Universitat de les Illes Balears

MASTER'S THESIS

IMPLEMENTATION OF OBSTACLE AVOIDANCE ALGORITHMS FOR UNDERWATER MULTI-ROBOT SYSTEMS

Antoni Raja Mateos

Master's Degree in Intelligent Systems (MUSI)

Specialisation: Mobile Robotics

Centre for Postgraduate Studies

Academic Year 2023-24

IMPLEMENTATION OF OBSTACLE AVOIDANCE ALGORITHMS FOR UNDERWATER MULTI-ROBOT SYSTEMS

Antoni Raja Mateos

Master's Thesis
Centre for Postgraduate Studies
University of the Balearic Islands

Academic Year 2023-24

Keywords:

Collision avoidance, Multi-Robot, Autonomus navigation, RVO, ORCA, Artificial Potential Fields, Stop-and-wait, AUV, ROS

Thesis Supervisor's Name: Antoni Martorell Torres

Thesis Supervisor's Name: José Guerrero Sastre

Implementation of Obstacle Avoidance Algorithms for Underwater Multi-Robot Systems

Antoni Raja Mateos
Thesis Supervisors: Antoni Martorell Torres, José Guerrero Sastre
Master's Thesis in Intelligent Systems (MUSI)
University of the Balearic Islands
07122 Palma, Balearic Islands, Spain
antoni.raja1@estudiant.uib.cat

***Abstract*—Navigation of multi-robot systems (MRS) in underwater environments can be challenging due to environmental conditions that can compromise the integrity of autonomous underwater vehicles (AUVs) as a result of possible collisions between them. For this reason, this Master's Thesis focuses on the design of a MRS that implements two obstacle avoidance algorithms, with the aim of carrying out numerous 3-D simulations, using the Robot Operating System (ROS) and the Sparus II AUV architecture, in order to obtain results with which to compare the efficiency of both algorithms. Specifically, the efficiency of the classical Artificial Potential Fields (APF) algorithm is compared with the Optimal Reciprocal Collision Avoidance (ORCA) algorithm. Furthermore, the performance of these algorithms when used in combination with a stop-and-wait (SAW) strategy is evaluated. The results obtained from the simulations, indicate that, in general, the ORCA algorithm outperforms the APF algorithm in terms of obstacle avoidance efficiency. Furthermore, the work concludes that the SAW strategy is less efficient in terms of navigation time but demonstrates greater robustness in terms of collision avoidance when applied, and tends to yield better results when combined with the APF algorithm. However, conducting numerous simulations is essential to determine the most appropriate navigation strategy based on the environmental characteristics.**

***Index Terms*—Collision avoidance, Multi-Robot, Autonomus navigation, RVO, ORCA, Artificial Potential Fields, Stop-and-wait, AUV, ROS.**

I. INTRODUCTION

In a world marked by constant technological evolution, emerging technologies have propelled the field of autonomous mobile robotics, revolutionizing our capacity to explore vast and unknown environments, such as the deep sea. These advancements have made mobile robotics play a vital role in remote exploration, having a crucial impact on emergency response efforts during natural and human-induced disasters. Furthermore, this technology exhibits tremendous potential, extending its applications to many critical sectors such as underwater mining, underwater infrastructure maintenance or even military operations [1]. Therefore, deploying multiple autonomous robots such as autonomous underwater vehicles (AUVs) allows to explore these environments in a more robust and flexible manner. From now on, these systems composed of multiple AUVs will be referred to as multi-robot systems (MRS). However, the deployment of MRS implies a potential risk of collision among AUVs.

The capacity to navigate and avoid obstacles in terrestrial robots relies on external sensors such as infrared sensors, ultrasonic sensors, and GPS (Global Positioning System), which provide vital information about the robot's surroundings [19]. However, the inherent characteristics of the sea, such as pressure, signal attenuation and lack of visibility, make autonomous navigation in underwater environments more difficult. In the case of underwater MRS, collision avoidance requires specific mechanisms. This implies the need for effective communication methods to share real-time information between AUVs, such as position and velocity, and to coordinate their actions appropriately.

In particular, seawater prevents the efficient transmission of electromagnetic signals in terms of bandwidth and data rate for long distances, making radio frequency-based communication, commonly used in terrestrial environments, unsuitable for such scenarios in this context. In response, a very common alternative is to use acoustic [24] or optical [25] communication methods. In underwater environments, optical communication provides higher data transmission rates and bandwidth, compared to acoustic signals, but it faces greater attenuation. Thus, acoustic communication methods are a preferable choice for long-distance communications with a small volume of data. [16].

Consequently, the main objective of this Master's Thesis is to implement and assess the effectiveness of the classic Artificial Potential Fields (APF) algorithm with the Optimal Reciprocal Collision Avoidance (ORCA) algorithm, while also comparing their efficacy when combined with a stop-and-wait (SAW) strategy. These algorithms serve as mechanisms for collision avoidance, although it's essential to note that alternative methods may be considered in specific situations. For this purpose, simulations will be carried out with a MRS, where communications are based on acoustic signals. This MRS will be based on Robot Operating System (ROS) [23], which is an open-source framework that provides a wide range of tools and libraries for building and managing complex robotic systems, facilitating communication between the various components of the robot,

hardware management, simulation and implementation of control algorithms. Furthermore, each simulated AUV will use the Sparus II architecture (shown in Figure 1), which is an easily deployable AUV that combines its torpedo-shape performance with hovering capability, endowing it with high maneuverability to explore underwater environments.

