaptive obstacle avoidance method is also tested by choosing from a random uniform distribution the initial heading and position, as illustrated in Figure 4-14b. The area from where the ROSbot is initialized is shown in green, which is at an $\frac{\pi}{4}$ angle to




**(a)** Single trajectory of the ROSbot's geometric center




**(b)** Hybrid mode q

| Parameter | Value | Unit (SI) |
|---|---|---|
| $\omega$ | 20 | $rad$ |
| $h$ | 1 | unitless |
| $c$ | 80 | unitless |
| $\alpha$ | 1.8 | unitless |
| $R$ | 0.1 | $m$ |
| $b$ | 0 | unitless |
| $V_c$ | 0.1 | $m/s$ |

**(c)** Source seeking parameters

| Parameter | Value | Unit (SI) |
|---|---|---|
| $\kappa$ | 3 | $m$ |
| $(\mu - \lambda)$ | $1 * 10^4$ | $unitless$ |
| $N$ | 2500 | $unitless$ |

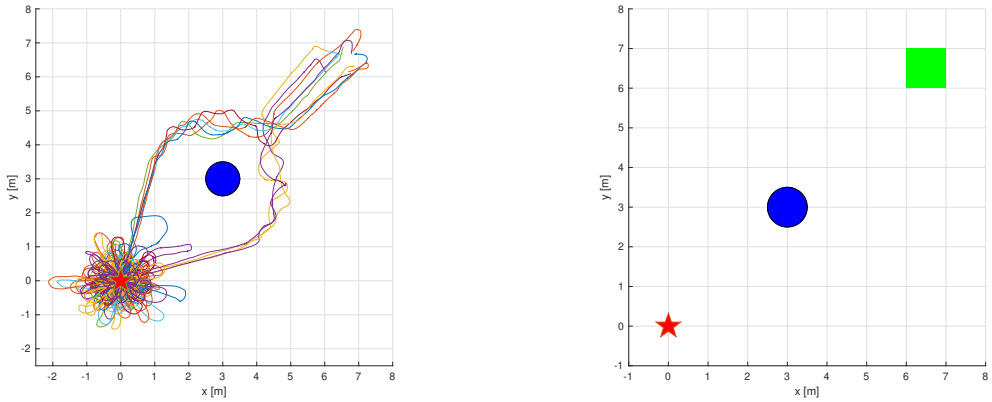
**(d)** Hybrid adaptive parameters

| Parameter | Value | Unit (SI) |
|---|---|---|
| $\rho$ | 0.75 | $m$ |
| $h$ | 0.5 | $m$ |

**(e)** Barrier function parameters

**Figure 4-13:** Simulation showing one trajectory, where using the hybrid adaptive feedback approach is capable of steering the ROSbot away from line $M$. With switch state graph and table with hybrid parameters

generate as many trajectories as possible, leading to the ROSbot getting trapped behind or crashing into the obstacle. The initial heading is chosen from a uniform random distribution defined by $[-\pi, \pi]$.




**(a)** 10 trajectories of the ROSbot's geometric center **(b)** Setup

**Figure 4-14:** Simulation showing trajectories of the ROSbot using an oscillating sensor source seeking approach, where the initial position is chosen from a uniform random distribution of the area shown in green and the heading from a uniform random distribution defined by $[-\pi, \pi]$. The parameter configurations shown in Figure 4-13 are used.

## 4-4 Conclusion

The detrimental scenario described by the authors in [19] and shown in Figure 4-2, is difficult to reproduce in simulations when using a unicycle vehicle performing perturbation based source seeking. Especially for the fixed sensor source seeking approach due to the sensor's movement coupled to the vehicle's movement. Because the source seeking controller uses a sensor at $R > 0$ distance away from the ROSbot's geometric center and the added $\alpha\omega \cos \omega kT$ perturbation term acts directly on the angular velocity of the ROSbot, so when the ROSbot approaches a local minimum the added perturbation causes the ROSbot to steer away. For the oscillating sensor source seeking approach, it was possible to let the ROSbot converge to line $M$ if the forward velocity was set to a constant by reducing the derivative $b$ term to zero. Thereby, only the angular velocity of the ROSbot is controlled by the source seeking algorithm.

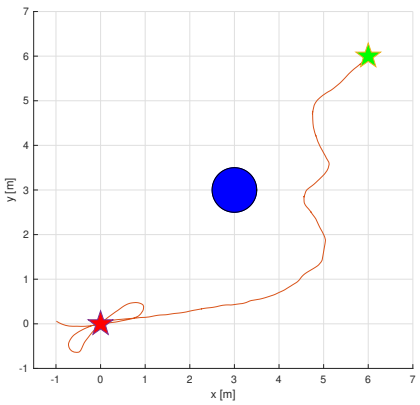
Hence, based on the observation that in simulation, the detrimental scenario described by [19] exists when using an oscillating sensor source seeking approach for a unicycle. It merits using a HAF obstacle method to prevent the ROSbot from following line $M$. The developed HAF obstacle avoidance approach is empirically not shown to suffer from crashing into the obstacle due to the existence of line $M$, as is shown in the simulation of Figure 4-13. As can be seen from the hybrid switch mode graph in Figure 4-13b, the HAF algorithm initially rapidly switches between states $q = 1$ and $q = 2$. If this rapid switching is undesirable, the $\mu$ and $\rho$ parameters can be increased to prevent the rapid switching between states. In Figure 4-15, a simulation run is shown with a 10 times larger $\mu$ and 0.5 $m$ longer $\rho$ w.r.t. the simulation in Figure 4-13. The hybrid switch mode graph in Figure 4-15b does not rapidly toggle between states $q = 1$ and $q = 2$ anymore.

Although the fixed sensor approach the pathology shown in Figure 4-2 was not reproducible when applying an artificial potential avoidance approach in simulation, the hybrid adaptive obstacle avoidance algorithm was also tested in Figure B-6. The HAF obstacle avoidance

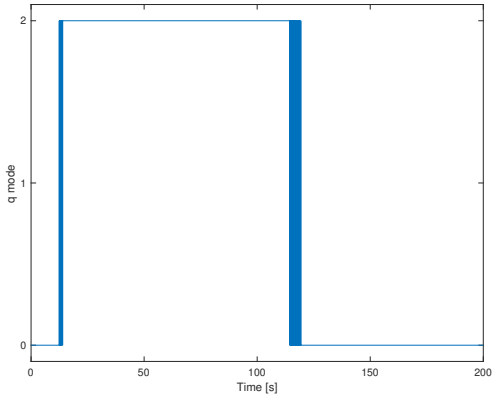
algorithm is shown to be capable of avoiding the obstacle when employed on the fixed sensor source seeking ROSbot. Although due to the looping source seeking trajectory of a fixed sensor unicycle, the path around the obstacle is not as smooth as the oscillating sensor approach.

The HAF obstacle avoidance algorithm used in this research is improved upon the approach of [19], where the location and relative location of the obstacle w.r.t. were assumed to be known *a priori* to control algorithm. Whereas, the hybrid adaptive avoidance approach developed in this thesis does not require these assumptions. Although in [19] the authors where capable of guaranteeing robustness, the assumptions the authors make are inconsistent when considering the objective of a perturbation based source seeking algorithm to be a gradient-free extremum seeking method. In [17], the authors make a similar observation and align the partitioning lines directly with the estimated gradient produced by a distributed source seeking algorithm. However, the perturbation-based source seeking method in this thesis is performed by a single-vehicle; therefore, a low-pass filter is added to slow down the rotation of the partitioning lines as ROSbot turns to steer away from the obstacle.

Based on the simulations, it was observed that the update rate of the LiDAR affects the performance of the hybrid adaptive obstacle avoidance method significantly. As also stated by the authors in [21], that the main factor for attenuation of noise being the update rate of the LiDAR. However, the recommended update rate of 10 Hz by the manufacturer [28] was found to be sufficient.




**(a)** Single trajectory of the ROSbot's geometric center




**(b)** Hybrid mode q

| Parameter | Value | Unit (SI) |
|---|---|---|
| $\omega$ | 20 | $rad$ |
| $h$ | 1 | unitless |
| $c$ | 80 | unitless |
| $\alpha$ | 1.8 | unitless |
| $R$ | 0.1 | $m$ |
| $b$ | 0 | unitless |
| $V_c$ | 0.1 | $m/s$ |

**(c)** Source seeking parameters

| Parameter | Value | Unit (SI) |
|---|---|---|
| $\kappa$ | 3 | $m$ |
| $(\mu - \lambda)$ | $1 * 10^5$ | $unitless$ |
| $N$ | 2500 | $unitless$ |

**(d)** Hybrid adaptive parameters

| Parameter | Value | Unit (SI) |
|---|---|---|
| $\rho$ | 1.25 | $m$ |
| $h$ | 0.5 | $m$ |

**(e)** Barrier function parameters

**Figure 4-15:** Simulation showing one trajectory, where using the HAF approach is capable of steering the ROSbot away from line $M$. The $\rho$ and $\mu$ parameters are increased w.r.t. the simulation in Figure 4-13 to prevent rapid switching between states $q = 1$ and $q = 2$.

# Chapter 5

# Formation control

Besides the objectives of source seeking and obstacle avoidance discussed in Chapters 3 and 4, the combined control algorithm should also be capable of navigating multiple robotic vehicles in a formation as was stated in Chapter 1. In this Chapter, an approach for including formation control to the algorithm is described whilst performing source seeking and obstacle avoidance. The formation considered in this chapter consists of three ROSbots that need to achieve these objectives. First, in section 5-1, a decentralized leader-follower formation control strategy will be introduced. One vehicle is designated as a leader vehicle to navigate towards a target, and the other two vehicles need to meet the formation control objectives. Thereafter, the multi-vehicle system's path planning method will be elaborated on in section 5-2. Then, in section 5-3 the implementation of the formation control strategy and formation path planning method on the ROSbot platform will be outlined. Subsequently, the simulation and results of the formation control algorithm are shown in section 5-4. Finally, in section 5-5 the results will be discussed.

## 5-1 Leader-Follower Strategy

A formation control strategy aims to maintain a desired formation shape and behavior of the multi-vehicle system. According to the authors in [12], the methods for achieving this can be categorized into three approaches, namely a *leader-follower*, *virtual structure* or *behavior-based* approach.

From these three different strategies, the leader-follower approach is best suited for this thesis research. Because the formation control algorithm should preferably not deteriorate the source seeking and obstacle avoidance performance of the combined algorithm established in sections 3-4 and 4-4. A leader-follower control method designates one vehicle to be the leader who acts as a group reference for the multi-vehicle system. The leader vehicle in the formation is responsible for acquiring all necessary information needed for navigating the multi-vehicle system. The other vehicles in the multi-vehicle system are considered to be followers. In the leader-follower structure, the followers' main goal is to operate under

the leader vehicle's guidance and maintain the multi-vehicle system's formation shape. The leader-follower strategy is chosen where the leader vehicle employs the combined control algorithm described in section 4-4 and the two follower vehicles are responsible for formation control.

The formation shape is maintained by controlling the desired relative angle and distance between the leader and the follower vehicles. To achieve this, the follower vehicles use a feedback controller to determine the appropriate linear and angular velocity for driving the error limits in Eq. (5-1.1) to zero. The error is computed by acquiring the current length $z$ and angle $\psi$ w.r.t. the leader and subtracting it with the desired length $z_d$ and angle $\psi_d$ w.r.t. the leader. In Figure 5-1, such a leader-follower scheme is illustrated with one leader and one follower.

$$\lim_{t\to\infty}(z - z_d) = 0 \tag{5-1.1a}$$

$$\lim_{t\to\infty}(\psi - \psi_d) = 0 \tag{5-1.1b}$$

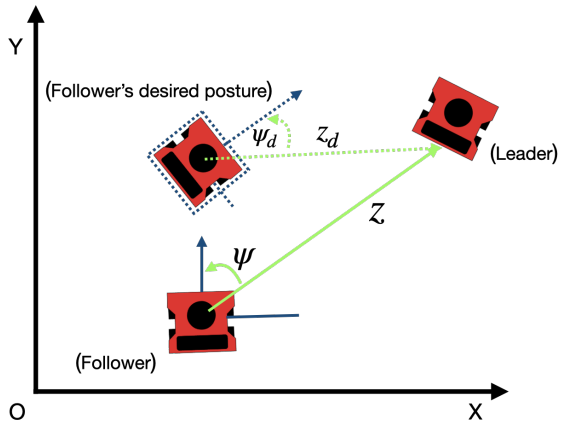



**Figure 5-1:** A figure depicting the control objective of the leader-follower formation scheme, where the follower vehicle's goal is maintain a desired posture w.r.t. the leader by reducing Eq. (5-1.1) to zero.

## 5-2 Follower Path Planning

To control for the relative distance $z$ and bearing angle $\psi$ between the follower and leader vehicle, the control law needs to use the forward and angular velocity $(v, \omega)$ as inputs. The control law of the follower vehicle is restricted to these velocity inputs due to non-holonomic constraints of ROSbot vehicle, as described in section 2-2. To construct the follower algorithm, an approach is chosen considered by [27] to be a reactive approach. A reactive approach is where the environment is only partially known to the vehicle, so the algorithm can only determine a specific area's trajectory. It needs to update and regenerate a trajectory reactively

whilst it is moving. It is assumed that the leader vehicle moves autonomously through the 2-D space without communication with the follower robots.

The leader and follower vehicle's coordinate frames shown in Figure 5-1 are used as a model for the followers' control algorithm. Where the coordinates $\tilde{x}_{ij}$ and $\tilde{y}_{ij}$ express the relative distance of the leader vehicle $i$ w.r.t. the follower vehicle $j$ in the local $x$-$y$ coordinate frame of the follower. The values of these coordinates can be acquired by sensors equipped on the follower ROSbot, as further described in 2-3.

$$\begin{bmatrix} \tilde{x}_{ij} \\ \tilde{y}_{ij} \end{bmatrix} = \begin{bmatrix} \cos\theta_i & \sin\theta_i \\ -\sin\theta_i & \cos\theta_i \end{bmatrix} \begin{bmatrix} x_j - x_i \\ y_j - y_i \end{bmatrix} \tag{5-2.1}$$

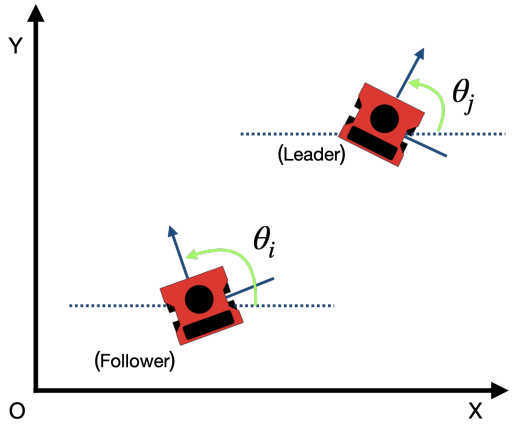



**Figure 5-2:** A figure illustrating how the heading $\theta$ of the leader and follower vehicle are defined w.r.t. the global $x$-$y$ coordinate frame

Using $\tilde{x}_{ij}$ and $\tilde{y}_{ij}$, the two variables $z$ and $\psi$ which the feedback controller of the follower vehicle tries to control for can be computed as described in Eq. (5-2.2). With $z$ being calculated by the Euclidean distance of the follower vehicle $i$ w.r.t. leader vehicle $j$ like stated in Eq. (5-2.2a). The bearing angle $\psi$ is measured in Eq. (5-2.2b) by utilizing the $atan2(y, x)$-function with the same reasoning as described in section 4-2-2 for distinguishing diametrically opposite directions. The bearing angle $\psi$ is the follower vehicle's orientation $i$ w.r.t. the leader vehicle $j$.

$$z = \sqrt{(\tilde{x}_{ij})^2 + (\tilde{y}_{ij})^2} \tag{5-2.2a}$$

$$\psi = \arctan2(\tilde{x}_{ij}, \tilde{y}_{ij}) \tag{5-2.2b}$$

In Eq. (5-2.3), the dynamics of $z$ and $\psi$ are expressed. Where the parameters $(\dot{r}_c)_F$ and $\dot{\theta}_F$ are defined as the maximum angular and forward velocity of the follower vehicle and $(\dot{r}_c)_L$ as the maximum forward velocity of the leader vehicle. These dynamics govern how the follower vehicle's feedback controller tries to maintain the desired $z_d$ and $\psi_d$.

$$\dot{z} = -(\dot{r}_c)_F \cos(\psi) + (\dot{r}_c)_L \cos(\theta_j - \theta_i - \psi) \tag{5-2.3a}$$

$$\dot{\psi} = -\dot{\theta}_F + \frac{(\dot{r}_c)_F}{z}\sin(\psi) + \frac{(\dot{r}_c)_L}{z}\sin(\theta_j - \theta_i - \psi) \tag{5-2.3b}$$

Following the work presented in [30], constraints are introduced on $z$ and $\psi$ to guarantee that the follower vehicle is capable of tracking the leader. Because the path planning algorithm should be suitable for the ROSbot platform, limitations of the sensor needed for measuring $z$ and $\psi$ need to be considered. The sensor limitations are taken into account by translating the constraints on $z$ and $\psi$ into barrier functions, as expressed in Eq. (5-2.4). Hence, ensuring that the follower vehicle tracks the leader within the defined sensor limits.

$$B_z(z) = \begin{cases} \beta_z \frac{(z-z_d)^2}{(z_{max}-z_d)^2}, & \text{if } z_d < z \leq z_{max} \\ \beta_z \frac{(z-z_d)^2}{(z_d-z_{min})^2}, & \text{if } z_{min} \leq z \leq z_d \end{cases} \tag{5-2.4a}$$

$$B_\psi(\psi) = \begin{cases} \beta_\psi \frac{(\psi-\psi_d)^2}{(\psi_{max}-\psi_d)^2}, & \text{if } \psi_d < \psi \leq \psi_{max} \\ \beta_\psi \frac{(\psi-\psi_d)^2}{(\psi_d-\psi_{min})^2}, & \text{if } \psi_{min} \leq \psi \leq \psi_d \end{cases} \tag{5-2.4b}$$

Where $[z_{min}, z_{max}]$ and $[\psi_{min}, \psi_{max}]$ define the bounds on relative distance and bearing angle of the follower w.r.t. the leader vehicle. The $\beta_z$ and $\beta_\psi$ are design parameters that are used to amplify the barrier value when the distance and/or bearing angle are close to their defined limits. The barrier functions Eq. (5-2.4a) and Eq. (5-2.4b) are used for the path planning control law drive the follower w.r.t. the leader vehicle under the defined constraints. It is assumed that the maximum forward velocity $(\dot{r}_c)_L$ is known to the follower vehicles and lower than the maximum forward velocity of the follower vehicles $(\dot{r}_c)_F$. Thereby ensuring that the follower vehicle capable of catching up with the leader. Using the barrier functions in Eq. (5-2.4) for the specified constraints, the control laws shown in Eq. (5-2.5) are constructed, which are based on the leader-tracking control laws in [30].

$$(\dot{r}_c)_i = \frac{1}{\cos(\psi)}(k_z \nabla B_z + (\dot{r}_c)_L \tanh\left(\frac{(\dot{r}_c)_L \nabla B_z}{\delta_z}\right)) \tag{5-2.5a}$$

$$\dot{\theta}_i = \frac{(\dot{r}_c)_i}{z}\sin(\psi) + (\frac{(\dot{r}_c)_L}{z})\tanh\left(\frac{(\dot{r}_c)_L \nabla B_\psi}{z\delta_\psi}\right) + k_\psi \nabla B_\psi \tag{5-2.5b}$$

In Eq. (5-2.5) $k_z$, $k_\psi$, $\delta_z$ and $\delta_\psi$ are constant positive parameters that have to satisfy the conditions described in Appendix C-1. $\nabla B_z$ and $\nabla B_\psi$ are gradients of barrier functions Eq. (5-2.4a) and Eq. (5-2.4b) w.r.t. their arguments $z$ and $\psi$.

### 5-2-1 Follower Obstacle Avoidance

Apart from controlling and maintaining the formation shape of the multi-vehicle system, the follower vehicles need to take other control objectives into account as well, namely:

- inter-vehicle collision avoidance
- obstacle avoidance

Hence, the control algorithm employed on the follower vehicles should also plan a path that meets these requirements. To achieve this, the research in [30] on formation control is utilized again. The obstacle avoidance method's basic principle is similar to the leader tracking approach described in section 5-2. The barrier functions shown in Eq. (5-2.4) for desired distance and bearing angle w.r.t. the leader vehicle are extended by an obstacle barrier function defined in Eq. (5-2.6).

$$B_{\text{obs}}(\psi_{\text{obs}}) = \begin{cases} \beta_{\text{obs}} \frac{(\psi_{\text{obs}} - \psi_{\text{obs},d})^2}{\psi_{\text{obs},d}^2}, & \text{if } 0 < \psi_{\text{obs}} \leq \psi_{\text{obs},d} \\ \beta_{\text{obs}} \frac{(\psi_{\text{obs}} + \psi_d)^2}{\psi_{\text{obs},d}^2}, & \text{if } -\psi_{\text{obs},d} \leq \psi_{\text{obs}} \leq 0 \end{cases} \tag{5-2.6}$$

The objective of obstacle barrier function Eq. (5-2.6) is to repel the follower vehicle away from the obstacle. The barrier function uses the orientation of the follower vehicle w.r.t. the obstacle ($\psi_{\text{obs}}$) as argument, similarly as defined in Eq. (5-2.4a) for $z$ and $\psi$. Where $\psi_{\text{obs}d}$ in Eq. (5-2.6), is defined as the critical bearing angle. The obstacle avoidance control law's objective is to steer the vehicle away from $\psi_{\text{obs}d}$. The obstacle avoidance method's principle is based on partitioning the space around the follower vehicle in regions as illustrated in Figure 5-3.