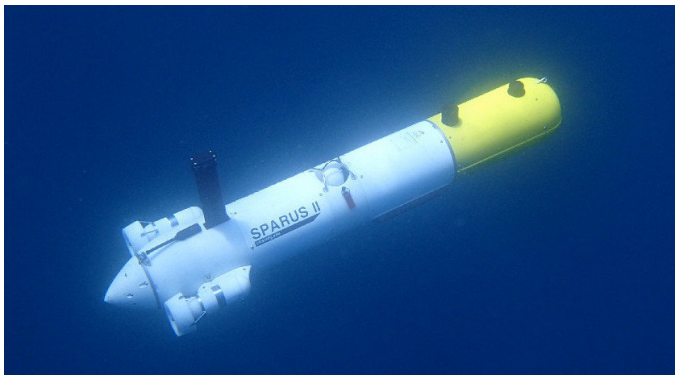


Figure 1: AUV Sparus II manufactured by IQUA Robotics [7].

Additionally, to understand how these AUVs would communicate in a real environment, an Autonomous Surface Vehicle (ASV) that facilitates acoustic communications to provide access to odometry information for all AUVs will be presented in Section V-A. However, it is important to clarify that it has not been implemented in this work, as it is not required to include it in the simulations.

Therefore, the main objectives of this Master's Thesis are:

1) Design a scalable MRS for two or more AUVs based on Robot Operating Systems (ROS).
2) Implement the APF and ORCA algorithms, along with the SAW strategy.
3) Test the algorithms under simulations and extract the optimal parameter combination for each algorithm.
4) Analyze and compare the results obtained from simulations using the APF and ORCA algorithms individually, as well as in combination with the SAW strategy, for different scenarios.

This paper is structured as follows: Section II provides a review of related work in the existing literature related to the field of collision avoidance for autonomous navigation. Section III describes the mathematical concepts applied by each algorithm, while Section IV details the implementation of the collision avoidance algorithms and the SAW strategy, and Section V summarizes the simulation environment employed to implement the collision avoidance algorithms. In Section VI, the results obtained after carrying out the simulations with each algorithm and strategy are presented. Section VII covers a discussion about the results obtained, and Section VIII contains the conclusions extracted from this work.

II. Related Work

In the field of autonomous navigation, there are several approaches and algorithms designed to avoid obstacles and to guide robots safely to their destination [8]. Some of these approaches include map-based planning, which allows the robot to calculate an obstacle-free path to reach its destination with methods such as A* [22], Rapidly-exploring Random Trees [15], and even Reinforcement Learning based strategies [28].

However, when it comes to applying these algorithms in large-scale underwater environments, significant challenges arise. These challenges include the need to process large amounts of data in real time, which requires substantial processing capacity, energy power and reliable communication. In the absence of an adequate computational infrastructure, the time required to plan a safe route can become excessively long. Furthermore, not all of these algorithms are equally efficient in changing and dynamic environments, where several vehicles navigate in close proximity to each other. To prevent this, strategies based exclusively on obstacle avoidance have been chosen as the focus. As previously stated, the chosen algorithms are the classical Artificial Potential Fields (APF) algorithm and the Optimal Reciprocal Collision Avoidance (ORCA) algorithm.

The current literature widely covers the implementation of several variants of the APF algorithms for autonomous navigation in 3-D environments [26, 31]. However, despite their successful application in MRS, a discernible trend emerges in which the consideration of potential collisions between AUVs is often overlooked [5, 30]. On the other hand, identifying similar applications with the ORCA algorithm is a complicated task, as a significant proportion of the implementations have been predominantly carried out in different environments [9, 29]. Furthermore, the current literature also addresses the use of diverse simulation environments focused on AUV control for autonomous navigation in underwater environments [18, 21]. However, it is important to note that these studies do not explicitly implement obstacle avoidance algorithms such as APF and ORCA. This accentuates the pressing need for research efforts that comprehensively address the dynamics associated with collisions between AUVs in the specific context of underwater environments.

III. Obstacle Avoidance Algorithms

The following sections will provide a general description of the obstacle avoidance algorithms that have been implemented.

A. Artificial Potential Fields (APF)

The Artificial Potential Fields (APF) algorithm, first introduced by Oussama Khatib in 1986 [14], is an easy-to-implement algorithm which consists of generating a vector representing the direction and intensity that the robot must take in order to avoid obstacles while moving towards the target. For this purpose, this vector is obtained by adding the attractive force, which points towards the robot's goal, and the sum of the repulsion forces generated by each detected obstacle. This is represented in Equation (1), where x represents the position

of the robot, n the number of detected obstacles and $\vec{F_{goal}}$ and $\vec{F_{obs}}$, the attractive and repulsion force vectors, respectively.

$$\vec{F_f} = \vec{F_{goal}}(x) + \sum_{obs=1}^{n} \vec{F_{obs}}(x) \quad (1)$$

As mentioned previously, there exist different implementations of the APF algorithm. In this work, the repulsion forces will be generated by other AUVs and the magnitude of the attractive force will be constant. Furthermore these vectors will be calculated as position vectors. This is explained in more detail in Section IV-B.

Nonetheless, it is worth mentioning that this algorithm is not a complete algorithm. This means it does not guarantee to find a solution in every situation, since the attractive and repulsion forces may have completely opposite directions, causing the robots to move back and forth indefinitely and encountering a trapping situation that they are unable to overcome. This also applies to those situations where the goal is close to an obstacle. Furthermore, this algorithm is not intended for MRS. To understand this, Figure 2a illustrates an ideal case and Figure 2b illustrates a case in which AUVs have entered in a trapping situation.

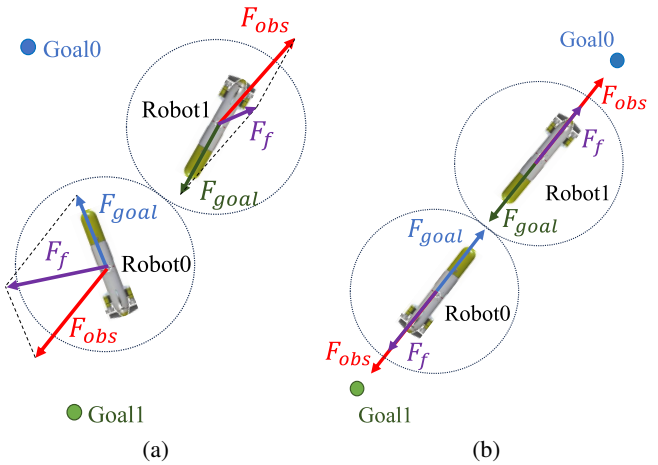


Figure 2: Implementation of the APF algorithm when AUVs detect another AUV as an obstacle. Where (a) represents an ideal case and (b) represents a trapping situation case.

B. Optimal Reciprocal Collision Avoidance (ORCA)

The Optimal Reciprocal Collision Avoidance (ORCA) algorithm is a geometric guidance strategy intended for MRS, which was first introduced in 2011 [3], and which calculates safe and evasive velocities of multiple mobile robots in real time.

The ORCA algorithm was introduced as a formal approach to the Reciprocal Velocity Obstacles (RVO) algorithm (first introduced in 2008 [4]), which is an enhancement of the Velocity Obstacles (VO) algorithm first introduced in 1998 [10]. The RVO algorithm applies reciprocity related to obstacle detection. This implies that a robot A will perceive robot B as an obstacle, and vice versa. Based on this, each robot will compute a potential collision area for each neighboring robot, which represents a region of space in which the main robot may have a collision with a neighboring robot. This allows to find various velocities that they can adopt to remain outside of these collision areas. However, it is necessary to obtain the $VO^A_B(v_B)$ region, which contains all possible velocities of robot A that will cause a collision with obstacle B, without applying reciprocity. This is shown in Equation (2).

$$VO^A_B(v_B) = \{v_A | \lambda(p_A, v_A - v_B) \cap B \oplus -A \neq \emptyset\} \quad (2)$$

Where v_A and v_B represent the velocities of robots A and B, respectively. While λ represents a ray starting from the current position p_A of the robot A in the direction of $v_A - v_B$, and which result with the intersection of the space resulting from the Minkowski sum between the space occupied by robot B and the reflection of the space occupied by robot A ($-A$) must provide a non-empty set. Being the Minkowski sum a classical operation used to obtain the sum of two sets of position vectors [2]. Thus, the result of this operation must provide a value of v_A that poses a potential risk of collision [27]. The λ expression is shown in Equation (3). Where p represents the initial position of the ray, v the direction vector and t the time at which it is calculated.

$$\lambda(p, v) = \{p + tv | t \geq 0\} \quad (3)$$

Consequently, the Equation (4) shows how the $RVO^A_B(v_B, v_A)$ region is obtained, which contains all velocities v'_A for robot A that are the average of the velocity v_A and the velocity v_B within $VO^A_B(v_B)$. This velocity v'_A would represent the set of velocities of robot A that will cause a collision with robot B in the $RVO^A_B(v_B, v_A)$ region.

$$RVO^A_B(v_B, v_A) = \{v'_A | 2v'_A - v_A \in VO^A_B(v_B)\} \quad (4)$$

Therefore, this implies that, if robot A follows a velocity within $RVO^A_B(v_B, v_A)$, then it will collide with robot B at some moment in time, and if the velocity is outside $RVO^A_B(v_B, v_A)$ then they will never collide. But if the velocity is on the boundary of $RVO^A_B(v_B, v_A)$ robots will touch, meaning it does not necessarily imply a collision [4].

The $RVO^A_B(v_B, v_A)$ region can be visualized as a translation of $VO^A_B(v_B)$, which apex is defined by the velocity v_B. Therefore, it can be obtained that the apex of $RVO^A_B(v_B, v_A)$ is at the midpoint between the velocities v_A and v_B, meaning by this that its position can be expressed as $\frac{v_A + v_B}{2}$. Figure 3 shows the interpretation of $RVO^A_B(v_B, v_A)$ from a geometrical point of view, where r_A and r_B are the radius that define the space occupied by the robots A and B, respectively.