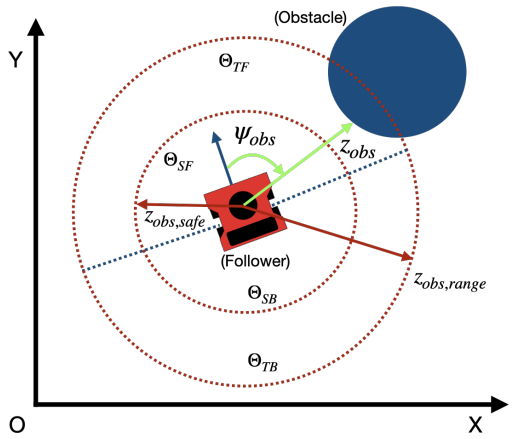



**Figure 5-3:** A figure depicting how to space around the follower vehicle is partitioned as defined in Eq. (5-2.7).

The regions shown in Figure 5-3 are determined by the relative distance between the follower vehicle and obstacle $z_{obs}$ and the bearing angle $\psi_{obs}$ w.r.t. the obstacle. In Eq. (5-2.7), the boundaries for these four regions are defined.

$$\Theta_{\text{TF}} \triangleq \{z_{\text{obs, safe}} < z_{\text{obs}} \leq z_{\text{obs, range}}, \quad |\psi_{\text{obs}}| \leq \frac{\pi}{2}\} \tag{5-2.7a}$$

$$\Theta_{\text{TB}} \triangleq \{z_{\text{obs, safe}} < z_{\text{obs}} \leq z_{\text{obs, range}}, \quad |\psi_{\text{obs}}| > \frac{\pi}{2}\} \tag{5-2.7b}$$

$$\Theta_{\text{SF}} \triangleq \{z_{\text{obs}} \leq z_{\text{obs, safe}}, \quad |\psi_{\text{obs}}| \leq \frac{\pi}{2}\} \tag{5-2.7c}$$

$$\Theta_{\text{SB}} \triangleq \{z_{\text{obs}} \leq z_{\text{obs, safe}}, \quad |\psi_{\text{obs}}| > \frac{\pi}{2}\} \tag{5-2.7d}$$

Where $z_{\text{obs, range}}$ is used to define the distance where the obstacle avoidance method begins affecting the follower vehicle's control law. The $z_{\text{obs, safe}}$ is the distance where the follower vehicle should stop tracking the leader vehicle and steer away from the obstacle to prevent a potential crash.

In Algorithm 2, the implemented obstacle avoidance approach's logic is presented. The approach is adjusted compared to the logic published in [30] so that the obstacle avoidance method is suitable to combine with tracking a leader that is performing source seeking. Since perturbation-based source seeking produces a looping trajectory of the leader vehicle, as further described in Chapter 3. In case an obstacle is in the region $\Theta_{\text{SF}}$ or $\Theta_{\text{SB}}$, the vehicle stops leader tracking, and all control action is devoted to navigating the vehicle away from the obstacle. Because of the leader vehicle's possible looping trajectory, the follower vehicle should also be enforced to drive away from the obstacle when the obstacle is in $\Theta_{\text{SB}}$.

**Algorithm 2** Obstacle Avoidance Algorithm Logic

**Input:** $z_{\text{obs}}$, $\psi_{\text{obs}}$
**Output:** $(\dot{r}_c)_i$, $\dot{\theta}_i$
  **function** AVOIDANCEMODE($\Theta_{\text{mode}}$)
    **if** $\Theta_{\text{TF}}$ **then**
      $(\dot{r}_c)_i = (\dot{r}_c)_{i,\text{track}}$, $\dot{\theta}_i = \min\{\dot{\theta}_{i,\text{track}} + \dot{\theta}_{i,\text{avoid}}, \dot{\theta}_{\text{sat}}\}$
    **else if** $\Theta_{\text{TB}}$ **then**
      $(\dot{r}_c)_i = (\dot{r}_c)_{i,\text{track}}$, $\dot{\theta}_i = \dot{\theta}_{i,\text{track}}$
    **else if** $\Theta_{\text{SF}}$ **then**
      $(\dot{r}_c)_i = (\dot{r}_c)_{i,\text{avoid}}$, $\dot{\theta}_i = -\dot{\theta}_{i,\text{avoid}}$
    **else if** $\Theta_{\text{SB}}$ **then**
      $(\dot{r}_c)_i = (\dot{r}_c)_{i,\text{avoid}}$, $\dot{\theta}_i = \dot{\theta}_{i,\text{avoid}}$
    **else**
      $(\dot{r}_c)_i = (\dot{r}_c)_{i,\text{track}}$, $\dot{\theta}_i = \dot{\theta}_{i,\text{track}}$
    **end if**
  **end function**

The control laws for steering a follower vehicle away from an obstacle are expressed in Eq. (5-2.8). Where the forward velocity of the ROSbot in Eq. (5-2.8a) is set constant in case, the follower vehicle is within $z_{\text{obs, safe}}$ distance of the obstacle. In Eq. (5-2.8b), the angular velocity control law is defined. The objective of Eq. (5-2.8b) is to guide the vehicle away from the obstacle.

$$(\dot{r}_c)_{i,\text{avoid}} = v_{\text{avoid}} \tag{5-2.8a}$$

$$\dot{\theta}_{i,\text{avoid}} = k_{\text{obs}} \nabla B_{\text{obs}} \tag{5-2.8b}$$

Similar as for the leader tracking control laws in Eq. (5-2.5), the $k_{\text{obs}}$ in Eq. (5-2.8b) is determined to be a positive constant that has to satisfy the conditions further elaborated on in Appendix C-2. The $\nabla B_{\text{obs}}$ in Eq. (5-2.8b) is the gradient of barrier function Eq. (5-2.6) w.r.t. to the bearing angle $\psi_{\text{obs}}$ between the ROSbot and obstacle.

### 5-2-2 Inter-Vehicle Collision Avoidance

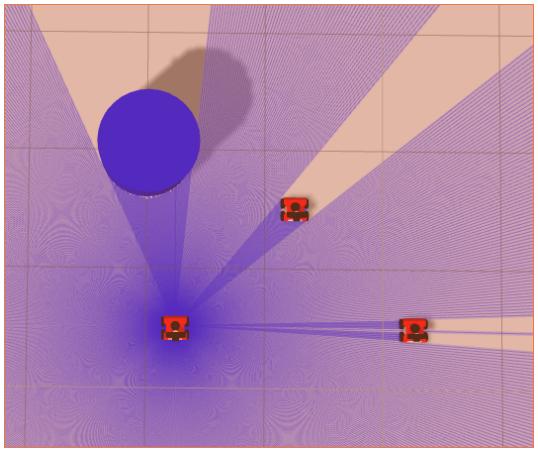
The final objective of the formation controller algorithm is to prevent collisions between follower vehicles. The logic for collision avoidance between follower vehicles is based on the obstacle avoidance approach. When a follower ROSbot detects another follower vehicle within $z_{col,safe}$ distance, the collision avoidance treats the other follower as an obstacle. The forward velocity of the vehicle is slowed down to $(\dot{r}_c)_{col}$ and the angular velocity $\dot{\theta}_{col}$ used to steer both followers away from each other. The $\dot{\theta}_{col}$ is computed similarly as for obstacle avoidance using a barrier function.

## 5-3 Implementation

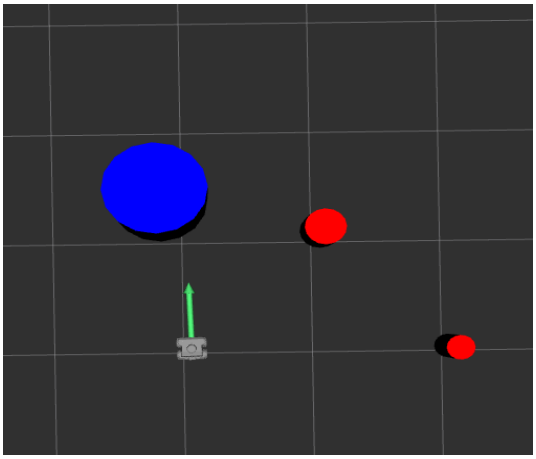
To employ the algorithm described in section 5-2. First, an approach is developed that allows the ROSbot vehicle to differentiate an obstacle from other ROSbot vehicles in the formation. Therefore, an assumption is made similar as for the experimental set-up described in section 4-3 where the obstacle is assumed to be a cylinder with a radius of $0.5[m]$. Because the dimensions of the ROSbot are significantly smaller, as explained in section 2-3, the Light Detection And Ranging (LiDAR) sensor equipped on the ROSbot can be utilized to discriminate between a ROSbot and an obstacle. To acquire information of nearby objects with the LiDAR, the *obstacle_detector*-package developed by [21] is used. As further elaborated on in section 4-2-1, detected objects' geometry is approximated by a circle consisting of a center and radius. Subsequently, all detected objects published to the */objects*-topic with their corresponding radius and relative distance in $[\tilde{x}_{ij}, \tilde{y}_{ij}]^T$, see Eq. (5-2.1).

Next, the follower vehicle's formation controller algorithm subscribes to the */objects*-topic. Where detected objects with a radius larger than $r_{thres}$ are classified as an obstacle and objects with a radius smaller than $r_{thres}$ as fellow ROSbot vehicles. Objects classified as fellow ROSbot vehicles are listed, with for each object their respective bearing angle $\psi$ and distance $z$ w.r.t. the follower vehicle. Based on the leader tracking algorithm, one object is designated as the leader vehicle, and the other listed objects as fellow follower vehicles. Lastly, the formation controller computes the appropriate control inputs. Subsequently, the control inputs are published to the */cmd_vel*-topic on which the ROSbot firmware is subscribed. In Figure 5-5, a scheme is shown to illustrate how the described ROS nodes communicate using the topics on the follower ROSbot vehicles.

After establishing the communication framework of the ROS nodes, as displayed in Figure 5-5, two different formation shapes are chosen in Figure 5-6. Namely, a triangle shaped formation



**(a)** A figure showing a simulation scenario in Gazebo, where the ROSbot emits LiDAR rays visible in blue



**(b)** Obtained LiDAR points of the surrounding objects are estimated into circular objects by the *obstacle_detector*-package and classified based on their radius into either an obstacle (blue) or a vehicle (red).

**Figure 5-4:** A simulation scenario showing how the *obstacle_detector*-package would convert LiDAR data into an detected objects, when the formation controller would try to maintain a triangle shape.

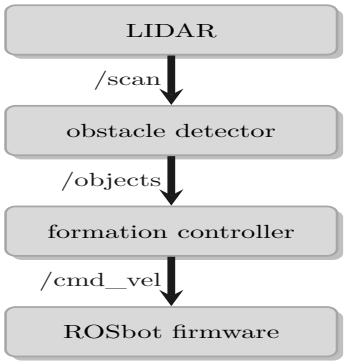
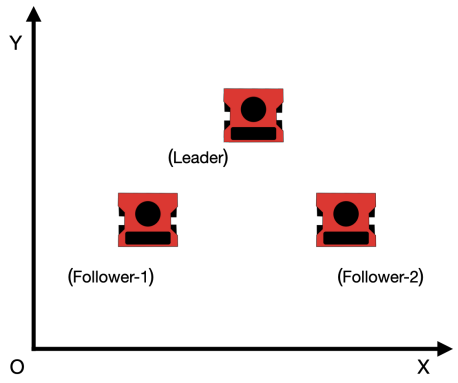
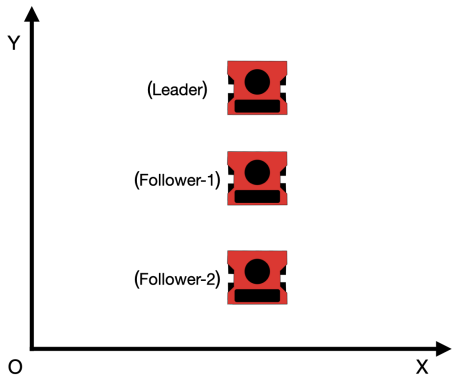



**Figure 5-5:** Communication scheme between ROS nodes

see Figure 5-6a and a snake-like formation shown in Figure 5-6b. Therefore, in section 5-4, both are simulated to investigate the developed formation controller's flexibility.




**(a)** A triangle shaped formation for the multi vehicle system




**(b)** A snake-like shaped formation for the multi vehicle system

**Figure 5-6:** Formation shapes for the multi vehicle system

## 5-4 Simulation & Results

To demonstrate the decentralized formation control algorithm's effectiveness, the formation control algorithm is simulated using a source seeking leader vehicle with both a fixed and oscillating sensor. First, the algorithm is studied when no obstacles are present in the simulation environment in section 5-4-1. The follower vehicles' tracking performance is determined by how the follower vehicles are capable of maintaining the desired angle $\psi_d$ and distance $z_d$ w.r.t. the leader vehicle. The tracking performance is defined based on the integral square error (ISE) term defined in Eq. (5-4.1).

$$ISE(z_i, \psi_i) = \int_{t=t_0}^{t=t_{end}} \sum (\psi_d - \psi_i)^2 + (z_d - z_i)^2 dt \tag{5-4.1}$$

The $z_i$ and $\psi_i$ arguments are computed based on the simulated Odometry data. Hence, the ISE is computed without noise and not based on LiDAR data, which is used as input for the follower formation control algorithm. Besides evaluating the trajectory and ISE of the formation, the relative distance $z$ and angle $\psi$ w.r.t. the leader vehicle based on LiDAR data are examined to evaluate the effect of tuning the parameters of the formation control algorithm.

After evaluating how the formation control parameters affect the tracking performance, in section 5-4-2, the obstacle avoidance method of the formation control algorithm is tested. An obstacle is placed in the simulation environment at $x : 3[m]$, $y : 3[m]$ (blue), which is assumed to be a cylinder with a radius of $0.5[m]$. The obstacle is placed at this location to interfere with the formation's path towards the source as much as possible, similar to the simulation set-up used in section 4-3.

In the plots showing the simulated trajectories of the formation in Appendix C-3 and C-4, the leader vehicle depicted in red, the first follower vehicle in green, and the second follower vehicle in magenta.

### 5-4-1 Obstacle-free

First, the formation control algorithm is studied in an obstacle-free environment to find an appropriate parameter configuration for tracking the source seeking leader vehicle. In Appendix C-3, both the formation shapes of Figure 5-6 are simulated with leader vehicle using a fixed and oscillating sensor.

### 5-4-2 Obstacle

After tuning the formation control algorithm for solely leader tracking in Appendix C-3, an obstacle is placed in the environment on the same location and dimensions as in section 4-3. By placing an obstacle in the environment, the formation control algorithm's obstacle avoidance performance is examined. In Appendix C-4, simulations are shown for both formation shapes and types of source seeking leader vehicles. The control algorithm's performance is analyzed by determining whether the obstacle was successfully avoided, and the formation shape was recovered afterward. To measure whether the algorithm can recover the formation shape, the ISE of the follower vehicles is examined if the growth rate of the ISE is reduced after avoiding the obstacle. A red shade is added to the perceived bearing angle $\psi$ and relative distance $z$ graphs to highlight when the follower vehicles' obstacle avoidance algorithm is activated.

## 5-5 Discussion

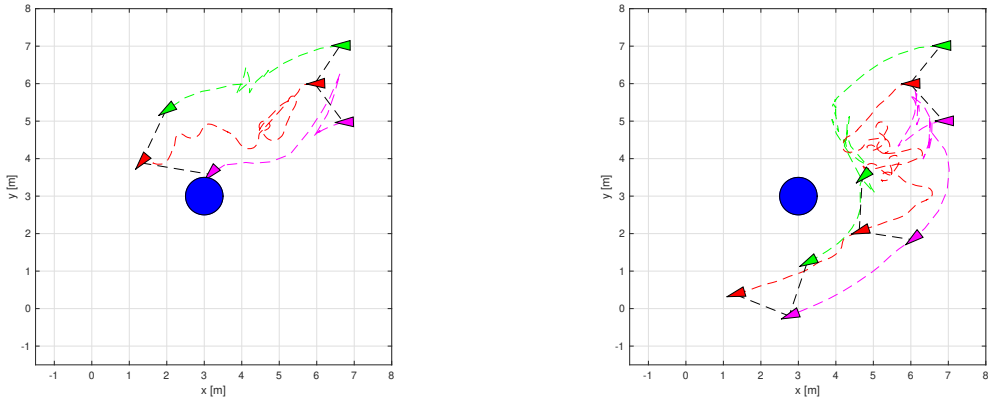
In section 5-1 and 5-2 objectives of the formation control algorithm are specified that will be used to conclude whether the implemented algorithm is successful. These objectives of the formation control algorithm are the following:

- Maintaining formation shape
- Obstacle avoidance
- Collision avoidance

Where obstacle and collision avoidance objectives should always be satisfied by the formation control algorithm, the followers' ability to maintain formation shape w.r.t. the leader is compared using the ISE of the follower vehicles. In case no obstacle is placed in the simulation environment, the formation controller is capable of maintaining formation shape for both the fixed and oscillating source seeking leader vehicle, as can be seen from the simulations shown in Appendix C-3. If a snake-like formation shape is chosen, the first follower vehicle tracking the leader vehicle has a significantly higher ISE compared to the second follower vehicle that is tracking the first follower vehicle, as can be seen in Figure C-3 and Figure C-4. Because the source seeking leader vehicle's trajectory towards the source is less smooth

than the first follower vehicle's trajectory. Therefore, in case a triangle-shaped formation is selected the ISE of both follower vehicles is similar since both vehicle track the leader vehicle directly, as illustrated in Figure C-1 and Figure C-2. Hence, in an obstacle-free environment, the formation control algorithm satisfies leader tracking and formation shape maintenance requirements.

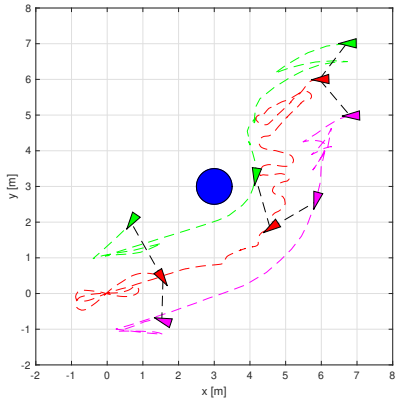
However, when an obstacle is introduced in the environment, the formation control problem becomes more challenging. Because the follower vehicle needs to balance adjusting its heading to avoid an obstacle and tracking the leader vehicle with the desired angle $\psi_d$, this balance poses especially a challenge for the triangle formation shape (Figure 5-6a) as one of the follower vehicles in the formation will require a significant steering adjustment to avoid crashing into the obstacle. Nevertheless, a too aggressive steering correction is undesirable since it could cause the follower vehicle to lose track of the leader vehicle because the leader vehicle is no longer within the boundaries set for the relative distance $z$ and bearing angle $\psi$. Another possibility that could occur due to too much steering adjustment is that the other follower vehicle could be misleadingly considered the leader vehicle as it suddenly is closest to the defined desired bearing angle $\psi_d$. The angular velocity produced by the follower obstacle avoidance algorithm is tuned using $k_{obs}$. Where a too low $k_{obs}$ results in the follower vehicle crashing as shown in Figure 5-7a and a too high $k_{obs}$ loosing track of the leader vehicle as depicted in Figure 5-7b.



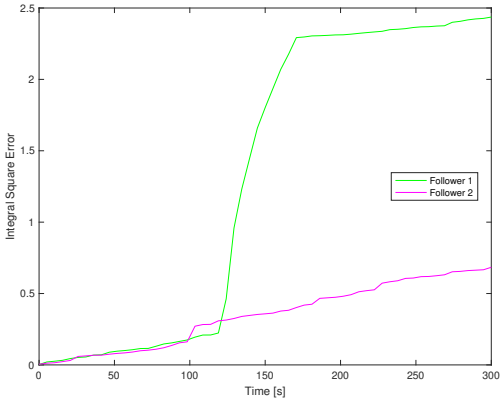

**(a)** Low $k_{obs} = 0.1$ **(b)** High $k_{obs} = 2$

**Figure 5-7:** Simulation showing the consequence of not applying enough steering adjustment in Figure 5-7a when too aggressive steering is applied in Figure 5-7b. As a result, the follower vehicle fails in leader tracking and considers the other follower vehicle as leader.

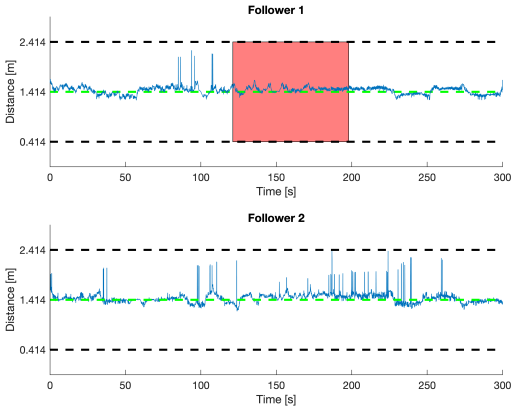
Using an appropriately tuned $k_{obs}$, the follower vehicle adjusts its steering without losing track of the leader vehicle. In Figure 5-8, a simulation is run using $k_{obs} = 0.6$ wherefrom the plot in Figure 5-8d it can be seen that the follower vehicle adjusts its heading significantly to avoid crashing into the obstacle. The ISE of the follower vehicle suffers from the steering correction as shown in Figure 5-8b which illustrates the balance between leader tracking and obstacle avoidance discussed above.



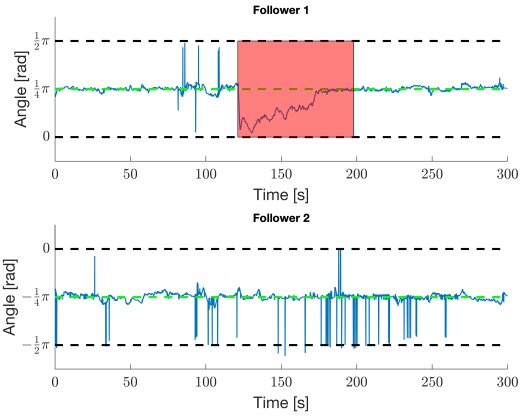

**(a)** Trajectories of ROSbot formation




**(b)** Integral Square Error of formation




**(c)** Distance w.r.t. leader vehicle




**(d)** Angle w.r.t. leader vehicle

**Figure 5-8:** Simulation of formation control algorithm with a leader vehicle using an oscillating sensor for source seeking, the parameters listed in table 5-1.

**Table 5-1:** Formation control parameters

| Parameter | Value | Unit (SI) |
|---|---|---|
| $\psi$ | $0.25\pi$ | $rad$ |
| $z$ | 1.414 | $m$ |
| $\psi_{range}$ | $0.25\pi$ | $rad$ |
| $z_{range}$ | 1 | $m$ |
| $\beta_z$ | 2.5 | $unitless$ |
| $\beta_\psi$ | 1.35 | $unitless$ |
| $k_z$ | 0.01 | $unitless$ |
| $\delta_z$ | 0.6 | $unitless$ |
| $k_\psi$ | 0.6 | $unitless$ |
| $\delta_\psi$ | 0.1 | $unitless$ |
| $\psi_{obs}$ | $0.5\pi$ | $rad$ |
| $k_{obs}$ | 0.6 | $unitless$ |
| $\beta_{obs}$ | 0.75 | $unitless$ |

For the snake-like formation shape (Figure 5-6b), obstacle avoidance is less difficult due to the leader vehicle being followed with the desired angle of $\psi_d = 0^o$. Therefore, requiring fewer adjustments to the heading of the follower vehicles. As a result, more emphasis can be placed on tracking the leader vehicle, thereby improving the formation's shape and maintaining the algorithm's performance. Therefore, the ISE plots of a snake-like formation performing obstacle avoidance shown in the simulations of Figure C-7 and Figure C-8 are similar to the no obstacle environment.