However, as stated previously, in 2011 a formal approach to the RVO algorithm called ORCA was introduced, which assumes that each robot can obtain the distance and relative

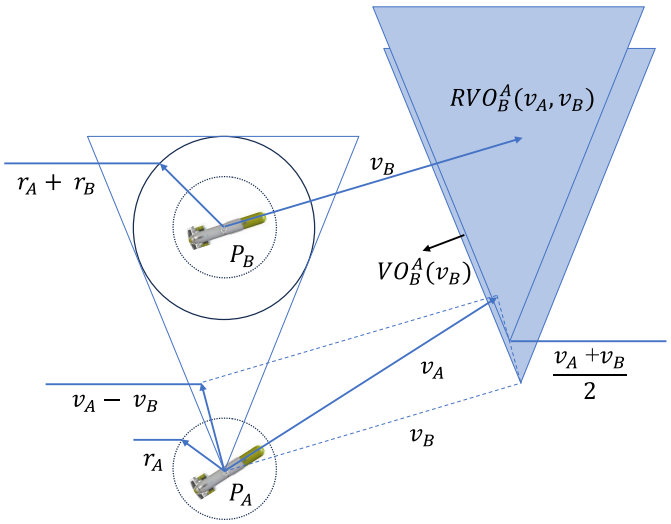


Figure 3: Obtaining the $RVO_B^A(v_B, v_A)$ region from the $VO_B^A(v_B)$ region [11].

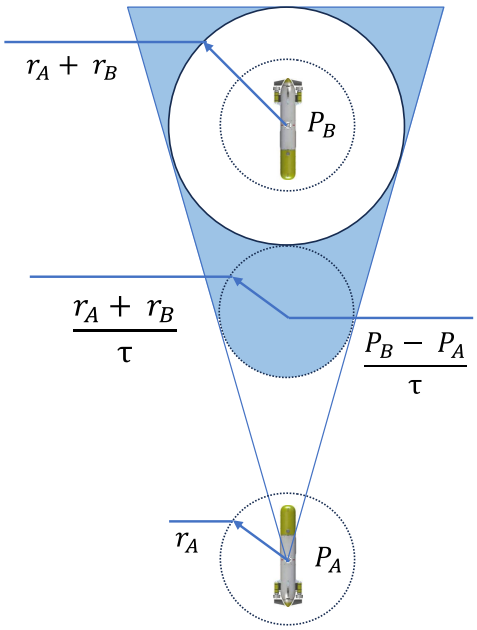


Figure 4: Representation of the $VO^{\tau}_{A|B}$ region in blue [11].

velocities with respect to the neighboring robot.

The velocities that can be taken by the neighbors and that can cause a collision within a certain time horizon τ are calculated. $VO^{\tau}_{A|B}$ contains this set of velocities, and it is calculated as shown in Equation (5) [11, 12].

$$VO^{\tau}_{A|B} = \{v | \exists t \in [0, \tau] :: vt \in D(p_B - p_A, r_A + r_B)\} \quad (5)$$

This set of velocities are the relative velocities of robot A with respect to robot B, and they are computed at a time instant t within the interval $[0, \tau]$. From these velocities, the relative position is derived at the specific time t. The obtained relative position must fall within the region $D(p_B - p_A, r_A + r_B)$. This condition implies that robot A could enter this region at some point in time, suggesting a potential collision with robot B. The position of the $D(p_B - p_A, r_A + r_B)$ region is obtained applying the expression in Equation (6).

$$D(p, r) = \{q \,|\, ||q - p|| < r\} \quad (6)$$

Where q is the set of positions of $D(p_B - p_A, r_A + r_B)$ which distance to the vector $p_B - p_A$ is less than the sum of the robot radius $(r_A + r_B)$.

The region $VO^{\tau}_{A|B}$ is represented in blue in Figure 4. Where the circumference of radius $r_A + r_B$ represents the region $D(p_B - p_A, r_A + r_B)$, with robot A being positioned at the coordinate origin.

Nonetheless, the goal of ORCA is to select a new velocity v_A^{new} that most closely matches the robot's preferred velocity v_A^{pref}, which is the velocity that it would choose without taking into account any obstacle. Therefore, it is necessary to obtain the $ORCA^{\tau}_{A|B}$ half-planes, applying the expression in Equation (7).

$$ORCA^{\tau}_{A|B} = \begin{cases} \{v | (v - (v_A + \frac{1}{2}u))n \geq 0\} & \text{if}(v_A - v_B) \notin VO^{\tau}_{A|B} \\ \{v | (v - (v_A^{opt} + \frac{1}{2}u))n \leq 0\} & \text{if}(v_A - v_B) \in VO^{\tau}_{A|B} \end{cases} \quad (7)$$

In this Equation (7), if the vector $v_A - v_B$ is not within $VO^{\tau}_{A|B}$, then robot A will follow any velocity which projection $v - (v_A + \frac{1}{2}u)$ in direction n is positive, causing robot A to move away from robot B. Where n is the normal of u. But if the relative velocity is within $VO^{\tau}_{A|B}$, then robot A will follow the velocity which projection $v - (v_A^{opt} + \frac{1}{2}u)$ in direction n is not positive, allowing the robot A to follow a velocity that closely matches v_A^{opt}, which represents the optimization velocity. In [3] it is mentioned that the value of v_A^{opt} will depend on the applications and the environment.

In these projections, u is the vector with the minimum length from $(v_A - v_B)$ to the boundary of the $VO^{\tau}_{A|B}$ region that represents the smallest velocity change so that the vector $v_A - v_B$ is out of $VO^{\tau}_{A|B}$, and which is calculated as shown in Equation (8).

$$u = (argmin_{v \in \partial VO^{\tau}_{A|B}} ||v - (v_A - v_B)||) - (v_A - v_B) \quad (8)$$

In the previous Equation (7), the weight of this vector u is reduced by half, since it is assumed that the two robots involved in the collision avoidance will implement this strategy.

Figure 5 represents how the $ORCA^{\tau}_{A|B}$ half-planes are obtained, in case the vector $v_A - v_B$ is within $VO^{\tau}_{A|B}$.

Once the half-planes $ORCA^{\tau}_{A|B}$ have been obtained it is possible to obtain the region $ORCA^{\tau}_{A}$ as well, which contains

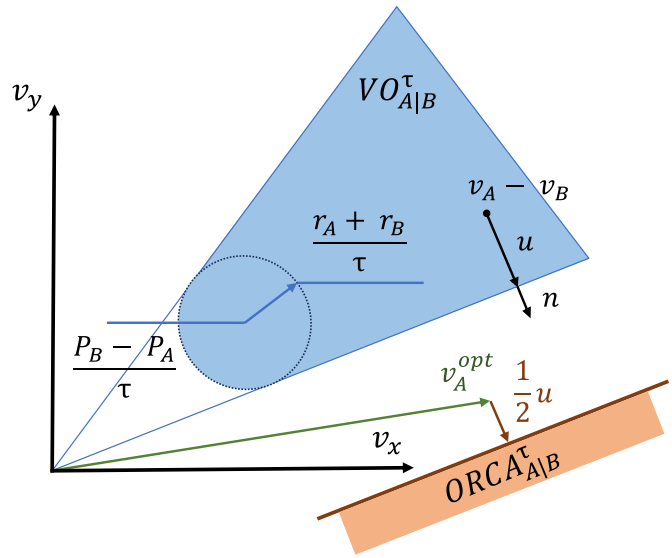


Figure 5: Obtaining $ORCA^{\tau}_{A|B}$ when $v_A - v_B$ is within $VO^{\tau}_{A|B}$. [3, 12].

the set of safe velocities considering all neighboring robots, as shown in Equation (9).

$$ORCA^{\tau}_{A} = D(0, v_A^{max}) \cap \left(\bigcap_{B \neq A} ORCA^{\tau}_{A|B} \right) \quad (9)$$

As can be observed in this Equation (9), the $ORCA^{\tau}_{A}$ is obtained as the result from the intersection between the region $D(0, v_A^{max})$, located at the coordinate origin with a radius equal to the maximum velocity v_A^{max} of robot A, and the intersection of the half-planes $ORCA^{\tau}_{A|B}$ for all neighboring robots.

And finally, the new velocity that the robot must take is the closest one to v_A^{pref} within $ORCA^{\tau}_{A}$, as expressed in Equation (10).

$$v_A^{new} = argmin_{v \in ORCA^{\tau}_{A}} ||v - v_A^{pref}|| \quad (10)$$

IV. Collision Avoidance Algorithms

This section presents the decisions taken for the code implementation of the obstacle avoidance algorithms and the SAW strategy[1].

A. Simulation Parameters

To delimit the collision avoidance zone, it has been decided to define a parameter called CRIT_DIST, which value represents the maximum obstacle detection distance. In addition, the parameter safety_radius has been introduced, which delimits a safety sphere around the AUV, and with which, if another safety sphere of the AUV is detected to have come into contact, a collision is considered to have occurred.

The Sparus II AUV can move horizontally and vertically, in addition to being able to rotate around the Z-axis [7], allowing it to move in 3-D environments. Therefore, when assigning a new velocity to the AUV, it is necessary to calculate the rotation angle α needed to orient the AUV in the direction to the projection of the new velocity vector v_{new} on the XY plane. For this purpose, the parameter ORI_ERROR has been introduced, which represents the maximum allowed difference between the current position of the AUV and the new orientation provided by v_{new}, so that the AUV can proceed to move. Figure 6 illustrates these parameters.

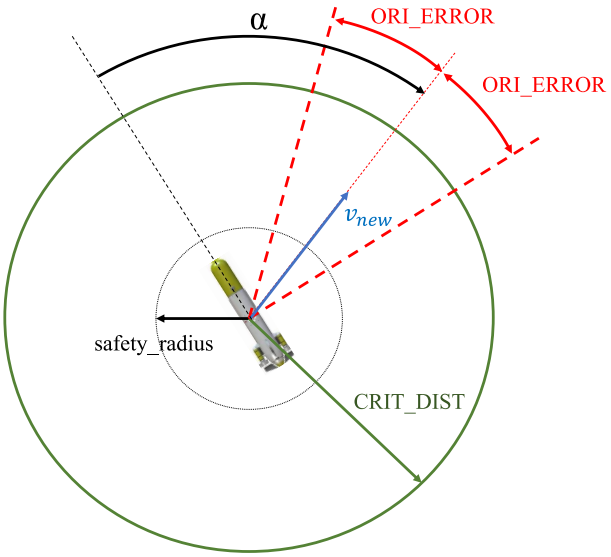


Figure 6: Representation of the AUV parameters

However, more parameters related to the coordination and navigation of the AUVs must be defined. Table I presents a description and the default value of all simulation parameters, along with the rest of the parameters introduced in the implementation of the obstacle avoidance algorithms and the SAW strategy. Note that some values are actually ranges of values, meaning that they will be modified before simulating, during the parameter setting phase presented in Section VI-B, as they are considered to be the most relevant of all. Regarding the use of the time_coefficient parameter, it will be explained in Section V-B.