In this Chapter, an approach for adding formation control to the combined algorithm, developed in Chapter 4, is described. The algorithm is empirically shown to achieve the specified formation control objectives successfully. The triangle formation shape (Figure 5-6a) is shown to require more control effort by the follower vehicles to combine both formation shape maintenance and obstacle avoidance w.r.t. the snake-like shape (Figure 5-6b). For both formation shapes shown in Figure 5-6, it is crucial to have a sufficiently high scan rate of the LiDAR. A similar observation was made in section 4-4 for the hybrid adaptive feedback obstacle avoidance method. Because the dimensions of the leader ROSbot vehicle are relatively small, the vehicle is not always converted properly by the *obstacle_detector*-package into an object with correct relative position w.r.t. the follower vehicle's LiDAR.

# Chapter 6

# Conclusions & Further Research

In this chapter, conclusions based on the research objectives listed in Chapter 1 will be presented. Furthermore, recommendations will be given for further research topics based on the results found in this thesis research.

## 6-1 Conclusion

As elaborated on in Chapter 1, the deployment of autonomous vehicles to explore an unknown environment has been a growing research field. This thesis research focused on the practical implementation of autonomous vehicles to locate the source of a physical process with an unknown spatial distribution. A group of three autonomous vehicles should navigate towards the unknown source without crashing into an obstacle. Therefore, in Chapter 1, the following research objective was stated:

**Main Research Objective**

Develop and implement a control algorithm capable of steering a group of three autonomous unicycle vehicles towards an unknown source while avoiding an obstacle that interferes with its path and without inter-vehicle collisions. The combined control algorithm should be suitable to deploy on the ROSbot autonomous robot platform.

The above-stated research objective was divided into three sub-objectives in Chapter 1. Each of the three sub-objectives presented different research challenges: source seeking, obstacle avoidance, and formation control.

For the source seeking algorithm being that the control algorithm should able to be deployed on the ROSbot platform showed to be difficult. Due to the kinematic constraints caused by the ROSbot being a unicycle as described section 2-2. Therefore, research published by [7] and [3] was applied and tested on the ROSbot platform in simulation in Chapter 3. Where the dynamics of the ROSbot were found to be inhibiting the ROSbot from following the gradient

of the signal field accurately when a fixed sensor was used to probe the signal field. However, the oscillating sensor approach did not suffer from the ROSbot's dynamics, inhibiting the ROSbot from following the signal field gradient. Because the sensor movement is decoupled from the vehicle's movement. Convergence around the source was shown to be worse for the oscillating sensor source seeking ROSbot though compared to the fixed sensor approach. Because the derivative term used to slow down the forward speed was lower due to the high pass filtered signal containing more of the signal's steady-state component, as explained in section 3-4.

Adding the need for the source seeking ROSbot to avoid obstacles, as a consequence, introduced the problem of a detrimental line $M$, as further elaborated on in section 4-1-1. To prevent the ROSbot following line $M$, the authors in [19] presented a hybrid adaptive feedback (HAF) law solution that guarantees to be robust against the ROSbot following the detrimental line. However, the authors' solution in [19] suffers from some significant drawbacks. Namely, that the obstacle location and the obstacle orientation w.r.t. the source are assumed to be known *a priori* to the source seeking vehicle. These assumptions reduce such an obstacle avoidance algorithm's applicability to be deployed on an autonomous vehicle to explore unknown environments. Therefore, in Chapter 4 a HAF obstacle avoidance algorithm was developed that does not need the two before-mentioned assumptions. The obstacle's location w.r.t. the vehicle was determined based on the Light Detection And Ranging (LiDAR) sensor equipped on the ROSbot platform. To overcome the assumption that the orientation of the obstacle w.r.t. source is needed *a priori* to draw the partitioning lines of the hybrid sets around the obstacle. It was assumed that the ROSbot on average is orientated towards the source because it follows the gradient of the signal field estimated by the source seeking controller. Thereby, the relative angle w.r.t. the obstacle is used to determine how the partitioning lines should be drawn, as illustrated in Figure 4-8. To prevent sudden turns of the ROSbot needed to avoid the obstacle or the looping trajectory of the fixed sensor approach, causing the partitioning lines to rotate simultaneously, a low-pass filter is added. Thereby slowing down the partitioning lines' rotation, which determines the hybrid state w.r.t. the obstacle the ROSbot is in.

In addition to the developed algorithm in Chapter 4, the final part of this thesis research was to enable a group of three autonomous vehicles to drive in a predefined formation shape towards the source whilst being able to avoid an obstacle. In Chapter 5, a solution is presented to realize such a control algorithm, where a decentralized leader-follower architecture was chosen based on the research by [30]. Only the follower vehicles in this control architecture are utilized to achieve the formation control objectives. The control objective of the follower vehicles is to try to maintain a desired angle and distance w.r.t. the leader vehicle (see Eq. (5-1.1)), which are measured based on the LiDAR data obtained by the ROSbot. Two different formation shapes are used to test the formation control algorithm, a snake-like and triangle-shaped formation as illustrated in Figure 5-6. To steer the follower vehicles away from an obstacle, a similar control law as defined in Eq. (5-2.8) is used for leader tracking (see Eq. (5-2.3)). Since a decentralized leader-follower control architecture is used, and the follower vehicles can only determine where the leader vehicle is based on the relative position of the nearby ROSbots, leader tracking could fail. Because if too aggressive steering adjustment is applied, it can lead the follower vehicle to follow the other follower vehicle of the formation misleadingly. The other follower vehicle is closer to the desired bearing $\psi_d$ than the leader vehicle due to too much steering adjustment. Therefore, in section 5-5 an elaboration is given under which

parameter configurations the formation algorithm would fail and under which parameters successful simulations were performed.

The final combined algorithm developed in this thesis satisfies the main research objective. The algorithm is empirically shown not the suffer from the detrimental line $M$, which would occur if an artificial potential function (APF) approach would be used. On top of that, a practical implementation study is performed to deploy the combined algorithm on the Husarion ROSbot platform for experiments. In Figure 6-1, a simulation is shown of a formation of three ROSbot vehicles with one leader vehicle conducting source seeking equipped with an oscillating sensor. The other two follower vehicles follow the leader vehicle while maintaining a triangle formation shape (see Figure 5-6a) and avoid the obstacle. The developed algorithm is developed to be suitable to deploy on the ROSbot platform and simulated under the conditions further described in sections 3-4, 4-4 and 5-5. These conditions are used in all simulations to reflect closely how the ROSbot would experience the environment in a physical experiment.

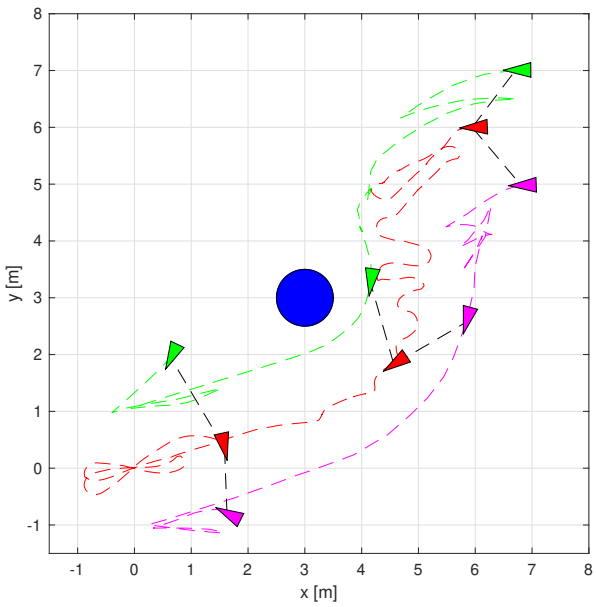



**Figure 6-1:** The simulated trajectory of a formation with three ROSbots, where the source is located at $x : 0[m]$, $y : 0[m]$

## 6-2 Future Research

This thesis research's objective was to conduct a practical implementation study on an algorithm following the main research objective. However, there remains a lot of open research questions and challenges to be studied. Here, some future research recommendations will be discussed.

### Physical experiments

Unfortunately, because of restrictions imposed due to the global pandemic, no physical experiments could be performed. Conducting physical experiments using the developed combined algorithm on three Husarion ROSbots could validate the conclusions drawn in this Chapter. The laboratory experiments could be designed to generate the sensor and signal emitting source virtually by using a motion capture system to track the location of the ROSbots. If the experiments' results are successful, a physical sensor can be constructed based either on the more simple fixed sensor or on the oscillating sensor design. Subsequently, the combined algorithm could be tested to seek the source of an actual physical process.

### Gradient field research

One important assumption made for the HAF obstacle avoidance law was that the obstacle does not influence the source's signal field. However, some signal emitting physical processes can produce signals that can be altered by obstacles in an environment. E.g., a toxic gas plume could be such a source where the flow can be affected by an obstacle and thus the signal field. Further research into what types of signal fields the HAF method is applicable would be interesting, as it would increase the method's applicability.

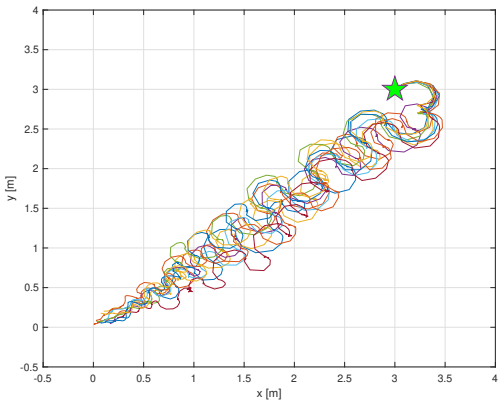
### Distributed source seeking

In this thesis report, a single-vehicle is used to estimate the gradient of the signal field. Employing multiple vehicles to estimate the gradient could improve the source-seeking algorithm's reliability because if one vehicle fails, the other vehicles in the formation could continue navigating towards the source. The authors in [2] present a promising approach suitable for multiple unicycles estimate the signal field's gradient, based on local strength measurements. A disadvantage of using multiple vehicles to estimate the gradient is that a communication network between vehicles needs to be established, which is not needed in the algorithm developed in this thesis research. Further research, however, in how a distributed source seeking approach for unicycles could be employed to explore an unknown cluttered environment could valuable to assess the practical limitations and performance compared to the algorithm developed in this thesis.

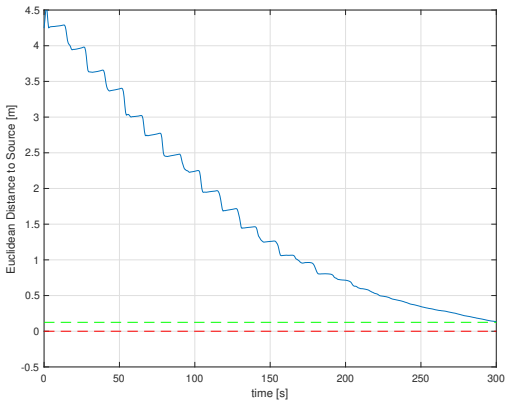
# Appendix A

# Source Seeking Simulations

## A-1 Fixed Sensor




**(a)** All trajectories




**(b)** Distance to source

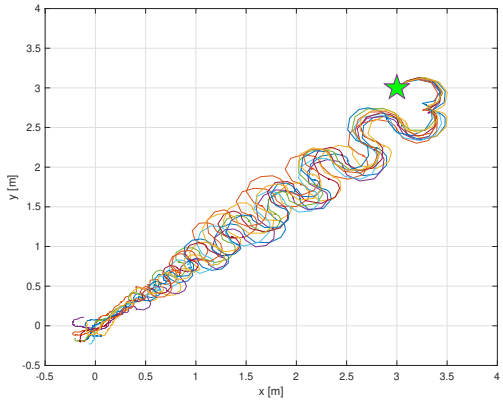
| Parameter | Value | Unit (SI) |
|---|---|---|
| $\omega$ | 0.25 | $rad$ |
| $h$ | 1 | unitless |
| $c$ | 80 | unitless |
| $\alpha$ | 1.8 | unitless |
| $R$ | 0.1 | $m$ |
| $b$ | 0.5 | unitless |
| $V_c$ | 0.005 | $m/s$ |

**(c)** Tuning parameters

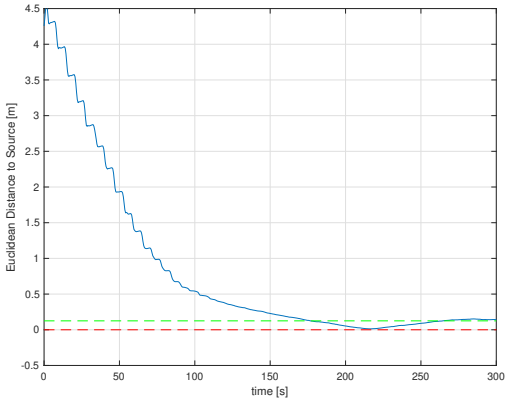
| Characteristic | Value | Unit (SI) |
|---|---|---|
| Time to source | 300 | $s$ |
| Path length | 4.13 | $m$ |
| Path consistency | 0.201 | unitless |
| Convergence | 0.152 | $m$ |

**(d)** Performance characteristics

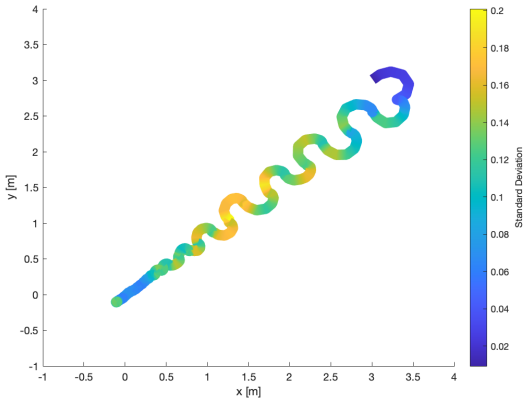
**Figure A-1:** Simulation of 10 trajectories of the source seeking controller using a fixed sensor approach, where the ROSbot's geometric center is analyzed. The source is located at $x : 0[m]$, $y : 0[m]$.




**(a)** All trajectories




**(b)** Distance to source




**(c)** Mean trajectory with standard deviation

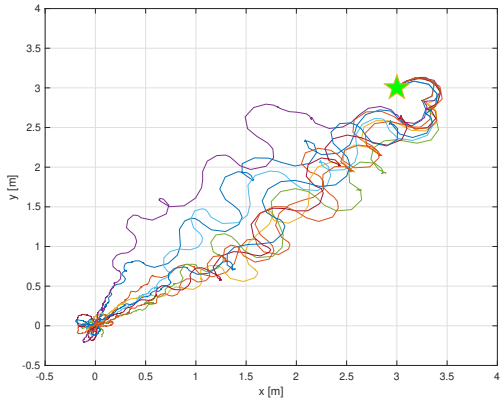
| Parameter | Value | Unit (SI) |
|---|---|---|
| $\omega$ | 0.5 | $rad$ |
| $h$ | 1 | unitless |
| $c$ | 80 | unitless |
| $\alpha$ | 1.8 | unitless |
| $R$ | 0.1 | $m$ |
| $b$ | 0.5 | unitless |
| $V_c$ | 0.005 | $m/s$ |

**(d)** Tuning parameters

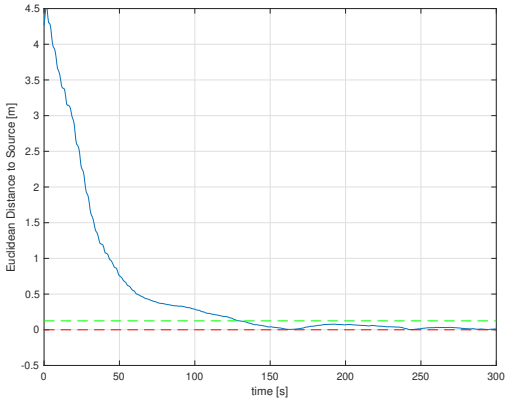
| Characteristic | Value | Unit (SI) |
|---|---|---|
| Time to source | 176.8 | $s$ |
| Path length | 4.13 | $m$ |
| Path consistency | 0.201 | unitless |
| Convergence | 0.152 | $m$ |

**(e)** Performance characteristics

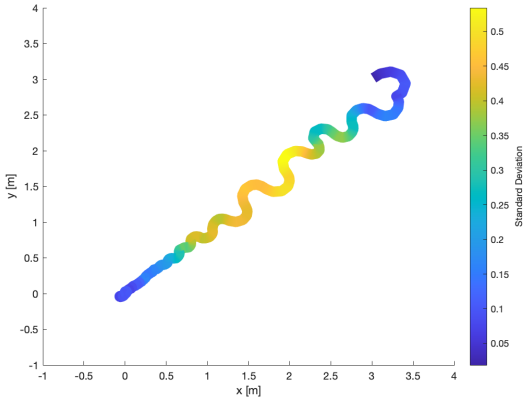
**Figure A-2:** Simulation of 10 trajectories of the source seeking controller using a fixed sensor approach, where the ROSbot's geometric center is analyzed. The source is located at $x : 0[m]$, $y : 0[m]$.




**(a)** All trajectories




**(b)** Distance to source




**(c)** Mean trajectory with standard deviation

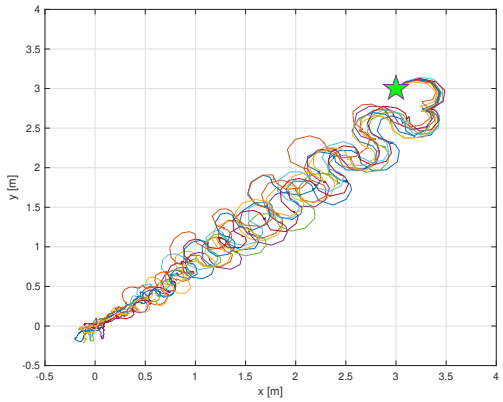
| Parameter | Value | Unit (SI) |
|---|---|---|
| $\omega$ | 1 | $rad$ |
| $h$ | 1 | unitless |
| $c$ | 80 | unitless |
| $\alpha$ | 1.8 | unitless |
| $R$ | 0.1 | $m$ |
| $b$ | 0.5 | unitless |
| $V_c$ | 0.005 | $m/s$ |

**(d)** Tuning parameters

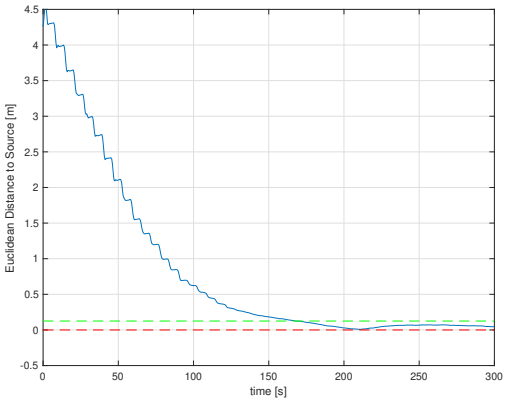
| Characteristic | Value | Unit (SI) |
|---|---|---|
| Time to source | 129.7 | $s$ |
| Path length | 4.14 | $m$ |
| Path consistency | 0.534 | unitless |
| Convergence | 0.124 | $m$ |

**(e)** Performance characteristics

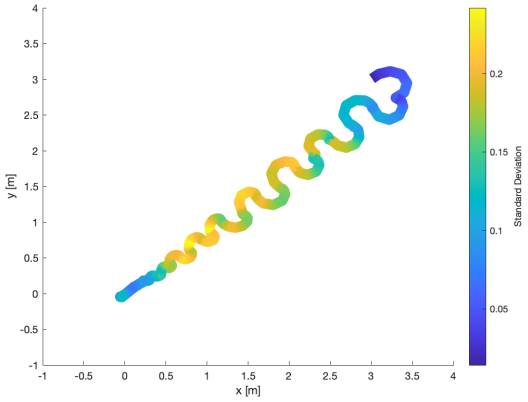
**Figure A-3:** Simulation of 10 trajectories of the source seeking controller using a fixed sensor approach, where the ROSbot's geometric center is analyzed. The source is located at $x : 0[m]$, $y : 0[m]$.




**(a)** All trajectories




**(b)** Distance to source




**(c)** Mean trajectory with standard deviation

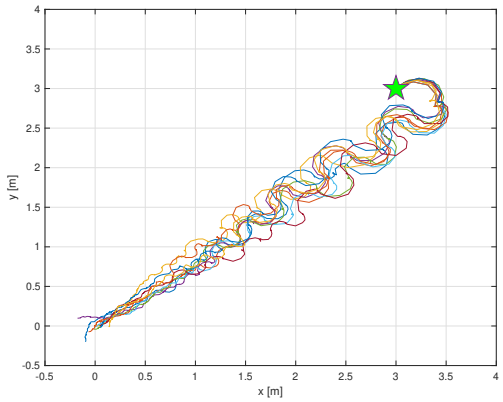
| Parameter | Value | Unit (SI) |
|---|---|---|
| $\omega$ | 0.5 | $rad$ |
| $h$ | 1 | unitless |
| $c$ | 160 | unitless |
| $\alpha$ | 1.8 | unitless |
| $R$ | 0.1 | $m$ |
| $b$ | 0.5 | unitless |
| $V_c$ | 0.005 | $m/s$ |

**(d)** Tuning parameters

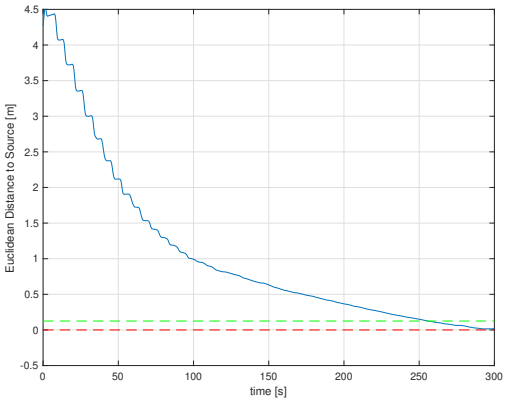
| Characteristic | Value | Unit (SI) |
|---|---|---|
| Time to source | 168.1 | $s$ |
| Path length | 4.13 | $m$ |
| Path consistency | 0.241 | unitless |
| Convergence | 0.125 | $m$ |

**(e)** Performance characteristics

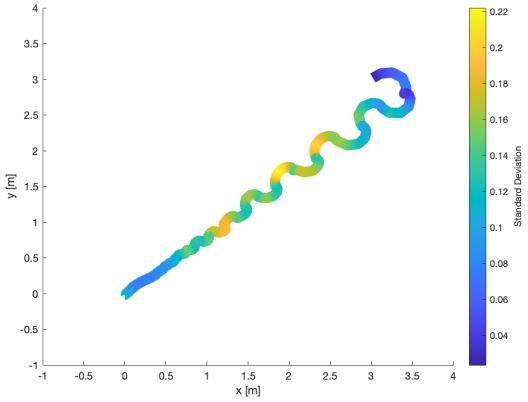
**Figure A-4:** Simulation of 10 trajectories of the source seeking controller using a fixed sensor approach, where the ROSbot's geometric center is analyzed. The source is located at $x : 0[m]$, $y : 0[m]$.




**(a)** All trajectories




**(b)** Distance to source




**(c)** Mean trajectory with standard deviation

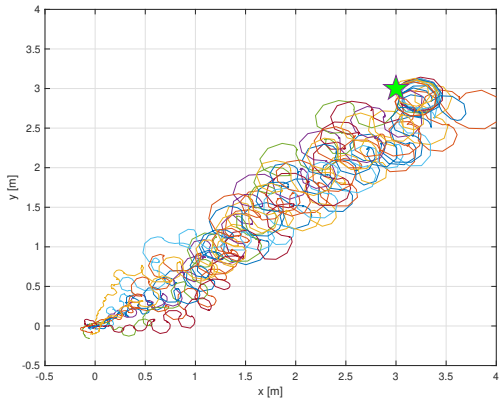
| Parameter | Value | Unit (SI) |
|---|---|---|
| $\omega$ | 0.5 | $rad$ |
| $h$ | 2 | unitless |
| $c$ | 80 | unitless |
| $\alpha$ | 1.8 | unitless |
| $R$ | 0.1 | $m$ |
| $b$ | 0.5 | unitless |
| $V_c$ | 0.005 | $m/s$ |

**(d)** Tuning parameters

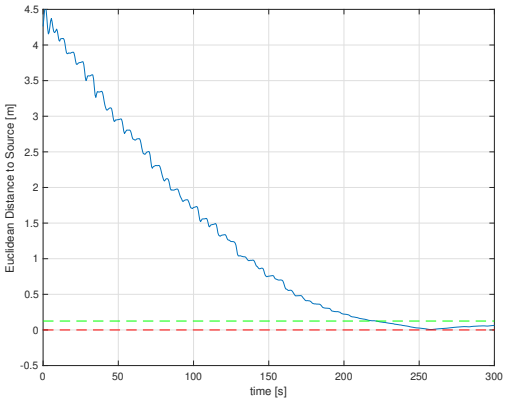
| Characteristic | Value | Unit (SI) |
|---|---|---|
| Time to source | 256.5 | $s$ |
| Path length | 4.14 | $m$ |
| Path consistency | 0.222 | unitless |
| Convergence | 0.124 | $m$ |

**(e)** Performance characteristics

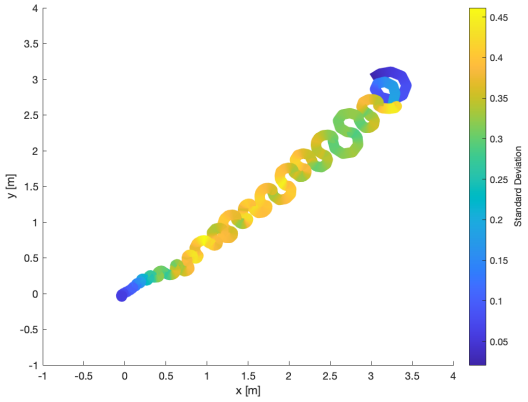
**Figure A-5:** Simulation of 10 trajectories of the source seeking controller using a fixed sensor approach, where the ROSbot's geometric center is analyzed. The source is located at $x : 0[m]$, $y : 0[m]$.




**(a)** All trajectories




**(b)** Distance to source




**(c)** Mean trajectory with standard deviation

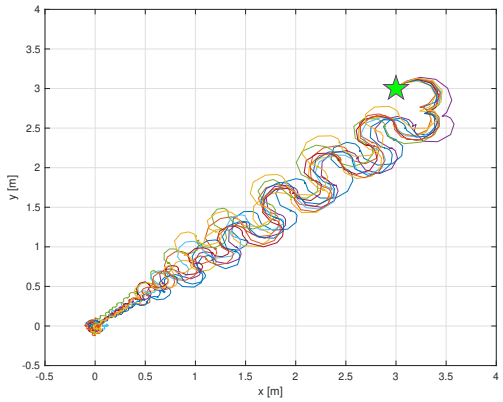
| Parameter | Value | Unit (SI) |
|---|---|---|
| $\omega$ | 0.5 | $rad$ |
| $h$ | 0.5 | unitless |
| $c$ | 80 | unitless |
| $\alpha$ | 1.8 | unitless |
| $R$ | 0.1 | $m$ |
| $b$ | 0.5 | unitless |
| $V_c$ | 0.005 | $m/s$ |

**(d)** Tuning parameters

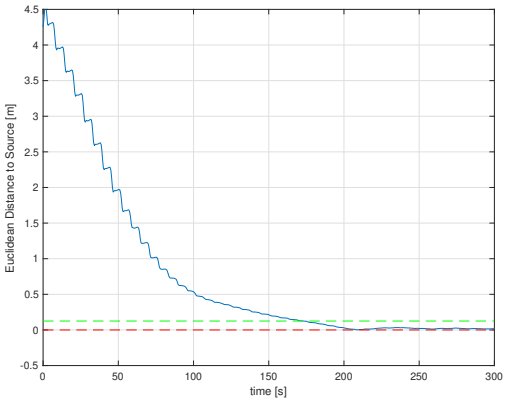
| Characteristic | Value | Unit (SI) |
|---|---|---|
| Time to source | 221.4 | $s$ |
| Path length | 4.14 | $m$ |
| Path consistency | 0.461 | unitless |
| Convergence | 0.124 | $m$ |

**(e)** Performance characteristics

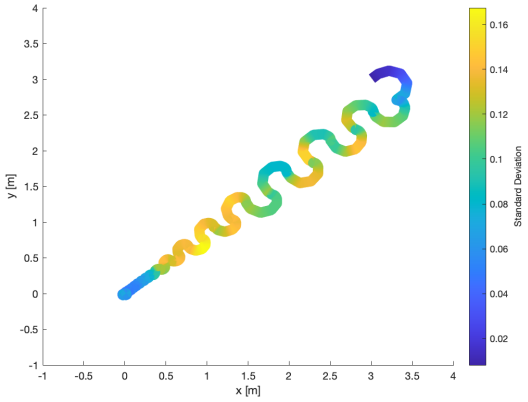
**Figure A-6:** Simulation of 10 trajectories of the source seeking controller using a fixed sensor approach, where the ROSbot's geometric center is analyzed. The source is located at $x : 0[m]$, $y : 0[m]$.




**(a)** All trajectories




**(b)** Distance to source




**(c)** Mean trajectory with standard deviation

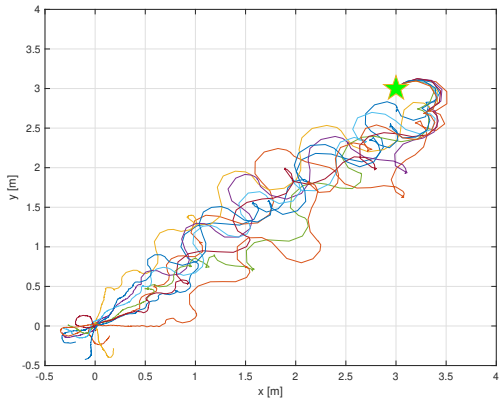
| Parameter | Value | Unit (SI) |
|---|---|---|
| $\omega$ | 0.5 | $rad$ |
| $h$ | 1 | unitless |
| $c$ | 80 | unitless |
| $\alpha$ | 3.6 | unitless |
| $R$ | 0.1 | $m$ |
| $b$ | 0.5 | unitless |
| $V_c$ | 0.005 | $m/s$ |

**(d)** Tuning parameters

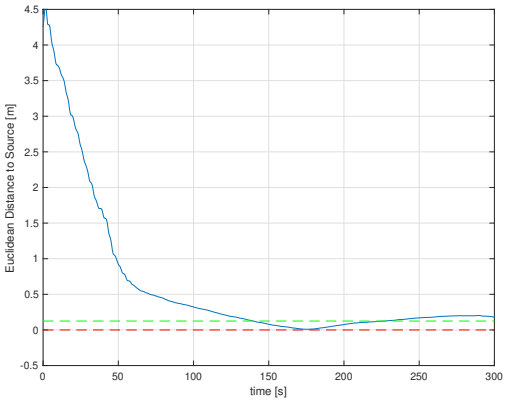
| Characteristic | Value | Unit (SI) |
|---|---|---|
| Time to source | 173.3 | $s$ |
| Path length | 4.13 | $m$ |
| Path consistency | 0.167 | unitless |
| Convergence | 0.124 | $m$ |

**(e)** Performance characteristics

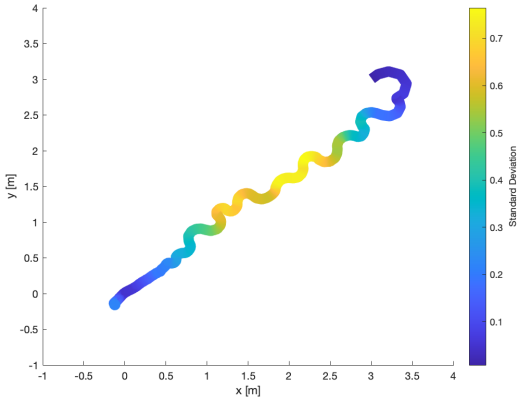
**Figure A-7:** Simulation of 10 trajectories of the source seeking controller using a fixed sensor approach, where the ROSbot's geometric center is analyzed. The source is located at $x : 0[m]$, $y : 0[m]$.




**(a)** All trajectories




**(b)** Distance to source




**(c)** Mean trajectory with standard deviation

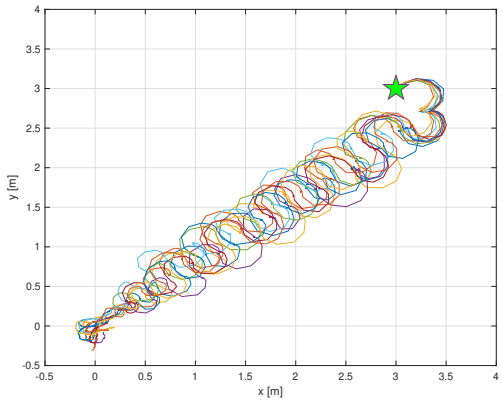
| Parameter | Value | Unit (SI) |
|---|---|---|
| $\omega$ | 0.5 | $rad$ |
| $h$ | 1 | unitless |
| $c$ | 80 | unitless |
| $\alpha$ | 0.9 | unitless |
| $R$ | 0.1 | $m$ |
| $b$ | 0.5 | unitless |
| $V_c$ | 0.005 | $m/s$ |

**(d)** Tuning parameters

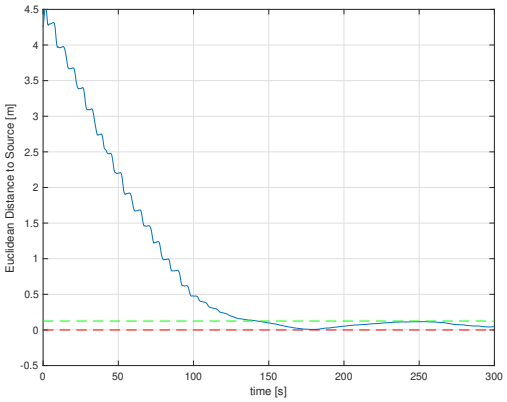
| Characteristic | Value | Unit (SI) |
|---|---|---|
| Time to source | 140.2 | $s$ |
| Path length | 4.13 | $m$ |
| Path consistency | 0.764 | unitless |
| Convergence | 0.204 | $m$ |

**(e)** Performance characteristics

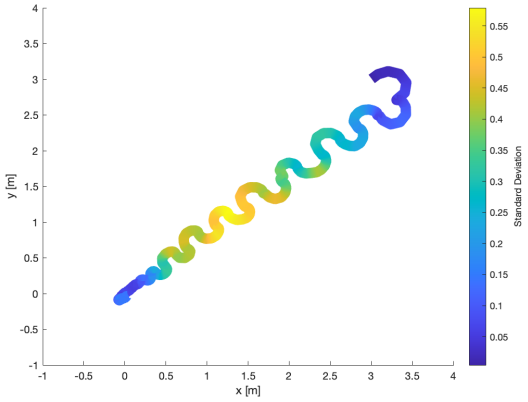
**Figure A-8:** Simulation of 10 trajectories of the source seeking controller using a fixed sensor approach, where the ROSbot's geometric center is analyzed. The source is located at $x : 0[m]$, $y : 0[m]$.




**(a)** All trajectories




**(b)** Distance to source




**(c)** Mean trajectory with standard deviation

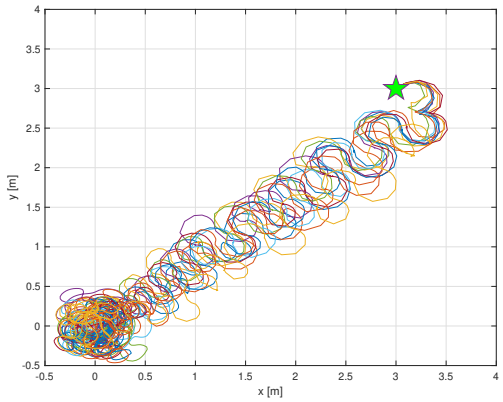
| Parameter | Value | Unit (SI) |
|---|---|---|
| $\omega$ | 0.5 | $rad$ |
| $h$ | 1 | unitless |
| $c$ | 80 | unitless |
| $\alpha$ | 1.8 | unitless |
| $R$ | 0.1 | $m$ |
| $b$ | 1 | unitless |
| $V_c$ | 0.005 | $m/s$ |

**(d)** Tuning parameters

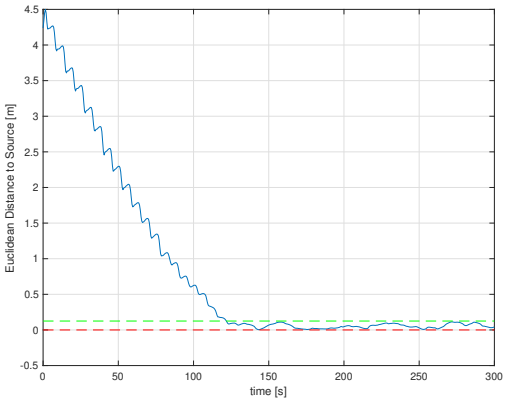
| Characteristic | Value | Unit (SI) |
|---|---|---|
| Time to source | 142.8 | $s$ |
| Path length | 4.13 | $m$ |
| Path consistency | 0.579 | unitless |
| Convergence | 0.124 | $m$ |

**(e)** Performance characteristics

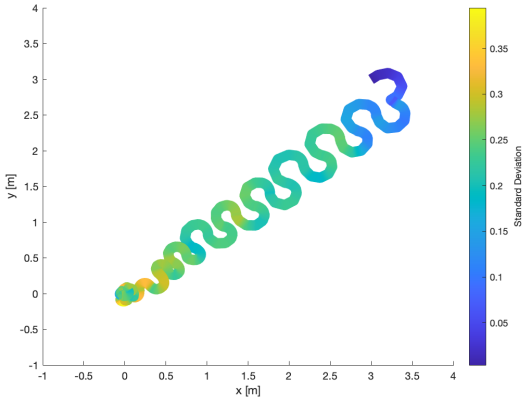
**Figure A-9:** Simulation of 10 trajectories of the source seeking controller using a fixed sensor approach, where the ROSbot's geometric center is analyzed. The source is located at $x : 0[m]$, $y : 0[m]$.




**(a)** All trajectories




**(b)** Distance to source




**(c)** Mean trajectory with standard deviation

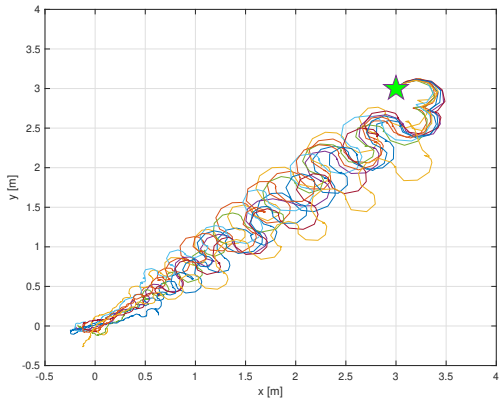
| Parameter | Value | Unit (SI) |
|---|---|---|
| $\omega$ | 0.5 | $rad$ |
| $h$ | 1 | unitless |
| $c$ | 80 | unitless |
| $\alpha$ | 1.8 | unitless |
| $R$ | 0.1 | $m$ |
| $b$ | 0.5 | unitless |
| $V_c$ | 0.05 | $m/s$ |

**(d)** Tuning parameters

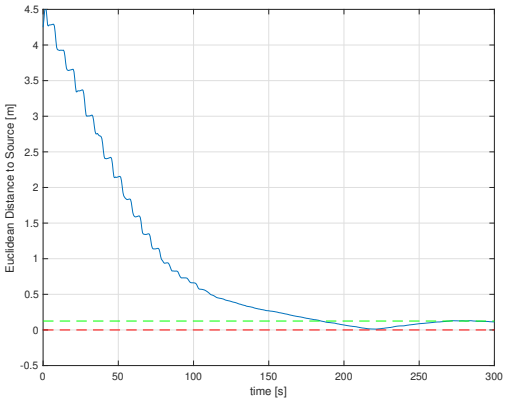
| Characteristic | Value | Unit (SI) |
|---|---|---|
| Time to source | 122.2 | $s$ |
| Path length | 4.13 | $m$ |
| Path consistency | 0.332 | unitless |
| Convergence | 0.113 | $m$ |

**(e)** Performance characteristics

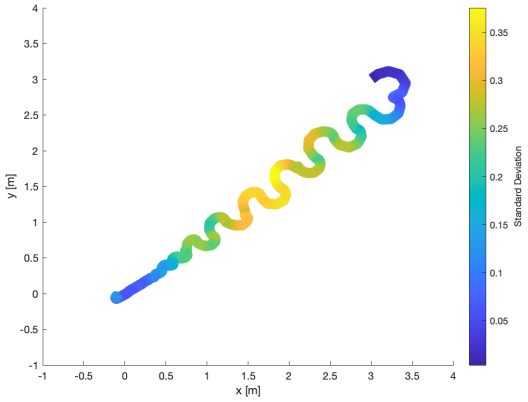
**Figure A-10:** Simulation of 10 trajectories of the source seeking controller using a fixed sensor approach, where the ROSbot's geometric center is analyzed. The source is located at $x : 0[m]$, $y : 0[m]$.




**(a)** All trajectories




**(b)** Distance to source




**(c)** Mean trajectory with standard deviation

| Parameter | Value | Unit (SI) |
|---|---|---|
| $\omega$ | 0.5 | $rad$ |
| $h$ | 1 | unitless |
| $c$ | 80 | unitless |
| $\alpha$ | 1.8 | unitless |
| $R$ | 0.01 | $m$ |
| $b$ | 0.5 | unitless |
| $V_c$ | 0.005 | $m/s$ |

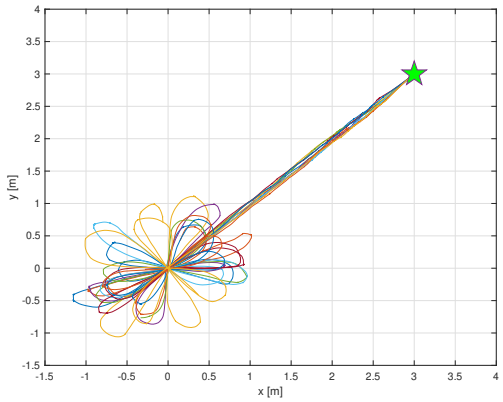
**(d)** Tuning parameters

| Characteristic | Value | Unit (SI) |
|---|---|---|
| Time to source | 184.8 | $s$ |
| Path length | 4.13 | $m$ |
| Path consistency | 0.376 | unitless |
| Convergence | 0.130 | $m$ |

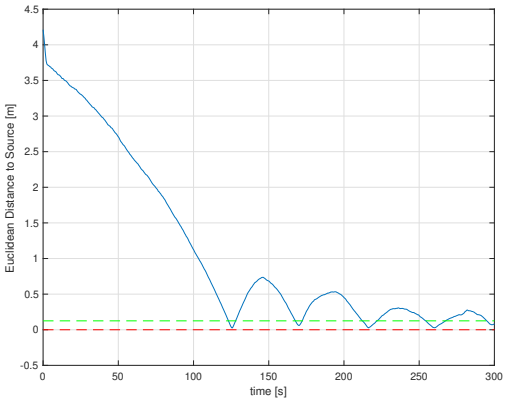
**(e)** Performance characteristics

**Figure A-11:** Simulation of 10 trajectories of the source seeking controller using a fixed sensor approach, where the ROSbot's geometric center is analyzed. The source is located at $x : 0[m]$, $y : 0[m]$.

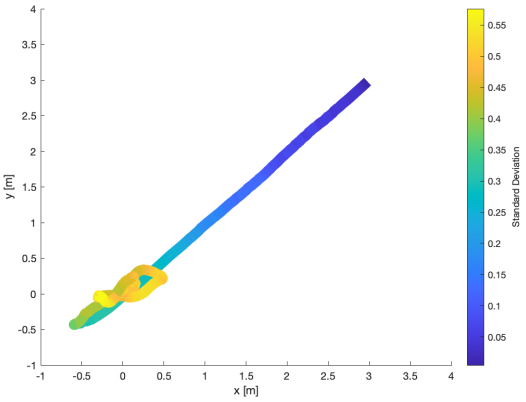
# A-2 Oscillating Sensor




**(a)** All trajectories




**(b)** Distance to source




**(c)** Mean trajectory with standard deviation

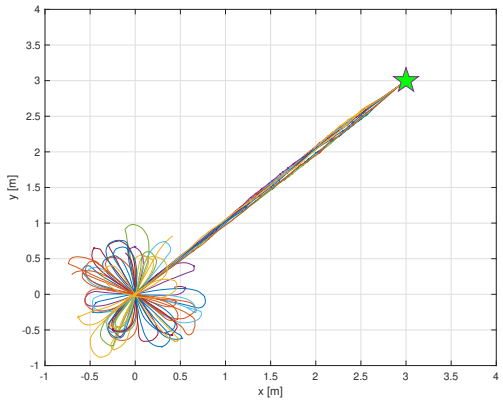
| Parameter | Value | Unit (SI) |
|---|---|---|
| $\omega$ | 30 | $rad/s$ |
| $h$ | 1 | unitless |
| $c$ | 80 | unitless |
| $\alpha$ | 1.8 | unitless |
| $R$ | 0.1 | $m$ |
| $b$ | -0.25 | unitless |
| $V_c$ | 0.05 | $m/s$ |

**(d)** Tuning parameters

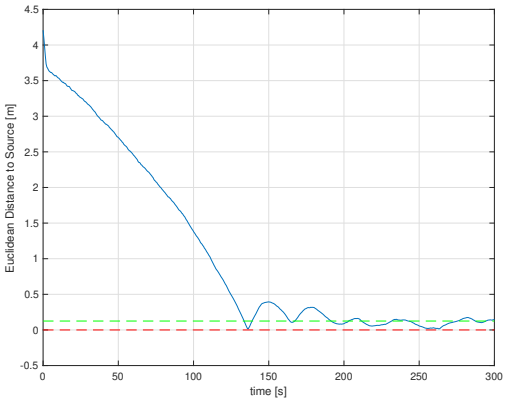
| Characteristic | Value | Unit (SI) |
|---|---|---|
| Time to source | 123.2 | $s$ |
| Path length | 4.09 | $m$ |
| Path consistency | 0.297 | unitless |
| Convergence | 0.784 | $m$ |

**(e)** Performance characteristics

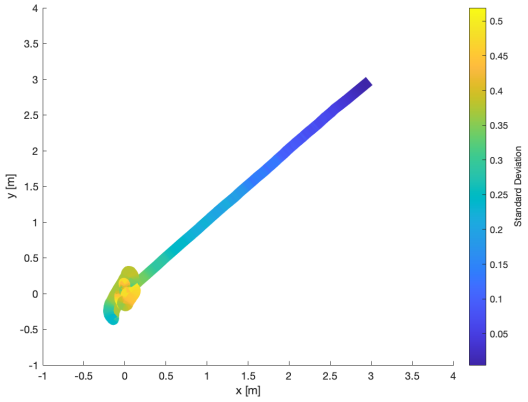
**Figure A-12:** Simulation of 10 trajectories of the source seeking controller using an oscillating sensor approach, where the ROSbot's geometric center is analyzed. The source is located at $x : 0[m]$, $y : 0[m]$.