B. APF Algorithm Implementation

The implementation of the APF algorithm is presented in Algorithm 1. As can be observed, in lines 5 to 9, if an obstacle is located at a distance obs_dist from the AUV, equal or smaller than CRIT_DIST, Vi is calculated, which is a magnitude that increases the closer the AUV is to an obstacle. Therefore, in line 10 the direction of the vector $\vec{V_i}$, which represents the repulsion force for a specific obstacle, is obtained from the position of the AUV AUV_pos, the obstacle position obs_pos and the distance to the obstacle

[1] https://github.com/ToniRaja/collision_avoidance

System Parameters		
Parameter	**Description**	**Value**
robot_name	Name of the AUV which follows the nomenclature pattern "robotX", where X represents the robot identifier.	"robotX"
num_robots	Number of AUVs that make up the MRS.	Adjustable
ALGOR	Indicates the collision avoidance algorithm to be applied (1 for the APF algoritm and 2 for the ORCA algorithm).	1 or 2
safety_radius	Safety distance at which, if an obstacle is detected, a collision is considered to have occurred.	3 meters
CRIT_DIST	Maximum obstacle detection distance.	6-20 meters
ORI_ERROR	Maximum error in the AUV orientation to adopt the new velocity v_{new}.	0.4-0.8
DES_VEL	AUV travel speed.	0.5 m/s
ROT_VEL	AUV rotation speed.	0.45 m/s
K_ROT_MIN	Rotation coefficient used when the AUV is moving.	0.15
K_ROT_MAX	Rotation coefficient used when the AUV is not moving.	0.3
time_coefficient	Coefficient used to obtain the navigation time limit.	10
APF Parameters		
Parameter	**Description**	**Value**
W1	Coefficient of the attractive force of the APF algorithm.	1.0
W2	Coefficient of the repulsion force of the APF algorithm.	0.5-500.0
ORCA Parameters		
Parameter	**Description**	**Value**
time_step	Discretization time used to predict the movements of the neighboring robots.	0.1-1.0 seconds
time_horizon	Contemplated time horizon until which the prediction of the movements of the neighboring robots is executed.	5.0-20.0 seconds
SAW Parameters		
Parameter	**Description**	**Value**
SAW_PRIO	Represents the priority of the AUV, employed to know if it is allowed to move when obstacles are detected.	Robot identifier

Table I: Parameters, descriptions and values.

obs_dist, and the intensity is defined by the Vi magnitude. Subsequently, in line 11, the vector $\vec{V_{obs}}$, is calculated as a sum of all $\vec{V_i}$ for each obstacle, thus obtaining the total repulsion force.

On the other hand, in line 12 the attractive force has been calculated as a unit vector $\vec{V_{obj}}$, obtained from the position of the goal obj_pos, the current position of the AUV AUV_pos and the distance to the objective obj_dist.

Therefore, in line 13 the $\vec{V_f}$ vector, which represents the direction that the AUV must follow to avoid obstacles and go towards its objective, is computed based on the vectors $\vec{V_{obj}}$ and $\vec{V_{obs}}$, and the parameters W1 and W2, which represent the attractive and repulsion coefficients, respectively.

Subsequently, based on the vector $\vec{V_f}$ and the angle ψ that represents the orientation of the AUV with respect to the global coordinate axis, the smallest angle α that the AUV must rotate to be oriented with the vector V_f is obtained, as shown in lines 14 and 15. Based on α, it is checked if the vector $\vec{V_f}$ points behind the AUV. If this is the case and also obstacles are detected within the CRIT_DIST range, α is recalculated so that the AUV can go backwards. Thus, AUVs will not waste time rotating and the navigation time will be shorter. Otherwise they will move forward. This is calculated in lines 16 to 26, where Vf_behind indicates if the vector $\vec{V_f}$ points backwards.

As can be observed in lines 27 to 41, based on the value of α and the direction of $\vec{V_f}$, a velocity V_x is assigned to the linear velocity of the AUV. If α is smaller than ORI_ERROR it means that the AUV has rotated enough, so the AUV must travel at a constant linear velocity DES_VEL. In case alpha is inside ORI_ERROR, the values of K_ROT_MIN, ROT_VEL and α are used to define the angular velocity W_z, resembling a proportional controller. However, in case alpha is outside ORI_ERROR, then V_x is 0, since the AUV must wait to be oriented correctly, and the W_z value will be defined by K_ROT_MAX, instead of K_ROT_MIN which is used for safer navigation.

C. ORCA Algorithm Implementation

The RVO2-3D library[2], published by Berg et al. [3], has been used to implemenent the 3-D version of the ORCA algorithm according to Section III-B. This library allows predicting the movements of neighboring AUVs within a time interval, returning a vector of velocities that the main AUV must adopt to avoid colliding with them while navigating towards its goal.

Some values required to implement the aforementioned functions are the size of the robots, the obstacle detection range, the maximum number of robots and their maximum speed. Therefore, the parameters safety_radius, CRIT_DIST, num_robots and DES_VEL will be used respectively to define this specifications. It is also required to specify the parameters time_step, which represents the discretization time used to predict the movements of neighboring AUVs, and the parameter time_horizon that represents the maximum time into the future until which the prediction is performed. See the RVO2-3D web page for a more detailed description[3]. These parameters are described in the Table I as well. Additionally, it is important to note that in these libraries the optimization velocity v_{opt} described in Section III-B is equivalent to the preferred velocity v_{pref}.