**(a)** All trajectories




**(b)** Distance to source




**(c)** Mean trajectory with standard deviation

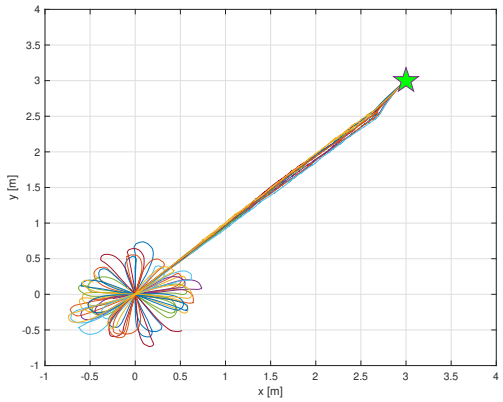
| Parameter | Value | Unit (SI) |
|---|---|---|
| $\omega$ | 20 | $rad/s$ |
| $h$ | 1 | unitless |
| $c$ | 80 | unitless |
| $\alpha$ | 1.8 | unitless |
| $R$ | 0.1 | $m$ |
| $b$ | -0.25 | unitless |
| $V_c$ | 0.05 | $m/s$ |

**(d)** Tuning parameters

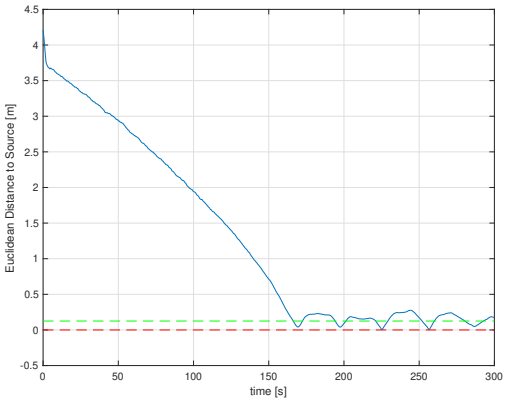
| Characteristic | Value | Unit (SI) |
|---|---|---|
| Time to source | 133.7 | $s$ |
| Path length | 4.09 | $m$ |
| Path consistency | 0.365 | unitless |
| Convergence | 0.394 | $m$ |

**(e)** Performance characteristics

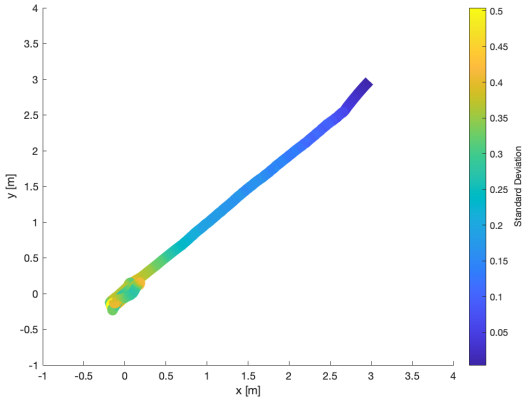
**Figure A-13:** Simulation of 10 trajectories of the source seeking controller using an oscillating sensor approach, where the ROSbot's geometric center is analyzed. The source is located at $x : 0[m]$, $y : 0[m]$.




**(a)** All trajectories




**(b)** Distance to source




**(c)** Mean trajectory with standard deviation

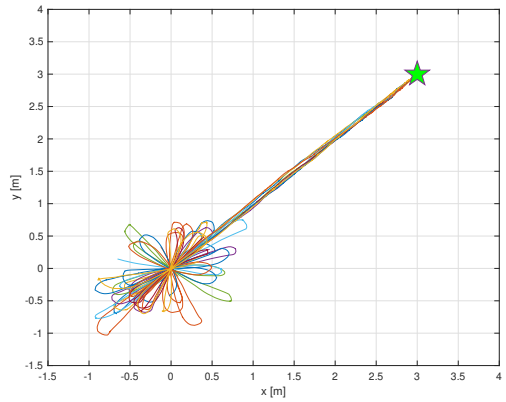
| Parameter | Value | Unit (SI) |
|---|---|---|
| $\omega$ | 10 | $rad/s$ |
| $h$ | 1 | unitless |
| $c$ | 80 | unitless |
| $\alpha$ | 1.8 | unitless |
| $R$ | 0.1 | $m$ |
| $b$ | -0.25 | unitless |
| $V_c$ | 0.05 | $m/s$ |

**(d)** Tuning parameters

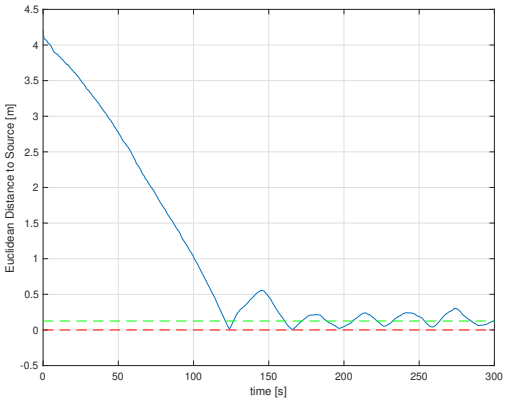
| Characteristic | Value | Unit (SI) |
|---|---|---|
| Time to source | 166.7 | $s$ |
| Path length | 4.10 | $m$ |
| Path consistency | 0.391 | unitless |
| Convergence | 0.269 | $m$ |

**(e)** Performance characteristics

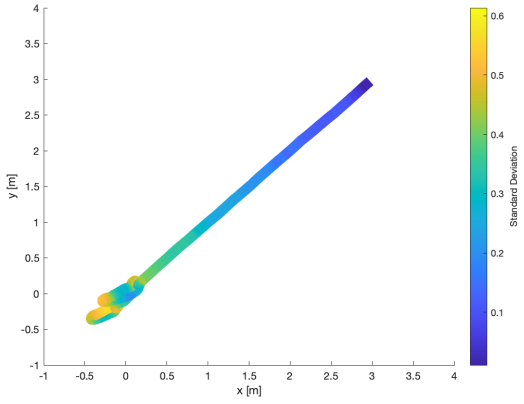
**Figure A-14:** Simulation of 10 trajectories of the source seeking controller using an oscillating sensor approach, where the ROSbot's geometric center is analyzed. The source is located at $x : 0[m]$, $y : 0[m]$.




**(a)** All trajectories




**(b)** Distance to source




**(c)** Mean trajectory with standard deviation

| Parameter | Value | Unit (SI) |
|---|---|---|
| $\omega$ | 20 | $rad/s$ |
| $h$ | 5 | unitless |
| $c$ | 80 | unitless |
| $\alpha$ | 1.8 | unitless |
| $R$ | 0.1 | $m$ |
| $b$ | -0.25 | unitless |
| $V_c$ | 0.05 | $m/s$ |

**(d)** Tuning parameters

| Characteristic | Value | Unit (SI) |
|---|---|---|
| Time to source | 121.2 | $s$ |
| Path length | 4.08 | $m$ |
| Path consistency | 0.432 | $unitless$ |
| Convergence | 0.554 | $m$ |

**(e)** Performance characteristics

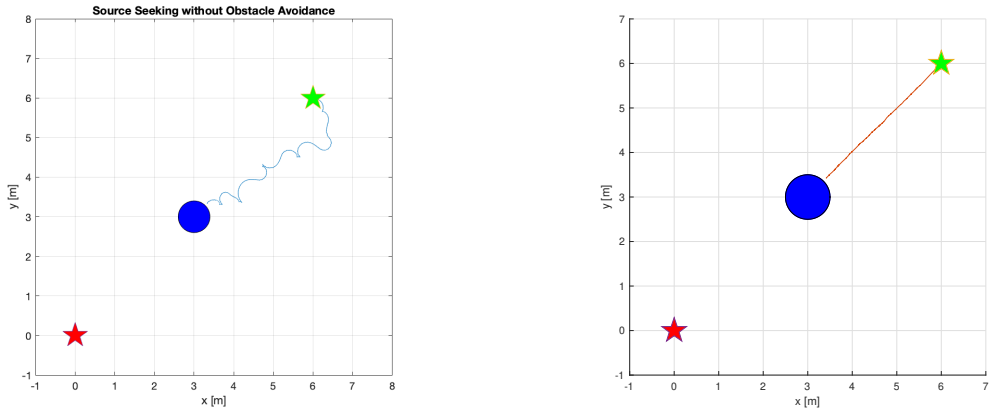
**Figure A-15:** Simulation of 10 trajectories of the source seeking controller using an oscillating sensor approach, where the ROSbot's geometric center is analyzed. The source is located at $x : 0[m]$, $y : 0[m]$.

# Appendix B

# Obstacle Avoidance Simulations

In this appendix Chapter, the simulated trajectories are shown for all the evualated obstacle avoidance methods in Chapter 4. In all simulations shown in this Chapter, the source is located at $x : 0[m]$, $y : 0[m]$(red star), the obstacle at $x : 3[m]$, $y : 3[m]$(blue) and the initial position at $x : 6[m]$, $y : 6[m]$(green star).

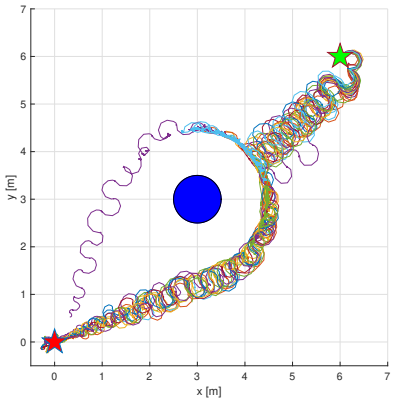
## B-1 Simulations without Obstacle Avoidance




**(a)** Fixed Sensor **(b)** Oscillating Sensor

**Figure B-1:** A figure showing how the trajectory of the ROSbot crashing into the obstacle if no obstacle avoidance method would be incorporated with the source seeking controller. The initial position of the ROSbot is located at $x : 6[m]$, $y : 6[m]$(green star) and source location at $x : 0[m]$, $y : 0[m]$(red star), with obstacle located at $x : 3[m]$, $y : 3[m]$(blue).

## B-2 Artificial Potential Function




**(a)** 20 trajectories of the ROSbot's geometric center

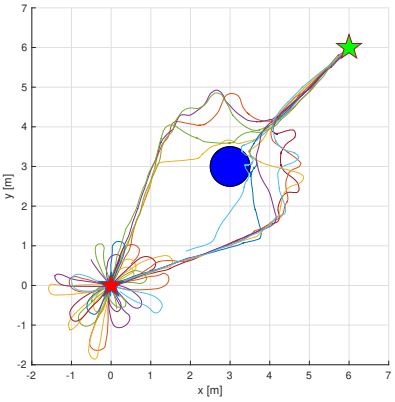
| Parameter | Value | Unit (SI) |
|---|---|---|
| $\omega$ | 0.5 | $rad$ |
| $h$ | 2 | unitless |
| $c$ | 80 | unitless |
| $\alpha$ | 1.8 | unitless |
| $R$ | 0.1 | $m$ |
| $b$ | 0.5 | unitless |
| $V_c$ | 0.005 | $m/s$ |

**(b)** Source seeking parameters

| Parameter | Value | Unit (SI) |
|---|---|---|
| $\rho_{apf}$ | 0.75 | $m$ |
| $h_{apf}$ | 0.5 | $m$ |

**(c)** Barrier function parameters

**Figure B-2:** Simulation showing 20 trajectories, where the artificial potential function obstacle avoidance method employed on a fixed sensor source seeking ROSbot does not suffer from line $M$.




**(a)** 10 trajectories of the ROSbot's geometric center

| Parameter | Value | Unit (SI) |
|---|---|---|
| $\omega$ | 20 | $rad$ |
| $h$ | 1 | unitless |
| $c$ | 80 | unitless |
| $\alpha$ | 1.8 | unitless |
| $R$ | 0.1 | $m$ |
| $b$ | 0 | unitless |
| $V_c$ | 0.1 | $m/s$ |

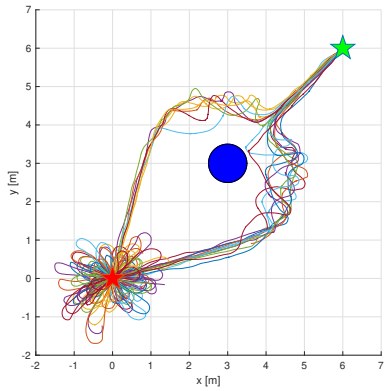
**(b)** Source seeking parameters

| Parameter | Value | Unit (SI) |
|---|---|---|
| $\rho_{apf}$ | 0.75 | $m$ |
| $h_{apf}$ | 0.5 | $m$ |

**(c)** Barrier function parameters

**Figure B-3:** Simulation showing 10 trajectories, where the artificial potential function obstacle avoidance method is not capable of steering the ROSbot away from line $M$ 4 out 10 times.

## B-3 Virtual Box




**(a)** All trajectories of the ROSbot's geometric center

| Parameter | Value | Unit (SI) |
|---|---|---|
| $\kappa$ | 3 | $m$ |
| $\mu$ | $1 * 10^4$ | *unitless* |
| $N$ | 2500 | *unitless* |

**(b)** Hybrid adaptive parameters

| Parameter | Value | Unit (SI) |
|---|---|---|
| $\rho$ | 0.75 | $m$ |
| $h$ | 0.5 | $m$ |

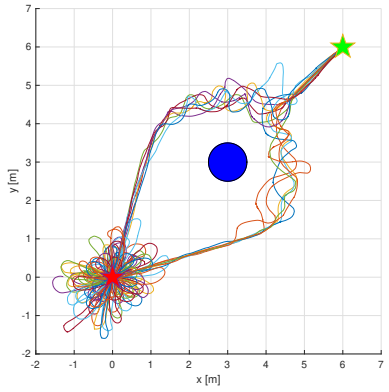
**(c)** Barrier function parameters

| Parameter | Value | Unit (SI) |
|---|---|---|
| $\omega$ | 20 | $rad$ |
| $h$ | 1 | unitless |
| $c$ | 80 | unitless |
| $\alpha$ | 1.8 | unitless |
| $R$ | 0.1 | $m$ |
| $b$ | 0 | unitless |
| $V_c$ | 0.1 | $m/s$ |

**(d)** Source seeking parameters

**Figure B-4:** Simulation showing $20$ trajectory, where using the a diamond box artificial potential approach is not capable of steering the ROSbot away from line $M$. $2$ times the ROSbot enters the safety zone.

## B-4 Hybrid Adaptive




**(a)** All trajectories of the ROSbot's geometric center

| Parameter | Value | Unit (SI) |
|---|---|---|
| $\kappa$ | 3 | $m$ |
| $\mu$ | $1 * 10^4$ | *unitless* |
| $N$ | 2500 | *unitless* |

**(b)** Hybrid adaptive parameters

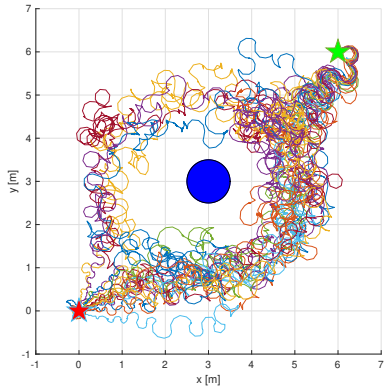
| Parameter | Value | Unit (SI) |
|---|---|---|
| $\rho$ | 0.75 | $m$ |
| $h$ | 0.5 | $m$ |

**(c)** Barrier function parameters

| Parameter | Value | Unit (SI) |
|---|---|---|
| $\omega$ | 20 | $rad$ |
| $h$ | 1 | unitless |
| $c$ | 80 | unitless |
| $\alpha$ | 1.8 | unitless |
| $R$ | 0.1 | $m$ |
| $b$ | 0 | unitless |
| $V_c$ | 0.1 | $m/s$ |

**(d)** Source seeking parameters

**Figure B-5:** Simulation showing one trajectory using an oscillating sensor source seeking approach, where using the hybrid adaptive approach is capable of steering the ROSbot away from line $M$. With switch state graph and table with hybrid parameters




**(a)** All trajectories of the ROSbot's geometric center

| Parameter | Value | Unit (SI) |
|---|---|---|
| $\kappa$ | 3 | $m$ |
| $\mu$ | $1 * 10^4$ | *unitless* |
| $N$ | 2500 | *unitless* |

**(b)** Hybrid adaptive parameters

| Parameter | Value | Unit (SI) |
|---|---|---|
| $\rho$ | 0.75 | $m$ |
| $h$ | 0.5 | $m$ |

**(c)** Barrier function parameters

| Parameter | Value | Unit (SI) |
|---|---|---|
| $\omega$ | 0.5 | $rad$ |
| $h$ | 2 | unitless |
| $c$ | 80 | unitless |
| $\alpha$ | 1.8 | unitless |
| $R$ | 0.1 | $m$ |
| $b$ | 0.5 | unitless |
| $V_c$ | 0.005 | $m/s$ |

**(d)** Source seeking parameters

**Figure B-6:** Simulation showing one trajectory using an fixed sensor source seeking approach, where using the hybrid adaptive approach is capable of steering the ROSbot away from line $M$. With switch state graph and table with hybrid parameters

# Appendix C

# Formation Control Simulations

## C-1 Conditions for Leader Tracking

$$\xi_z = \min\{\cos\psi_{\min}, \cos\psi_{\max}\}, \quad G_z = \max\{\ \frac{2\beta_z}{z_{\max} - z_d}, \frac{2\beta_z}{z_d - z_{\min}}\} \tag{C-1.1a}$$

$$\xi_\psi = \min\{\sin\psi_{\min}, \sin\psi_{\max}\}, \quad G_\psi = \max\{\ \frac{2\beta_\psi}{\psi_{\max} - \psi_d}, \frac{2\beta_\psi}{\psi_d - \psi_{\min}}\} \tag{C-1.1b}$$

$$k_z \leq \frac{v_F\xi_z - v_L}{G_z} \tag{C-1.2a}$$

$$k_\psi \leq \frac{\omega_F z_{\min} - v_L - v_F\xi_\psi}{G_\psi z_{\min}} \tag{C-1.2b}$$

$$z_{\min} < z_d - \frac{(z_{\min} - z_d)^2}{2\beta_z}\sqrt{\frac{k_z\delta_z}{k_z}} \tag{C-1.2c}$$

$$z_{\max} > z_d + \frac{(z_{\max} - z_d)^2}{2\beta_z}\sqrt{\frac{k_z\delta_z}{k_z}} \tag{C-1.2d}$$

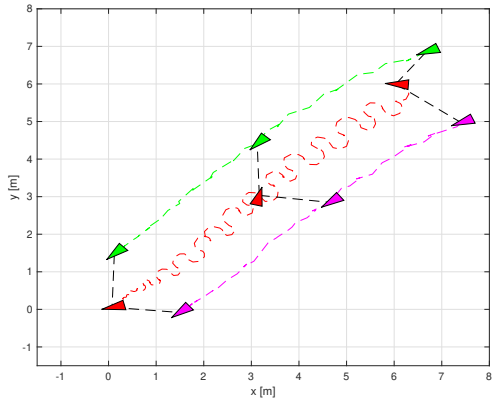
$$\psi_{\min} < \psi_d - \frac{(\psi_{\min} - \psi_d)^2}{2\beta_\psi}\sqrt{\frac{k_z\delta_\psi}{k_\psi}} \tag{C-1.2e}$$

$$\psi_{\max} > \psi_d + \frac{(\psi_{\max} - \psi_d)^2}{2\beta_\psi}\sqrt{\frac{k_z\delta_\psi}{k_\psi}} \tag{C-1.2f}$$

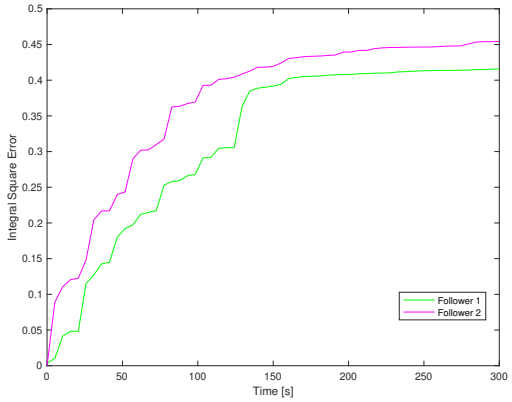
## C-2 Conditions for Obstacle Avoidance

$$k_{\text{obs}} \leq \frac{\omega_i}{G_{\text{obs}}}, \quad \text{where} \quad G_{\text{obs}} = \frac{2\beta_{\text{obs}}}{\psi_{\text{obs}}} \tag{C-2.1}$$

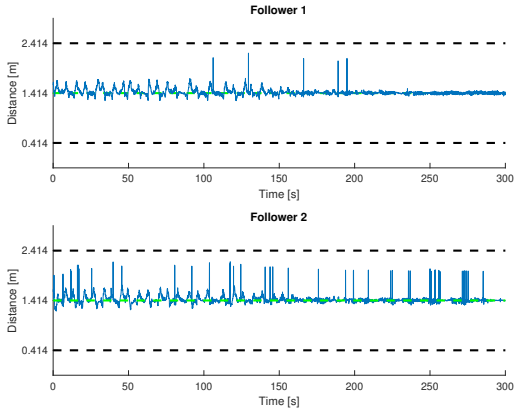
## C-3 No Obstacle



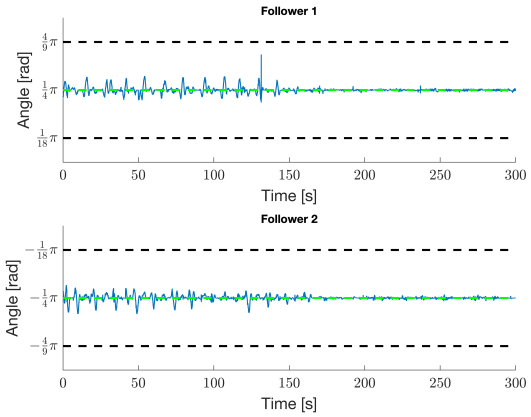

**(a)** Trajectories of ROSbot formation




**(b)** Integral Square Error of formation

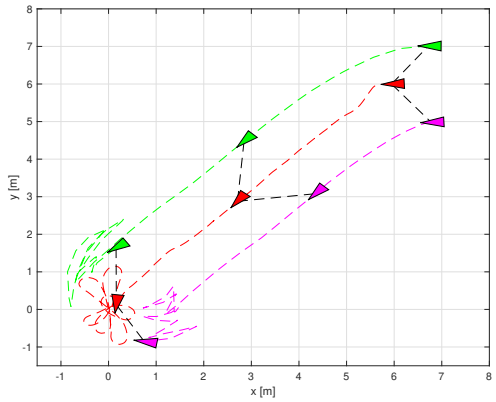



**(c)** Distance w.r.t. leader vehicle

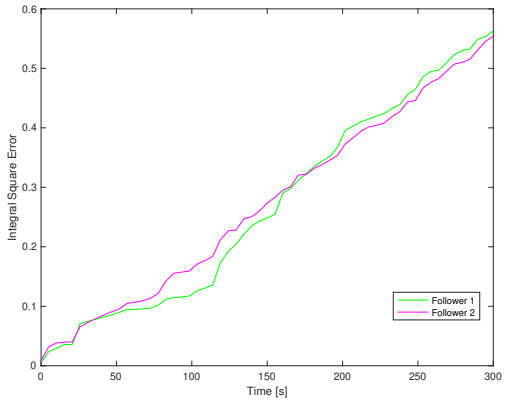



**(d)** Angle w.r.t. leader vehicle

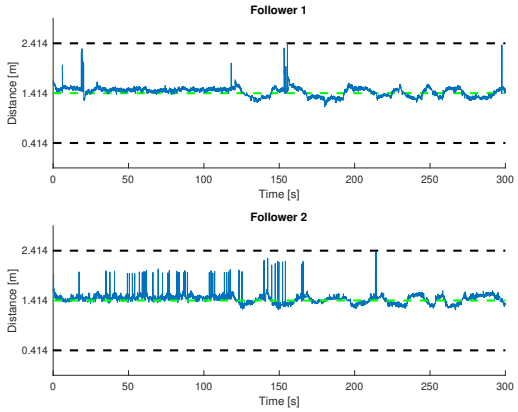
**Figure C-1:** Simulation of formation control algorithm with a leader vehicle using a fixed sensor for source seeking, the parameters listed in table C-1 are utilized. The source is located at $x : 0[m]$, $y : 0[m]$.