To implement this algorithm in a 3-D environment, the functions encapsulated within the RVOSimulator

[2]https://gamma.cs.unc.edu/RVO2/

[3]http://gamma-web.iacs.umd.edu/RVO2/documentation/3d-1.0/params.html

Algorithm 1 APF algorithm pseudocode

1: $\vec{V_{obs}} \leftarrow 0$
2: $\vec{V_{obj}} \leftarrow 0$
3: $Vf\_behind \leftarrow False$
4: **for** each detected obstacle **do**
5: **if** $obs\_dist \leq CRIT\_DIST$ **then**
6: $V_i \leftarrow \frac{CRIT\_DIST - obs\_dist}{CRIT\_DIST}$
7: **else**
8: $V_i \leftarrow 0$
9: **end if**
10: $\vec{V_i} \leftarrow V_i \frac{AUV\_pos - obs\_pos}{obs\_dist}$
11: $\vec{V_{obs}} \leftarrow \vec{V_{obs}} + \vec{V_i}$
12: $\vec{V_{obj}} \leftarrow \frac{obj\_pos - AUV\_pos}{obj\_dist}$
13: $\vec{V_f} \leftarrow W1 * \vec{V_{obj}} + W2 * \vec{V_{obs}}$
14: $\alpha \leftarrow atan2(V_{fy}, V_{fx})$
15: $\alpha \leftarrow atan2(sin(\alpha - \psi), cos(\alpha - \psi))$
16: **if** there are obstacles within CRIT_DIST range **then**
17: **if** $\alpha \leq -\frac{\pi}{2}$ **then**
18: $\alpha \leftarrow \alpha + \pi$
19: $Vf\_behind \leftarrow True$
20: **else if** $\alpha \geq \frac{\pi}{2}$ **then**
21: $\alpha \leftarrow \alpha - \pi$
22: $Vf\_behind \leftarrow True$
23: **end if**
24: **else**
25: $Vf\_behind \leftarrow False$
26: **end if**
27: **if** $\alpha \leq ORI\_ERROR$ and Vf_behind = True **then**
28: $V_x \leftarrow -DES\_VEL$
29: $W_z \leftarrow K\_ROT\_MIN * ROT\_VEL * \alpha$
30: **else if** $\alpha \leq ORI\_ERROR$ and Vf_behind = False **then**
31: $V_x \leftarrow DES\_VEL$
32: $W_z \leftarrow K\_ROT\_MIN * ROT\_VEL * \alpha$
33: **else**
34: $V_x \leftarrow 0$
35: $W_z \leftarrow K\_ROT\_MAX * ROT\_VEL * \alpha$
36: **end if**
37: **if** V_{fz} < 0 **then**
38: $V_z \leftarrow -DES\_VEL$
39: **else**
40: $V_z \leftarrow DES\_VEL$
41: **end if**
42: **end for**

class defined in the RVO2-3D library are employed. The $addAgent()$ function is utilized to facilitate the prediction of neighboring AUV movements by adding AUVs to the prediction. Subsequently, the functions $setAgentPosition()$, $setAgentVelocity()$, and $setAgentPrefVelocity()$ must be employed to constantly update the positions, velocities, and preferred velocities of the AUVs, respectively. The computation of the new velocity vector is facilitated by the $doStep()$ function, while the $getAgentVelocity()$ function is utilized to retrieve the new velocity vector that the main AUV must adopt [4]. Consequently, the value assigned to the linear velocity V_x of the AUV corresponds to the modulus of the X and Y components of the new velocity vector, and the V_z value is equivalent to the Z component of the vector.

The logic applied to rotate the AUV towards the projection on the XY plane of the new velocity vector is identical to the logic used in the APF algorithm, resembling a proportional controller, as can be observed in lines 27 to 36 of Algorithm 1. Furthermore, the navigation logic for moving forward or backward, is exactly the same as in the APF algorithm implementation as well, observable in lines 16 to 41.

D. SAW Strategy Implementation

The implementation of the stop-and-wait strategy is simple. Each AUV has a navigation priority defined by the SAW_PRIO parameter. At the start of the simulation, each AUV transmits its SAW_PRIO value to ensure that the rest of AUVs are aware of its priority. Therefore, when the outer spheres of two AUVs, defined by the CRIT_DIST parameter, come into contact, both AUVs compare their priorities. In the case of the AUV with the lower priority, it will stop moving, waiting for the other AUV to move far enough away, while the AUV with the higher priority will apply the selected collision avoidance algorithm, based on the value of the ALGOR parameter of Table I.

The value of the SAW_PRIO parameter is determined by the robot_name parameter, following the nomenclature "robotX," where "X" is an integer representing the AUV identifier, which value is unique, causing each AUV to have a different priority. Subsequently, the rule is that a lower identifier value corresponds to a higher navigation priority. This implies that the AUV with the name "robot0" will always have the highest priority to continue navigating, so it will never stop navigating when detecting other AUVs as obstacles.

V. Simulation Setup

This section presents and describes the simulation setup employed to carry out this work and how it has been used to obtain the results.