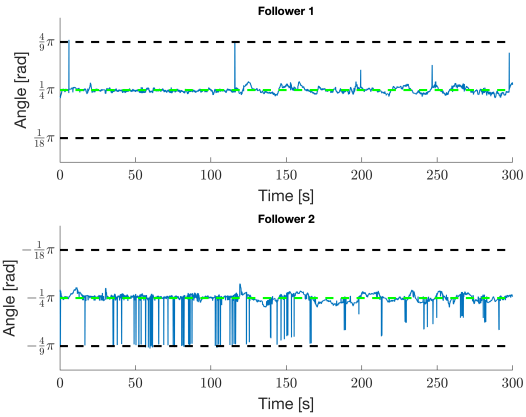

**(a)** Trajectories of ROSbot formation




**(b)** Integral Square Error of formation




**(c)** Distance w.r.t. leader vehicle




**(d)** Angle w.r.t. leader vehicle

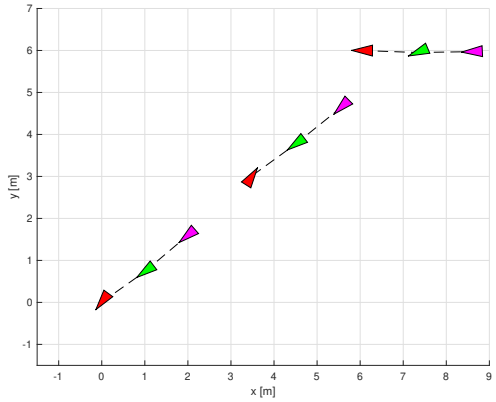
**Figure C-2:** Simulation of formation control algorithm with a leader vehicle using an oscillating sensor for source seeking, the parameters listed in table C-1 are utilized. The source is located at $x : 0[m]$, $y : 0[m]$.

**Table C-1:** Control parameters for triangle formation shape

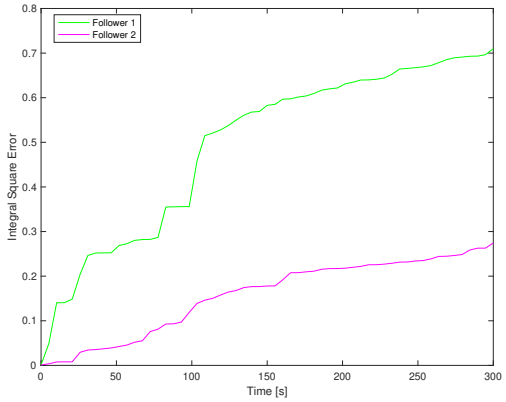
| Parameter | Value | Unit (SI) |
|---|---|---|
| $\psi$ | $\frac{1}{4}\pi$ | $rad$ |
| $z$ | 1.414 | $m$ |
| $\psi_{range}$ | $\frac{7}{36}\pi$ | $rad$ |
| $z_{range}$ | 1 | $m$ |
| $\beta_z$ | 2.5 | $unitless$ |
| $\beta_\psi$ | 1.35 | $unitless$ |
| $k_z$ | 0.01 | $unitless$ |
| $\delta_z$ | 0.6 | $unitless$ |
| $k_\psi$ | 0.6 | $unitless$ |
| $\delta_\psi$ | 0.1 | $unitless$ |

**Table C-2:** Control parameters for snake formation shape

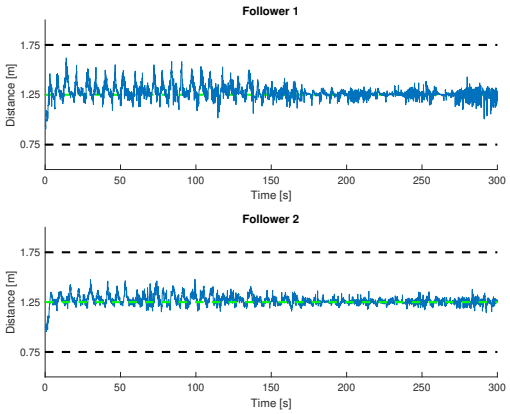
| Parameter | Value | Unit (SI) |
|---|---|---|
| $\psi$ | 0 | $rad$ |
| $z$ | 1.25 | $m$ |
| $\psi_{range}$ | $\frac{7}{36}\pi$ | $rad$ |
| $z_{range}$ | 0.5 | $m$ |
| $\beta_z$ | 2.5 | $unitless$ |
| $\beta_\psi$ | 1.35 | $unitless$ |
| $k_z$ | 0.01 | $unitless$ |
| $\delta_z$ | 0.6 | $unitless$ |
| $k_\psi$ | 0.6 | $unitless$ |
| $\delta_\psi$ | 0.1 | $unitless$ |



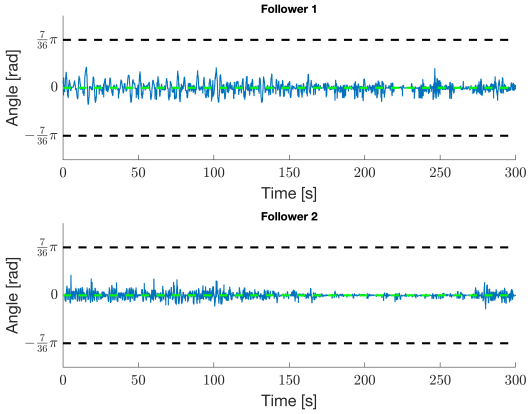

**(a)** Trajectories of ROSbot formation




**(b)** Integral Square Error of formation

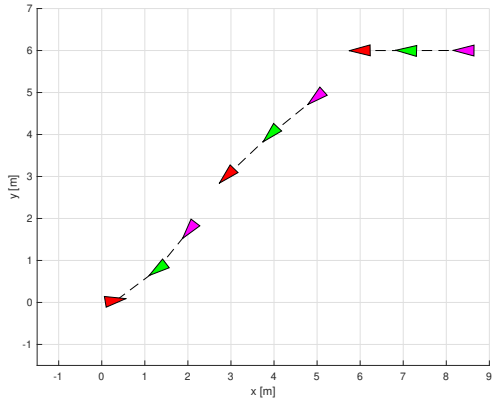



**(c)** Distance w.r.t. leader vehicle

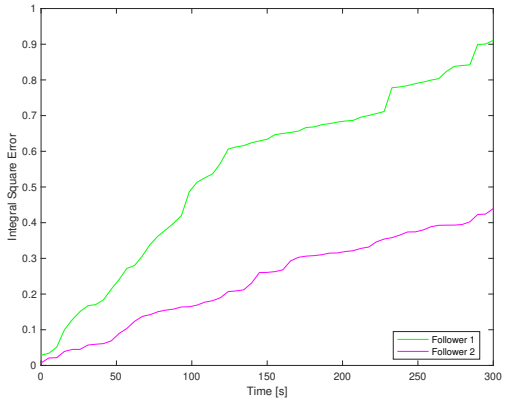



**(d)** Angle w.r.t. leader vehicle

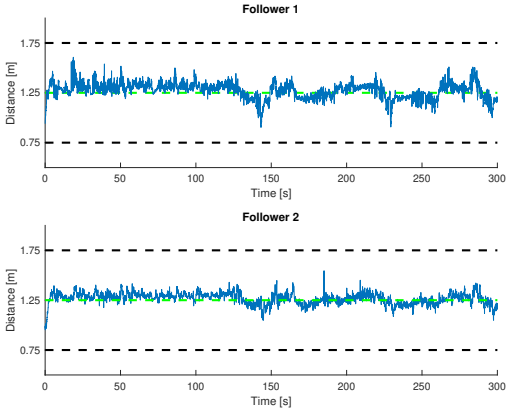
**Figure C-3:** Simulation of formation control algorithm with a leader vehicle using a fixed sensor for source seeking, the parameters listed in table C-2. The source is located at $x : 0[m]$, $y : 0[m]$.



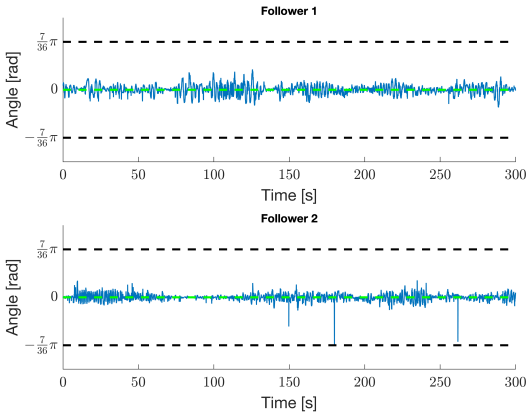

**(a)** Trajectories of ROSbot formation




**(b)** Integral Square Error of formation



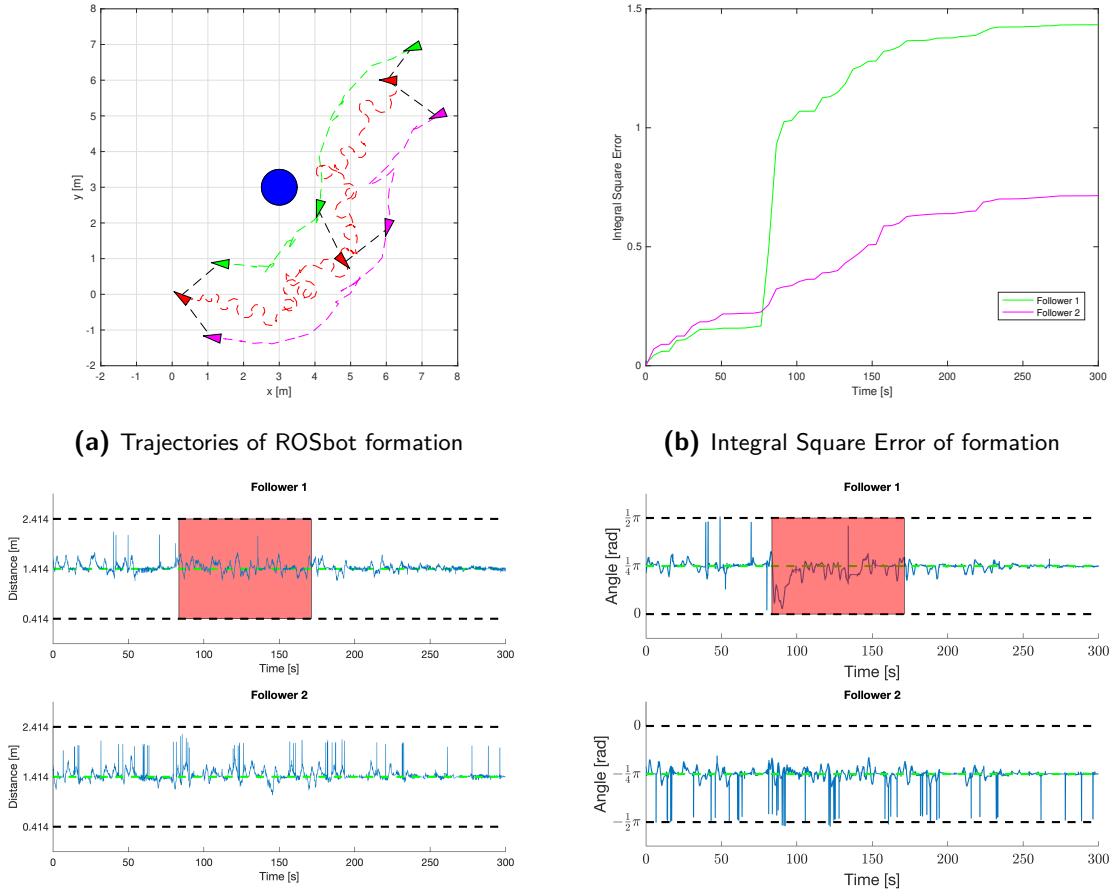

**(c)** Distance w.r.t. leader vehicle




**(d)** Angle w.r.t. leader vehicle

**Figure C-4:** Simulation of formation control algorithm with a leader vehicle using an oscillating sensor for source seeking, the parameters listed in table C-2. The source is located at $x : 0[m]$, $y : 0[m]$.

## C-4 Obstacle



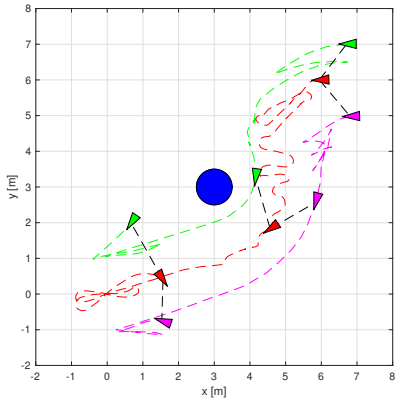

**(a)** Trajectories of ROSbot formation

**(b)** Integral Square Error of formation

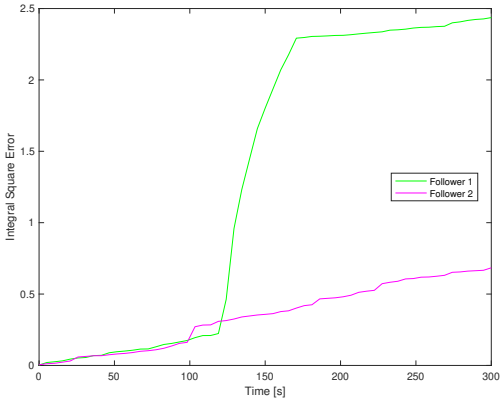
**(c)** Distance w.r.t. leader vehicle

**(d)** Angle w.r.t. leader vehicle

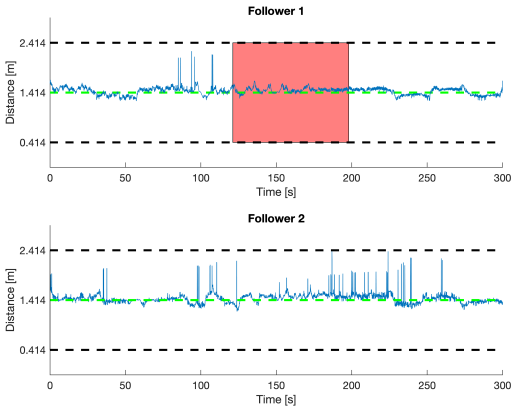
**Figure C-5:** Simulation of formation control algorithm with a leader vehicle using a fixed sensor for source seeking, the parameters listed in table C-3. The source is located at $x : 0[m]$, $y : 0[m]$.



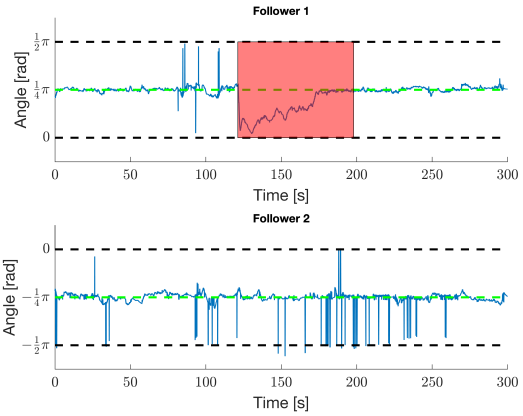

**(a)** Trajectories of ROSbot formation




**(b)** Integral Square Error of formation

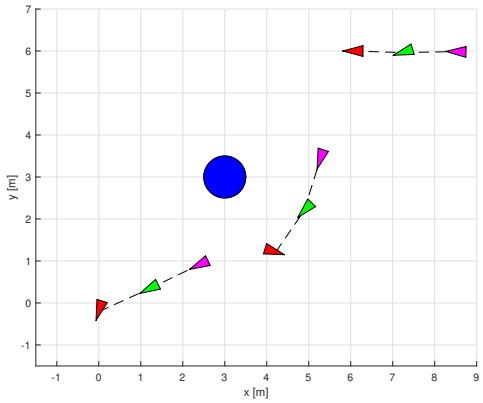



**(c)** Distance w.r.t. leader vehicle

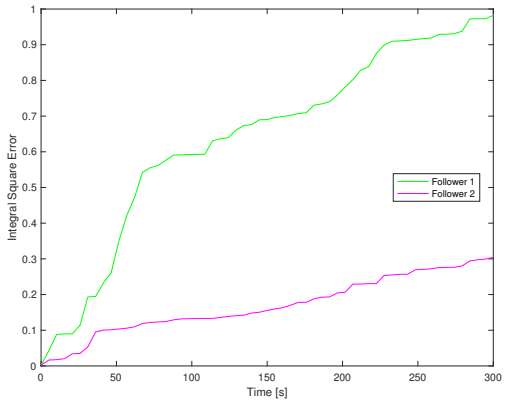



**(d)** Angle w.r.t. leader vehicle

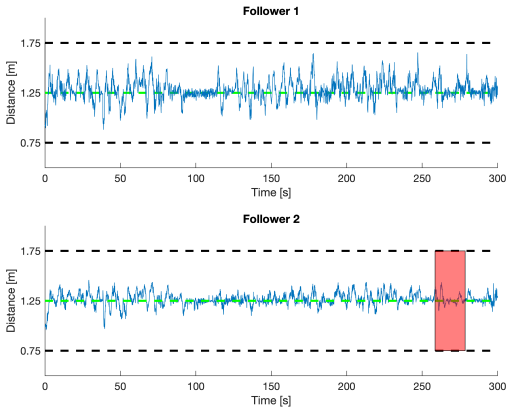
**Figure C-6:** Simulation of formation control algorithm with a leader vehicle using an oscillating sensor for source seeking, the parameters listed in table C-3. The source is located at $x : 0[m]$, $y : 0[m]$.



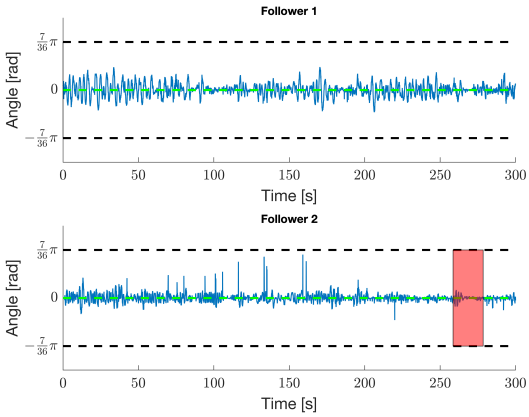

**(a)** Trajectories of ROSbot formation




**(b)** Integral Square Error of formation




**(c)** Distance w.r.t. leader vehicle




**(d)** Angle w.r.t. leader vehicle

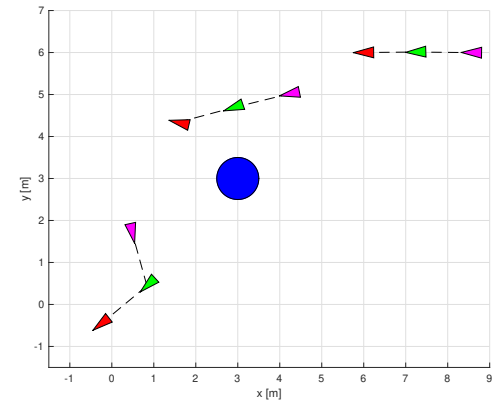
**Figure C-7:** Simulation of formation control algorithm with a leader vehicle using a fixed sensor for source seeking, the parameters listed in table C-4. The source is located at $x : 0[m]$, $y : 0[m]$.

**Table C-3:** Control parameters for triangle formation shape

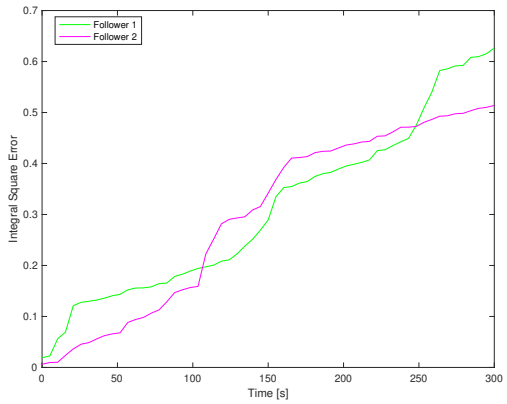
| Parameter | Value | Unit (SI) |
|---|---|---|
| $\psi$ | $\frac{1}{4}\pi$ | $rad$ |
| $z$ | 1.414 | $m$ |
| $\psi_{range}$ | $\frac{1}{4}\pi$ | $rad$ |
| $z_{range}$ | 1 | $m$ |
| $\beta_z$ | 2.5 | $unitless$ |
| $\beta_\psi$ | 1.35 | $unitless$ |
| $k_z$ | 0.01 | $unitless$ |
| $\delta_z$ | 0.6 | $unitless$ |
| $k_\psi$ | 0.6 | $unitless$ |
| $\delta_\psi$ | 0.1 | $unitless$ |
| $\psi_{obs}$ | $0.5\pi$ | $rad$ |
| $k_{obs}$ | 0.6 | $unitless$ |
| $\beta_{obs}$ | 0.75 | $unitless$ |

**Table C-4:** Control parameters for snake formation shape

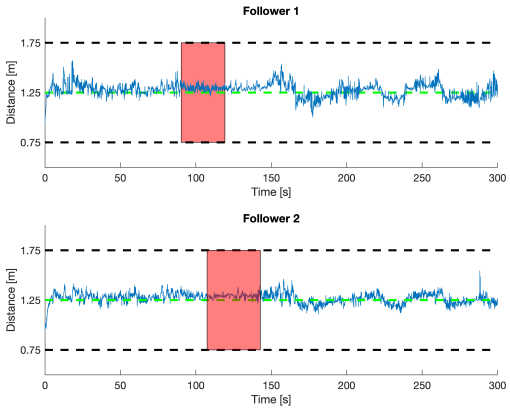
| Parameter | Value | Unit (SI) |
|---|---|---|
| $\psi$ | 0 | $rad$ |
| $z$ | 1.25 | $m$ |
| $\psi_{range}$ | $\frac{7}{36}\pi$ | $rad$ |
| $z_{range}$ | 0.5 | $m$ |
| $\beta_z$ | 2.5 | $unitless$ |
| $\beta_\psi$ | 1.35 | $unitless$ |
| $k_z$ | 0.01 | $unitless$ |
| $\delta_z$ | 0.6 | $unitless$ |
| $k_\psi$ | 0.6 | $unitless$ |
| $\delta_\psi$ | 0.1 | $unitless$ |
| $\psi_{obs}$ | $0.5\pi$ | $rad$ |
| $k_{obs}$ | 0.2 | $unitless$ |
| $\beta_{obs}$ | 0.75 | $unitless$ |



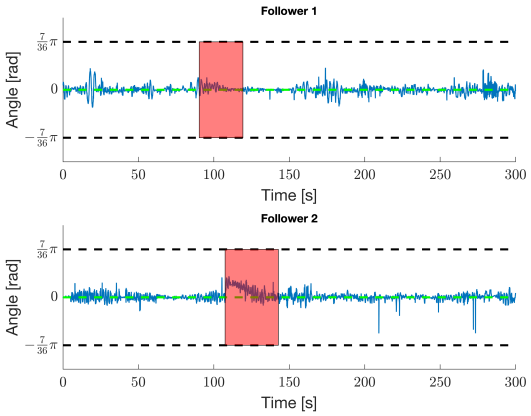

**(a)** Trajectories of ROSbot formation




**(b)** Integral Square Error of formation




**(c)** Distance w.r.t. leader vehicle




**(d)** Angle w.r.t. leader vehicle

**Figure C-8:** Simulation of formation control algorithm with a leader vehicle using an oscillating sensor for source seeking, the parameters listed in table C-4. The source is located at $x : 0[m]$, $y : 0[m]$.

# Appendix D

# Paper

# Multi-Agent Source Seeking in Unknown Environments

Max van der Linden, Suad Krilašević, Sergio Grammatico

***Abstract*—This paper presents a hybrid adaptive feedback (HAF) algorithm suitable for a non-holonomic mobile robot. The algorithm's objective is to navigate a group of three non-holonomic mobile robots towards the unknown source of a signal while maintaining a predefined formation shape and avoiding an obstacle. The HAF algorithm uses the approach presented by [1] as a framework to overcome topological obstructions caused by the obstacle, which would result in the robots getting trapped or crashing into the obstacle if a conventional artificial potential function (APF) approach is chosen. In contrast to [1], the algorithm presented in this paper does not require the location and orientation of the obstacle w.r.t. the source to be known *a priori*. Subsequently, two follower mobile robots are used that employ a decentralized leader-follower control strategy. The follower vehicles track the leader vehicle by maintaining the desired bearing angle and relative distance while avoiding an obstacle and preventing inter-vehicle collisions.**

**Index terms – Non-holonomic mobile robots, Source seeking, Hybrid adaptive feedback, Formation control.**

## I. Introduction

The deployment of autonomous mobile robots to explore unknown environments has been a growing field of research, as stated by [2], [3], and [4]. The use of autonomous robots for scientific or disaster response missions is auspicious when sending a human operator is undesirable, if the mission is too dangerous or burdensome [5]. A possible task for a mobile robot in such a mission could be to locate a source that emits a signal that attenuates over distance w.r.t. the source. These signals could be chemical, biological, or thermal in nature and generated by physical processes, e.g., gas leakage. For these unknown environments, position information is often unavailable due to the absence of GPS. Control algorithms capable of dealing with these types of source seeking problems have been applied to a wide variety of engineering applications [4].

One challenge that arises when using source seeking control to explore unknown environments is the addition of obstacle avoidance to the control objectives. A common method to guide a robotic vehicle around obstacles is to augment the signal the source seeking control law is using to find the source. Often, the signal is augmented by a potential function. However, as stated by [6], [1], and [7], obstacles that interfere with the vehicle's trajectory when using a smooth source seeking control law preclude robust stabilization of the algorithm. Because of the topological obstructions induced by the obstacle, it could lead the source seeking vehicle to prematurely assume it has reached the source from a certain set of initial conditions or hit the obstacle. In [1], a HAF law is presented that claims to overcome this problem.

However, the authors made two assumptions in [1] that limit the applicability of the algorithm for exploring unknown environments. The first assumption being that the obstacle's location is assumed to be known *a priori*, and the second is that obstacle's orientation w.r.t. the source is assumed to be known *a priori*. Therefore, this paper presents a novel approach, using the work presented in [1] as a framework, that does not require these two assumptions to be known *a priori*. Other than in [1], where a HAF algorithm is employed on a two-dimensional point mass, or in [8] where the HAF algorithm is used on a UAV modeled with single integrator dynamics. No research has yet been published on the applicability of a HAF approach to non-holonomic unicycles. Because a significant share of robotic vehicles are modeled as a non-holonomic unicycle, the HAF approach could offer an interesting solution to the problem described in [7].

Furthermore, this paper studies how to implement an algorithm suitable for navigating multiple vehicles in a coordinated formation towards a source while the HAF algorithm controls one leader vehicle. A decentralized leader-follower architecture is chosen similar to the approach presented in [9]. The approach allows each follower robot to compute their desired control input solely on their locally obtained measurements and does not require communication between vehicles.

The remainder of this paper is structured as follows: In Section II, the preliminary background for the mobile robot used in this paper is described. The problem statement of this paper is defined in Section III. Thereafter, in Section IV, the applied source seeking approach is presented. In Section V, by describing the developed HAF algorithm suitable for non-holonomic unicycles. Section VI, explains the implementation formation control strategy for the two follower vehicles. Finally, in Sections VIII based on simulations, conclusions are drawn on the algorithm's performance developed in this paper.

## II. Preliminaries

The algorithm presented in this paper is developed for mobile agents that can be modeled as a unicycle. The equations of motion corresponding to the unicycle's geometric center are shown in Eq. 1.

$$\dot{r}_c = v e^{j\theta} \tag{1a}$$

$$\dot{\theta} = \Omega \tag{1b}$$

The authors are with the Delft Center for Systems and Control, TU Delft, The Netherlands. E-mail address: m.j.vanderlinden@student.tudelft.nl.

The unicycle's geometric center is expressed as a complex variable $r_c$ in $2D$, $v$ and $\Omega$ are the forward and angular velocity inputs and $\theta$ is the unicycle's orientation. The forward velocity of the unicycle is defined by $\dot{r}_c$ and the unicycle's angular velocity by $\dot{\theta}$.

A sensor equipped on the unicycle is used to measure the relative distance w.r.t. nearby objects. The sensor is able to differentiate an object from being an obstacle or one of the formation's two other mobile robots.

## III. Problem Statement

This paper aims to develop and implement a control algorithm capable of steering a group of three autonomous unicycle vehicles towards an unknown source of a signal while avoiding an obstacle that interferes with its path and without inter-vehicle collisions. The vehicle can only acquire the scalar value of the signal strength emitted by the source locally. Because generally, it is not possible to measure the gradient of a signal field, and thus the gradient needs to be estimated based on local measurements only. The source emits a nonlinear signal field assumed to be quadratic, as given in Eq. 2.

$$J = f(x, y) = f^* - q_x(x - x^*)^2 - q_y(y - y^*)^2 \qquad (2)$$

The variables $q_x$ and $q_y$ are unknown positive constants, and $f^* = f(x^*, y^*)$ being the unknown maximum of the signal field at position: $[x^*, y^*]$. The scalar signal the vehicle can locally measure is defined by $J$.

Besides seeking an unknown source of a signal, the presented algorithm in this paper also needs to avoid an obstacle without *a priori* information on its location and orientation w.r.t. the source. The obstacle is also assumed to be static and not to influence the signal field produced by Eq. 2.

## IV. Source Seeking

Employing a unicycle vehicle to move in a particular manner to allow for gradient estimation is much more difficult to achieve than with a point mass due to the kinematic constraints on the vehicle, as stated by [10]. However, it is still possible to create a stable source seeking algorithm to navigate the robot towards the source of a signal, as will be further shown in this section.

The unicycle source seeking method of [11] is applied in this paper because the strategy appears to exhibit the best performance for mobile robots modeled as a unicycle. It employs extremum seeking to tune both $\Omega$ directly and $v$ indirectly. The velocity inputs are tuned by applying sinusoidal perturbation-based sources seeking directly on the angular speed and adding simple derivative-like feedback to a constant forward velocity $(V_c)$. Using this approach, according to the authors in [11], the best features from [2] and [10] are combined. This will result in the vehicle slowing down when approaching the source and converging to a closer proximity w.r.t. the source without diminishing the convergence speed.

To measure the scalar signal strength emitted by the source, described by Eq. 2, a sensor is used. The sensor's location can either be placed at a $R$ distance away from the vehicle's center or collocated with the center. The sensor will be placed at a $R > 0$ distance to improve the source-seeking algorithm's convergence rate, as noted in [2]. The authors in [2] demonstrate that by placing the sensor at a $R > 0$ compared to $R = 0$ distance from the vehicle's center, the trajectory towards the source is improved. Because mounting the sensor off-center on the vehicle will cause the sensor to "sweep" the signal field. Thereby requiring less movement of the vehicle itself to approximate the signal gradient, resulting in improved performance. When the sensor is collocated with the vehicle's center, the vehicle must perform much sharper turns. Hence, using the whole vehicle to probe the signal field and not just the vehicle's outer end.

Instead of the sensor design in [11] that uses a fixed sensor w.r.t. the mobile robot's heading, the sensor in this paper is modeled based on [12]. The authors present a design that decouples the mobile robot's heading from the sensor's location. By placing the sensor on the tip of an arm with length $R$, which is connected to the vehicle's geometric center. The arm can rotate w.r.t. the vehicle's center. As a result, the sensor is oscillating w.r.t. to the mobile robot's heading. In Eq. 3, the sensor dynamics are shown using the design of [12].

$$r_s = r_c + Re^{j(\theta+\theta_s)} \qquad (3)$$

As can be seen from Eq. 3, the sensor location $r_s$ is determined by unicycle's heading $\theta$, an additional angle $\theta_s$, the unicycle's geometric center $r_c$, and the off-center distance of the sensor $R$. The angle $\theta_s$ is determined by the sinusoidal perturbation term defined in Eq. 4.

$$\theta_s = \alpha \sin \omega t \qquad (4)$$

Hence, the sensor is capable of oscillating along the vehicle's geometric center $r_c$ and the sensor position w.r.t. the global $x$-$y$ coordinate frame becomes a function of the orientation of the vehicle $\theta$ and the angle between the centerline of the vehicle and sensor $\theta_s$. Thereby, the sinusoidal perturbation term does not directly affect the vehicle's dynamics, as in [11] where the position of the sensor is fixed w.r.t. the vehicle's heading $\theta$.

The source seeking controller developed by [12] employs the control laws for the input velocities shown in Eq. 5. Where extremum seeking is directly applied on the angular velocity input $\Omega$, which is expressed in Eq. 5a and derivative-like feedback on the forward speed input $v$ in Eq. 5b. The signal strength measurements are fed into a high-pass filter to remove the "DC-component" of the filter's output that is expressed by $\xi$ in Eq. 5.

$$\Omega = c\xi \sin(\omega t) \qquad (5a)$$

$$v = V_c + b\xi \qquad (5b)$$

In Eq. 5a, parameter $c$ is used as gain for the estimated angular gradient $\xi \sin(\omega t)$. The $b$ parameter acts as a derivative term on the forward velocity in Eq. 5b.

## V. Obstacle Avoidance

In order to study the claimed benefit of using a HAF law to avoid obstacles. First, the conditions under which the conventional APF approach would fail are studied in Section V-A while the mobile robot is using the source seeking controller described in Section IV. Thereafter, the HAF approach is elaborated on in Section V-B. Finally, in Sections V-C and V-D two solutions are presented that overcome the assumptions needed for the HAF approach of [1].

### A. Artificial Potential Function

As illustrated in Figure 1 from [13], when applying the source seeking controller directly to navigate towards a source in an environment with an obstacle can be problematic. Due to the existence of a detrimental line $M$, which occurs even if an APF method is applied to repel the mobile robot away from the obstacle. The line $M$ exists because of topological obstructions induced by the obstacle, which results in the mobile robot to converge to line $M$ and not to set $K$ needed to reach the source.

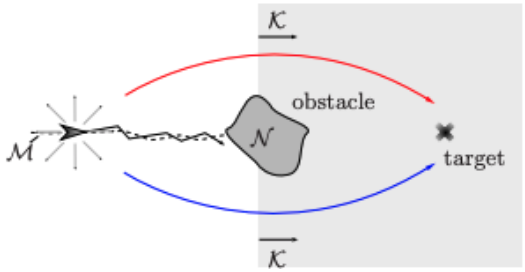



Fig. 1: Global steering towards a target with obstacle avoidance, from [13]. From initial conditions above the (dashed) line $M$, the trajectories approach set $K$ from above the obstacle, while from initial conditions below the (dashed) line, the trajectories approach the set $K$ from below the obstacle. In the presence of measurement noise, a trajectory could stay in a neighborhood of the (dashed) line, potentially causing the vehicle to crash into the obstacle.

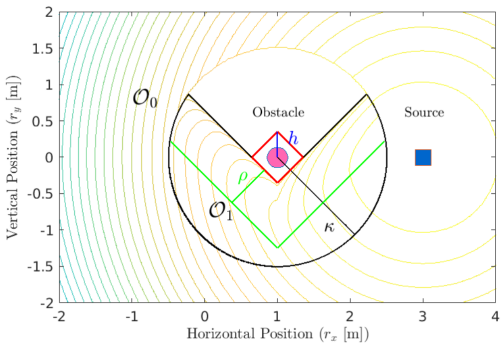
### B. Hybrid Adaptive Feedback

To prevent the situation depicted in Figure 1 from happening, a HAF algorithm that assures that the vehicle reaches set $K$ is developed. Similar to [3], the state space $\mathbb{O}_q$ of the vehicle's adaptive control law is divided into states $q \in \{0, 1, 2\}$. Which transforms Eq. 2 into a mode-dependent localization function $J_q(x, y)$ shown in Eq. 6, where the output of Eq. 6 is fed to a HAF law.

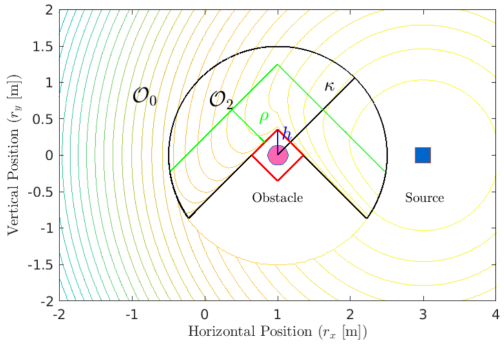
$$J_q(x, y) = -J(x, y) + B(d_q(x, y), J(x, y)) \tag{6}$$

Where $d_q(x, y) = |[x, y]^T|^2_{\mathbb{R}^2 \setminus \mathbb{O}_q}$ is a function that maps the position of the vehicle $[x, y]^T \in \mathbb{R}^2$ to the squared value of its distance to the set $\mathbb{R}^2 \setminus \mathbb{O}_q$, see Figure 2. Hence, for the two switch states $q \in \{1, 2\}$, a specific localization function is computed for each switch state, e.g. $J_1(x, y)$ and $J_2(x, y)$. The resulting $d_q(x, y)$ is an argument for the barrier function $B(\cdot)$, which is defined in Eq. 7 for a logarithmic barrier function. The computed barrier values are multiplied by $(|J| + 1)$ to ensure that the barrier value is dominant w.r.t. the measured scalar signal strength. To preserve continuity of the mode depended signal, the barrier function should reduce to zero when the squared value distance equals $\rho$ as stated by [3].

$$B(z, J) = \begin{cases} (|J| + 1)(z - \rho)^2 \log\left(\frac{\rho}{z}\right), & \text{if } z \in [0, \rho] \\ 0, & \text{if } z > \rho, \end{cases} \tag{7}$$




(a) Partitioned space for modes 0 and 1




(b) Partitioned space for modes 0 and 2

Fig. 2: Partitioned state space into two modes $q \in \{1, 2\}$. Where a virtual box (red) is created around the obstacle with tunable height $h$, the edges of the box are extended to intersect the detection perimeter $\kappa$. The edges determine the mode-dependent spaces with $\rho$ being the limit for the barrier function, from [3].

The logic for the HAF algorithm used in this paper is based on [3], the authors add an extra state ($q = 0$) w.r.t. the HAF method of [1]. The $q = 0$ state is used as a radial bound in case the mobile robot is far enough w.r.t. the obstacle to not require augmentation of the signal for obstacle avoidance.

The HAF logic is as follows, if the vehicle's distance w.r.t. the obstacle $(z)$ is larger than distance $\kappa$, the corresponding switch state is $q = 0$ and the scalar signal is not augmented by Eq. 7. If $z$ is smaller than $\kappa$, the mobile robot can be in a space that covers $\mathbb{O}_1$, $\mathbb{O}_2$ or covered by both.

If the mobile robot is in the space only covered by $\mathbb{O}_1$, then $q = 1$ and $J_1(x, y)$ is used as signal for the source seeking controller. Alternatively, if the space is only covered by $\mathbb{O}_2$,

then $q = 2$ and $J_2(x, y)$ is used. In case the mobile robot is in the space covered by both $\mathbb{O}_1$ and $\mathbb{O}_2$. The hybrid switch logic is used, namely, if $q = 1$ and $J_1 \geq (\mu - \lambda)J_2$ then the state is updated to $q = 2$ else $q = 1$. Vice versa, if $q = 2$ and $J_2 \geq (\mu - \lambda)J_1$ then $q = 1$ else $q = 2$. If the state is $q = 0$, then as initial mode $q = 1$ is selected and $J_1(x, y)$ is used as signal strength.

The parameter $\mu > 1$ is used to prevent recurrent jumps between the two modes, while the $\lambda \in (0, \mu - 1)$ parameter is applied to guarantee that $O_1$ and $O_2$ overlap for establishing robustness, as stated by [1].

### C. Obstacle Location

To determine where the obstacle is located in the environment, a sensor on the mobile robot is used, as explained in Section II. The obstacles are approximated with a circular geometric model, which outputs the circle's estimated radius and center point w.r.t. the unicycle.

Based on the information acquired by the unicycle's sensor, the HAF algorithm can determine whether the unicycle is in state $q = 0$ or in the hybrid states $q \in \{1, 2\}$ shown in Figure 2.

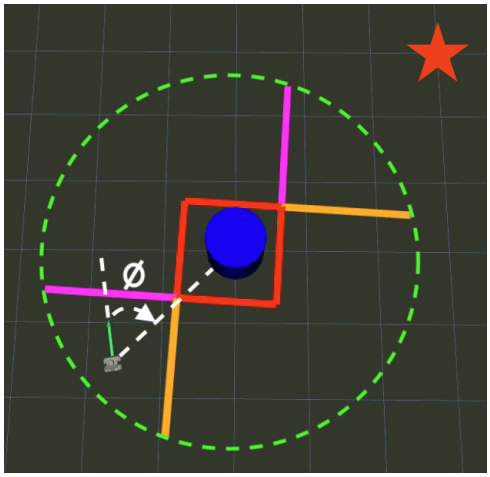
### D. Partitioning Lines

Since the obstacle's location and its orientation w.r.t. the source is not assumed to be known *a priori*, an approach has to be developed to draw the partitioning lines of the state space as shown in Figure 2.

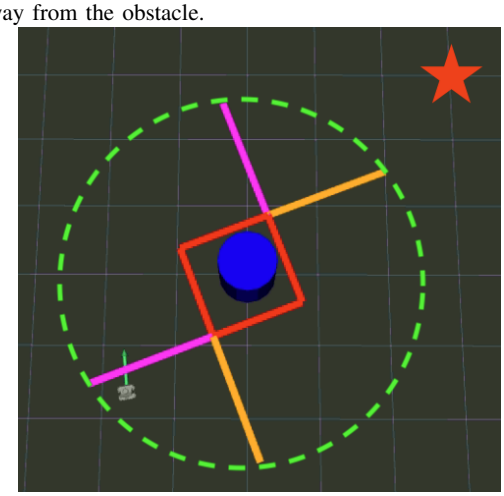
To start, the acquired sensor data on the obstacle is approximated using the circular geometric model described in section V-C. The algorithm works as follows; first, a virtual box around the detected circular obstacle is drawn based on the circle's computed center point and radius, as described in section V-C. If the center point of the obstacle is within $\kappa$ distance of the vehicle and the vehicle is orientated straight towards the obstacle. Then, the partitioning lines can be drawn from the closest corner of the virtual box w.r.t. the robot. The lines are drawn with a $\frac{1}{4}\pi$ angle from the line between the robot and the closest virtual box corner until they intersect $\kappa$.

However, if the vehicle is not orientated straight towards the obstacle and the source, the partitioning lines' drawing becomes more difficult. As discussed above, the orientation of the obstacle w.r.t. the source in the global coordinate frame is not assumed to be known *a priori*. Therefore, first, the angle $\phi$ between the robot and obstacle is computed. Thereby, assuming that the robot is orientated towards the source, the angle $\phi$ determines how much the partitioning lines should be rotated. So, in case the robot is orientated at a $\phi$ angle away from the obstacle as illustrated in Figure 3a. The partitioning lines would need to be rotated by $\phi$, as shown in Figure 3b.

In [1], it is assumed that the correct orientation of the partitioning lines w.r.t. the source is known to the vehicle. However, this assumption implies knowing the heading towards the source of the signal field *a priori* which defeats the need for a source seeking method that estimates the gradient. When the robot is within $\kappa$ distance of the obstacle, and it makes a sharp turning movement while staying close to its location, the partitioning lines would therefore need to be rotated equally. However, this behavior is not desirable because sudden turns are a property of the employed source seeking method and not due to significant gradient changes of the signal field. Therefore, a low-pass filter is applied on $\phi$ to prevent high-frequency changes in $\phi$ to influence the partitioning lines' rotation. As low-pass filter, a moving average filter is used with length $N$.




(a) The partitioning lines are not aligned with the robot's heading. Because, the robot is now orientated at an $\phi$ angle away from the obstacle.



(b) The partitioning lines rotated by $\phi$ corresponding to how far the mobile robot is rotated away from the source.

Fig. 3: Figure showing how in Rviz the partitioning lines (in magenta and orange) are drawn w.r.t. to the source (red star) when the mobile robot is orientated away from the source and obstacle (blue cylinder). The robot is located in the bottom left corner with the green arrow being its heading and $\phi$ describing the angle between the robot's heading and the obstacle.

## VI. Formation Control

A decentralized leader-follower strategy will be employed in this paper for the formation control algorithm. In a leader-follower strategy, one vehicle is designated as a leader vehicle

to navigate the formation towards a source while utilizing the HAF algorithm, and the other two follower vehicles need to meet the formation control objectives. The followers' main objective is to operate under the leader vehicle's guidance and maintain the multi-vehicle system's formation shape.

The formation shape is maintained by controlling the desired relative angle $\psi_d$ and distance $z_d$ between the leader and the follower vehicles, as shown in Figure 4. To achieve this, the follower vehicles use a feedback controller for leader-tracking from [9] to drive the error limits in Eq. 8 to zero.

$$\lim_{t\to\infty}(z - z_d) = 0 \tag{8a}$$

$$\lim_{t\to\infty}(\psi - \psi_d) = 0 \tag{8b}$$

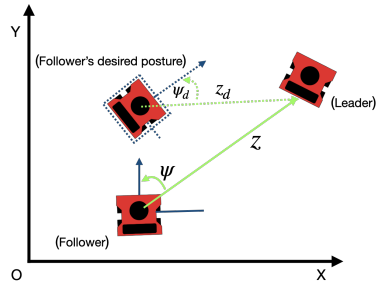



Fig. 4: The leader-follower formation scheme's control objective, where the follower vehicle's goal is to maintain a desired posture w.r.t. the leader by reducing Eq. 8 to zero.

The errors in Eq. 8, are computed by the difference between the perceived distance $z$ $[m]$ and angle $\psi$ $[rad]$ w.r.t. the leader and $\psi_d$ $[rad]$ and $z_d$ $[m]$.

The approach presented by [9], is employed on the follower vehicles to control the formation shape and avoid an obstacle. [9] use barrier functions to steer the follower vehicles to a predefined $\psi_d$ and $z_d$ w.r.t. the leader. If an obstacle is within proximity of the follower vehicle, another barrier function with angle $\psi_{obs}$ as argument, is used to steer the follower vehicle away from the obstacle. In Figure 5, the basic principle is illustrated. Whenever the vehicle is within $z_{obs,range}$ and in $\Theta_{TF}$, the follower vehicle's angular velocity input is adjusted using the barrier function with $\psi_{obs}$ as argument to steer the vehicle away from the obstacle. If the vehicle is within $z_{obs,safe}$, the angular velocity input is solely used for avoiding the obstacle and not tracking the leader. In that case, the vehicle's forward velocity is set to a small positive if in $\Theta_{SB}$ or small negative if in $\Theta_{SF}$. If the vehicle is in $\Theta_{TB}$, no obstacle avoidance action is required.

By employing this method, a flexible triangle-formation is maintained by the follower vehicles. The follower vehicles can differentiate between fellow vehicles and an obstacle using the obtained radius of an object by the unicycle's sensor, as described in Section II. To prevent collisions between follower vehicles, a similar approach is used as described above for obstacle avoidance that adjusts the angular velocity using a barrier function.

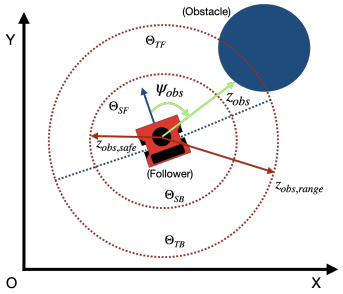



Fig. 5: Follower vehicle's obstacle approach based on [9].

## VII. Simulation Results

As an experimental platform, the Husarion ROSbot [14] is utilized, which a wheeled mobile robot. The dynamics of the ROSbot are modeled as a unicycle with two virtual wheels through the robot's geometric center, as expressed in Eq. 1.

In a Gazebo simulator environment, a simulation scenario is created where the obstacle is placed at $x : 3[m]$, $y : 3[m]$ (blue), which is a cylinder with a radius of $0.5[m]$. The parameters for the signal field in Eq. 2, are set to $f^* = 1$, $q_x = 1$, $q_y = 1$, $x^* = 0$ and $y^* = 0$. The initial position of the ROSbot formation is set as follows, the leader vehicle using the HAF algorithm is located at $x : 6[m]$, $y : 6[m]$, and the other two follower vehicles at $x : 6[m]$, $y : 7.4142[m]$ and $x : 7.4142[m]$, $y : 6[m]$. All vehicles have their initial heading pointed towards the source. Using this set-up, the obstacle is aligned with the source and ROSbot. Thereby, if the ROSbot would follow the gradient of the signal field produced by the objective function in Eq. 2, it would crash. The formation's leader vehicle is shown in red, and the other two follower vehicles of the formation in green and magenta. For all simulations, the control objective of the follower vehicles is to maintain a triangle-shaped formation with $\psi_d = \frac{1}{4}\pi[rad]$ and $z_d = \sqrt{2}[m]$.

First, the conditions under which the APF method would fail are studied because of the scenario depicted in Figure 1. In Figure 6, a simulation run is shown where the leader vehicle crashes into the obstacle due to leader vehicle converging to line $M$, while using the parameters listen in Table Ia and Ib.

Subsequently, the HAF algorithm is evaluated while utilizing the same source seeking and barrier potential parameters listed in Table Ia and Ib as for the simulation where the APF method fails. As can be seen from Figure 7a, the formation successfully avoids the obstacle with the HAF parameters of Table Ic. The corresponding graph for the hybrid switch state $q$ is shown in Figure 7b.

## VIII. Conclusions

The final combined algorithm developed in this paper satisfies the main research objective. The algorithm is empirically shown not to suffer from the detrimental line $M$, which would occur if an APF approach is used. As can be seen when comparing Figure 6 to Figure 7a. On top of that, the algorithm is suitable to be implemented on a Husarion ROSbot

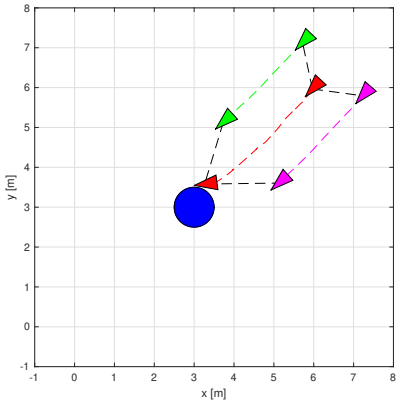
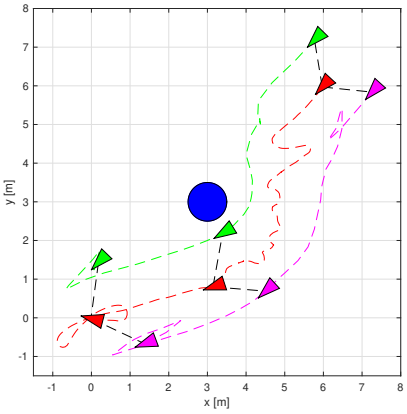
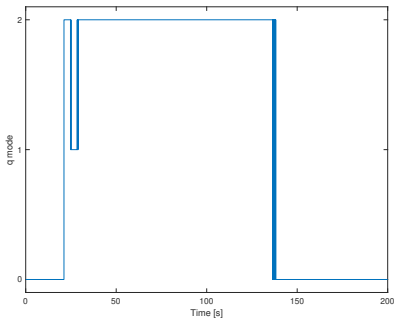



Fig. 6: One simulation run where the APF method employed on the leader vehicle is not capable of steering the formation away from the detrimental line $M$.




(a) Trajectory of the formation




(b) Hybrid switch state graph

Fig. 7: One simulation run where the HAF method employed on the leader vehicle prevents convergence to line $M$ and avoids the obstacle.

platform, and the algorithm is simulated in a Gazebo simulator environment to emulate physical reality. If switching between the hybrid states $q \in \{1, 2\}$ is considered to be too fast, the $\mu$ parameter can be increased.

| Parameter | Value | Unit (SI) |
|---|---|---|
| $\omega$ | 20 | $rad$ |
| $h$ | 1 | unitless |
| $c$ | 80 | unitless |
| $\alpha$ | 1.8 | unitless |
| $R$ | 0.1 | $m$ |
| $b$ | 0 | unitless |
| $V_c$ | 0.1 | $m/s$ |

(a) Source Seeking

| Parameter | Value | Unit (SI) |
|---|---|---|
| $\rho$ | 0.75 | $m$ |
| $h$ | 0.5 | $m$ |

(b) Barrier function

| Parameter | Value | Unit (SI) |
|---|---|---|
| $\kappa$ | 3 | $m$ |
| $(\mu - \lambda)$ | $1 * 10^5$ | *unitless* |
| $N$ | 2500 | *unitless* |

(c) HAF

TABLE I: Parameters used for simulation

There are several interesting areas for further research, including performing physical experiments, studying how the HAF performs when obstacles affect the signal strength and shape, and researching how the HAF can be applied to a multi-agent source seeking unicycle formation like presented in [15].

## REFERENCES


[1] J. I. Poveda, M. Benosman, A. R. Teel, and R. G. Sanfelice, "A hybrid adaptive feedback law for robust obstacle avoidance and coordination in multiple vehicle systems," in *2018 Annual American Control Conference (ACC)*. IEEE, 2018, pp. 616–621.

[2] J. Cochran and M. Krstic, "Source seeking with a nonholonomic unicycle without position measurements and with tuning of angular velocity part i: Stability analysis," in *2007 46th IEEE Conference on Decision and Control*. IEEE, 2007, pp. 6009–6016.

[3] H. Mohr, K. Schroeder, and J. Black, "Distributed source seeking and robust obstacle avoidance through hybrid gradient descent," in *2019 IEEE Aerospace Conference*. IEEE, 2019, pp. 1–8.

[4] C. Zhang and R. Ordóñez, *Extremum-seeking control and applications: a numerical optimization-based approach*. Springer Science & Business Media, 2011.

[5] A. S. Matveev, M. C. Hoy, and A. V. Savkin, "Extremum seeking navigation without derivative estimation of a mobile robot in a dynamic environmental field," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 3, pp. 1084–1091, 2015.

[6] H.-B. Dürr, M. S. Stanković, D. V. Dimarogonas, C. Ebenbauer, and K. H. Johansson, "Obstacle avoidance for an extremum seeking system using a navigation function," in *2013 American Control Conference*. IEEE, 2013, pp. 4062–4067.

[7] R. G. Sanfelice, M. J. Messina, S. E. Tuna, and A. R. Teel, "Robust hybrid controllers for continuous-time systems with applications to obstacle avoidance and regulation to disconnected set of points," in *2006 American Control Conference*. IEEE, 2006, pp. 6–pp.

[8] H. Mohr, "Uav implementation of distributed robust target location in unknown environments," in *2020 IEEE Aerospace Conference*. IEEE, 2020, pp. 1–10.

[9] Y. Wang, D. Wang, S. Yang, and M. Shan, "A practical leader–follower tracking control scheme for multiple nonholonomic mobile robots in unknown obstacle environments," *IEEE Transactions on Control Systems Technology*, vol. 27, no. 4, pp. 1685–1693, 2018.

[10] C. Zhang, D. Arnold, N. Ghods, A. Siranosian, and M. Krstic, "Source seeking with non-holonomic unicycle without position measurement and with tuning of forward velocity," *Systems & control letters*, vol. 56, no. 3, pp. 245–252, 2007.

[11] N. Ghods and M. Krstic, "Speed regulation in steering-based source seeking," *Automatica*, vol. 46, no. 2, pp. 452–459, 2010.

[12] J. Cochran, A. Siranosian, N. Ghods, and M. Krstic, "Source seeking with a nonholonomic unicycle without position measurements and with tuning of angular velocity—part ii: Applications," in *2007 46th IEEE Conference on Decision and Control*. IEEE, 2007, pp. 1951–1956.

[13] R. G. Sanfelice, *Robust hybrid control systems*. University of California, Santa Barbara, 2007.

[14] Husarion. Rosbot manual. [Online]. Available: https://husarion.com/manuals/rosbot-manual/

[15] L. Briñón-Arranz, L. Schenato, and A. Seuret, "Distributed source seeking via a circular formation of agents under communication constraints," *IEEE Transactions on Control of Network Systems*, vol. 3, no. 2, pp. 104–115, 2015.

# Bibliography


[1] Kartik B Ariyur and Miroslav Krstić. *Real time optimization by extremum seeking control*. Wiley Online Library, 2003.

[2] Lara Briñón-Arranz, Luca Schenato, and Alexandre Seuret. Distributed source seeking via a circular formation of agents under communication constraints. *IEEE Transactions on Control of Network Systems*, 3(2):104–115, 2015.

[3] Jennie Cochran and Miroslav Krstic. Source seeking with a nonholonomic unicycle without position measurements and with tuning of angular velocity part i: Stability analysis. In *2007 46th IEEE Conference on Decision and Control*, pages 6009–6016. IEEE, 2007.

[4] W. de Leeuw, M. Ricke, C. Rosier, T. Wielenga, and S. Grammatico. A multi rosbot laboratory setup for experimenting autonomous driving maneuvers. In *2020 28th Mediterranean Conference on Control and Automation (MED)*, pages 532–537, 2020. doi: 10.1109/MED48518.2020.9183306.

[5] Hans-Bernd Dürr, Miloš S Stanković, Dimos V Dimarogonas, Christian Ebenbauer, and Karl Henrik Johansson. Obstacle avoidance for an extremum seeking system using a navigation function. In *2013 American Control Conference*, pages 4062–4067. IEEE, 2013.

[6] Nima Ghods. *Extremum seeking for mobile robots*. PhD thesis, UC San Diego, 2011.

[7] Nima Ghods and Miroslav Krstic. Speed regulation in steering-based source seeking. *Automatica*, 46(2):452–459, 2010.

[8] M Hamza. Extremum control of continuous systems. *IEEE Transactions on Automatic Control*, 11(2):182–189, 1966.

[9] Michael Hoy, Alexey S Matveev, and Andrey V Savkin. Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey. *Robotica*, 33 (3):463–497, 2015.

[10] Husarion. Rosbot manual. URL https://husarion.com/manuals/rosbot-manual/.

[11] Husarion. Rosbot description. https://github.com/husarion/rosbot_description, 2020.

[12] Yuanchang Liu and Richard Bucknall. A survey of formation control and motion planning of multiple unmanned vehicles. *Robotica*, 36(7):1019–1047, 2018.

[13] Alexey S Matveev, Michael C Hoy, and Andrey V Savkin. Extremum seeking navigation without derivative estimation of a mobile robot in a dynamic environmental field. *IEEE Transactions on Control Systems Technology*, 24(3):1084–1091, 2015.

[14] Zhiqiang Miao, Hang Zhong, Jie Lin, Yaonan Wang, Yanjie Chen, and Rafael Fierro. Vision-based formation control of mobile robots with fov constraints and unknown feature depth. *IEEE Transactions on Control Systems Technology*, 2020.

[15] Hannah Mohr. Uav implementation of distributed robust target location in unknown environments. In *2020 IEEE Aerospace Conference*, pages 1–10. IEEE, 2020.

[16] Hannah Mohr, Kevin Schroeder, and Jonathan Black. Distributed source seeking and robust obstacle avoidance through hybrid gradient descent. In *2019 IEEE Aerospace Conference*, pages 1–8. IEEE, 2019.

[17] Hannah Dornath Mohr. *Target Locating in Unknown Environments Using Distributed Autonomous Coordination of Aerial Vehicles*. PhD thesis, Virginia Tech, 2019.

[18] Joannes Oroko and GN Nyakoe. Obstacle avoidance and path planning schemes for autonomous navigation of a mobile robot: a review. In *Proceedings of Sustainable Research and Innovation Conference*, pages 314–318, 2014.

[19] Jorge I Poveda, Mouhacine Benosman, Andrew R Teel, and Ricardo G Sanfelice. A hybrid adaptive feedback law for robust obstacle avoidance and coordination in multiple vehicle systems. In *2018 Annual American Control Conference (ACC)*, pages 616–621. IEEE, 2018.

[20] M. Przybyla. Obstacle detector. https://github.com/tysik/obstacle_detector, 2019.

[21] Mateusz Przybyła. Detection and tracking of 2d geometric obstacles from lrf data. In *2017 11th International Workshop on Robot Motion and Control (RoMoCo)*, pages 135–141. IEEE, 2017.

[22] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[23] Tomáš Rouček, Martin Pecka, Petr Čížek, Tomáš Petříček, Jan Bayer, Vojtěch Šalanský, Daniel Heřt, Matěj Petrlík, Tomáš Báča, Vojěch Spurný, François Pomerleau, Vladimír Kubelka, Jan Faigl, Karel Zimmermann, Martin Saska, Tomáš Svoboda, and Tomáš Krajník. Darpa subterranean challenge: Multi-robotic exploration of underground environments. In Jan Mazal, Adriano Fagiolini, and Petr Vasik, editors, *Modelling and Simulation for Autonomous Systems*, pages 274–290, Cham, 2020. Springer International Publishing.

[24] Ricardo G Sanfelice. *Robust hybrid control systems*. University of California, Santa Barbara, 2007.

[25] Ricardo G Sanfelice, Michael J Messina, S Emre Tuna, and Andrew R Teel. Robust hybrid controllers for continuous-time systems with applications to obstacle avoidance and regulation to disconnected set of points. In *2006 American Control Conference*, pages 6–pp. IEEE, 2006.

[26] SeeedStudio. Rplidar - product details. URL https://www.seeedstudio.com/.

[27] Yogang Singh, Sanjay Sharma, Robert Sutton, and Daniel Hatton. Path planning of an autonomous surface vehicle based on artificial potential fields in a real time marine environment. 2017.

[28] Shanghai Slamtec. Rplidar a2 datasheet. URL https://www.robotshop.com/media/files/pdf2/ld208_slamtec_rplidar_datasheet_a2m8_v1.0_en.pdf.

[29] Max J. van der Linden. Multi-agent source seeking in unknown environments. *Literature Study for Delft Center of Systems and Control*, 2020.

[30] Yuanzhe Wang, Danwei Wang, Shuai Yang, and Mao Shan. A practical leader–follower tracking control scheme for multiple nonholonomic mobile robots in unknown obstacle environments. *IEEE Transactions on Control Systems Technology*, 27(4):1685–1693, 2018.

[31] Greg Welch, Gary Bishop, et al. An introduction to the kalman filter, 1995.

[32] Chunlei Zhang and Raúl Ordóñez. *Extremum-seeking control and applications: a numerical optimization-based approach*. Springer Science & Business Media, 2011.

[33] Chunlei Zhang, Daniel Arnold, Nima Ghods, Antranik Siranosian, and Miroslav Krstic. Source seeking with non-holonomic unicycle without position measurement and with tuning of forward velocity. *Systems & control letters*, 56(3):245–252, 2007.

# Glossary

## List of Acronyms

| | |
|---|---|
| **DCSC** | Delft Center for Systems and Control |
| **ROS** | Robot Operating System |
| **OS** | operating system |
| **DOF** | degrees of freedom |
| **SBC** | single-board computer |
| **LiDAR** | Light Detection And Ranging |
| **ZOH** | Zero-Order Hold |
| **APF** | artificial potential function |
| **HAF** | hybrid adaptive feedback |
| **ISE** | integral square error |
| **IMU** | inertial measurement unit |

## List of Symbols

**Formation Control Variables**

| | |
|---|---|
| $\beta_\psi$ | Angle barrier function parameter |
| $\beta_z$ | Distance barrier function parameter |
| $\beta_{obs}$ | Obstacle barrier function parameter |
| $\delta_\psi$ | Constant positive design parameter for leader tracking control law |
| $\delta_z$ | Constant positive design parameter for leader tracking control law |
| $\psi$ | Measured bearing angle w.r.t. leader vehicle $rad$ |
| $\psi_d$ | Desired bearing angle w.r.t. leader vehicle $rad$ |

$\psi_{max}$, $\psi_{min}$ Upper and lower bound for bearing angle w.r.t. leader $rad$
$\psi_{obs,d}$ Critical bearing angle w.r.t. obstacle $rad$
$\psi_{obs}$ Bearing angle w.r.t. obstacle $rad$
$k_\psi$ Constant positive design parameter for leader tracking control law
$k_z$ Constant positive design parameter for leader tracking control law
$k_{obs}$ Constant positive design parameter for obstacle avoidance control law
$v_{avoid}$ Constant forward velocity for obstacle avoidance control law
$z$ Measured distance w.r.t. leader vehicle $m$
$z_d$ Desired distance w.r.t. leader vehicle $m$
$z_{col,safe}$ Safe distance w.r.t. to other follower vehicle $m$
$z_{max}$, $z_{min}$ Upper and lower bound for distance w.r.t. leader $m$
$z_{obs,range}$ Distance w.r.t. obstacle where obstacle avoidance is active $m$
$z_{obs,safe}$ Distance w.r.t. obstacle where leader tracking stops $m$
$z_{obs}$ Distance w.r.t. obstacle $m$

**Obstacle Avoidance Parameters**

$\kappa$ Perimeter for obstacle avoidance
$\lambda$ Parameter to ensure existence of solution under small perturbations
$\mu$ Parameter to prevent rapid toggling between hybrid states
$\phi$ Bearing angle between ROSbot and obstacle $rad$
$\rho$ Repulsion range
$q$ Hybrid switch mode
$z$ Euclidean distance ROSbot w.r.t. obstacle

**Source Seeking Variables**

$\omega$ Perturbation frequency $rad/s$
$\theta$ The heading of the ROSbot w.r.t. global coordinate frame
$\xi$ High-pass filtered signal strength
$a$ Amplitude parameter for added perturbation term
$b$ Forward velocity derivative gain
$c$ Gain for estimated gradient
$f^*$ unknown extremum value of cost function
$h$ Cut-off frequency of high-pass filter
$q_x$, $q_y$ Positive constants of quadratic cost function
$R$ Distance of sensor w.r.t. geometric center of the ROSbot
$r_c$ Position of the geometric center of the ROSbot
$r_s$ Position of the sensor
$T$ Sampling rate $Hz$
$u$ system input
$V_c$ Added forward velocity constant $m/s$
$x$ measurable state
$x^*$, $y^*$ Unknown position of cost function extremum
$y$ output value