A. Sparus II hardware and control architecture

As stated previously, it has been decided to use the AUV Sparus II architecture. This AUV manufactured by IQUA Robotics has a torpedo shape hull of 1.6 meters in length and 0.23 meters in hull diameter, capable of diving to a maximum depth of 200 meters and reaching a maximum speed of 3 knots (equivalent to 1.53 meters per second), also equipped with a 1.9 kWh lithium-ion (Li-ion) battery [6]. This AUV also has two horizontal thrusters and a vertical one, to control its surge, heave and yaw degrees of freedom (DOFs) [7], allowing the AUV to navigate in 3-D environments, as stated

[4] http://gamma-web.iacs.umd.edu/RVO2/documentation/3d-1.0/class_r_v_o_1_1_r_v_o_simulator.html

in Section IV-A.

In terms of software architecture, this AUV uses the COLA2 (Component-Oriented Layer-Based Control Architecture) architecture [20] to control it, which is a three-layer based architecture designed to facilitate autonomous and cooperative operations for underwater exploration and mapping missions. One of the layers composing COLA2 is the Reactive Layer, which allows the AUV to precisely determine its position and orientation in space, using components such as pressure sensors, IMU (Inertial Measurement Unit), GPS (Global Positioning System) and DVL (Doppler Velocity Log). Additionally, this layer facilitates immediate responses by controlling the AUV thrusters. COLA2 also includes the Mission Layer, which is responsible for high-level mission planning, goal definition, and coordination of activities. On the other hand, the Execution Layer acts as an interface between the other two layers, to ensure precise maneuvering and responsiveness to achieve the objectives set by the Mission Layer.

Furthermore, in [7], the COLA2 architecture has been implemented with the Sparus II AUV as a software architecture based on Robot Operating System (ROS), which is an open-source framework that facilitates the development and operation of robotic systems by providing a decentralized architecture based on nodes that exchange information through topics. These topics are unidirectional message channels that allow nodes to publish data and subscribe to receive it. In addition to topics, ROS uses services to enable synchronous communication between nodes, offering a comprehensive set of tools for robot programming. Therefore, this COLA2 software architecture has been implemented in the simulation setup employed in this work, since this allows to perform simulations with several Sparus II units and thus simulate a MRS, with which the collision avoidance algorithms and the SAW strategy can be tested.

Additionally, the COLA2 architecture also implements 4 different relevant navigation strategies as services:

1) Goto strategy: it allows an AUV to travel to a target at a specific speed. Once the AUV is at a close distance to the goal, within a tolerance range, the AUV is considered to have reached its destination.
2) Keep position: it allows the robot to remain in proximity to a specific position, staying within a predefined tolerance range around that position.
3) Section: it allows following the line between two waypoints and implements a velocity adjustment that depends on the distance of the vehicle to the initial and final waypoints, causing to slow down at the initial and final points and accelerating in the middle.
4) Missions: This strategy simply combines the goto strategy with the section strategy to be able to generate more complex navigation trajectories.

To facilitate the coordination of movements among the AUVs within the MRS, it is essential to possess global knowledge regarding the position and orientation of each AUV. The ASV Xiroi II, developed by the Systems Robotics and Vision group (SRV)[17], serves this purpose by establishing communication with the AUVs through an Acoustic Communication Link (ACL). Thus, AUVs can access to this information through the acoustic channel. This communication is effective within a maximum range of 80 meters, operating at a frequency of around 0.1 Hz. Figure 7 illustrates the communication scheme between the AUVs and the ASV. Nevertheless, it is important to note that the implementation of this ASV has not been carried out in this work, as it is not required for the simulations, since AUVs communicate through ROS' topics.

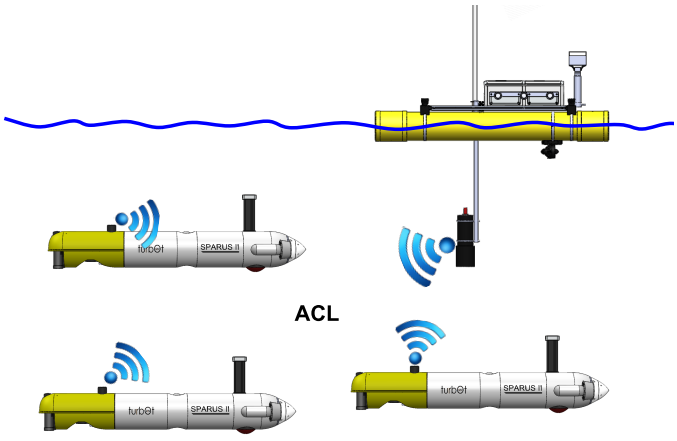


Figure 7: Communication scheme of Sparus II AUVs with Xiroi II ASV [17].

B. AUVs Deployment

To assess the efficacy of collision avoidance algorithms and the SAW strategy, it is essential to deploy multiple AUVs in a manner that intentionally induces collisions between them. This section describes the considerations and decisions made to execute this testing scenario.

The collision avoidance algorithms must be functional in a 3-D environment. For this purpose, the ROS tool RViz [13] has been used to test their correct operation. This tool allows exploring 3-D environments and simulating fully configurable robot models. In this way, thanks to RViz, a 3-D model of the Sparus II can be simulated. The Sparus II 3-D model in RViz is shown in Figure 8.

The main idea is to make the AUVs travel from an initial position to a final position in such a way as to force a collision. These positions can be specified in the simulation system itself so the AUVs can follow navigation paths composed of several waypoints. Specifically, in this work, two types of scenarios have been considered, depending on the number of AUVs to be simulated. For this purpose, the AUVs are positioned in such a way that they form a cube which edges are 20 meters long. Therefore, the AUVs will

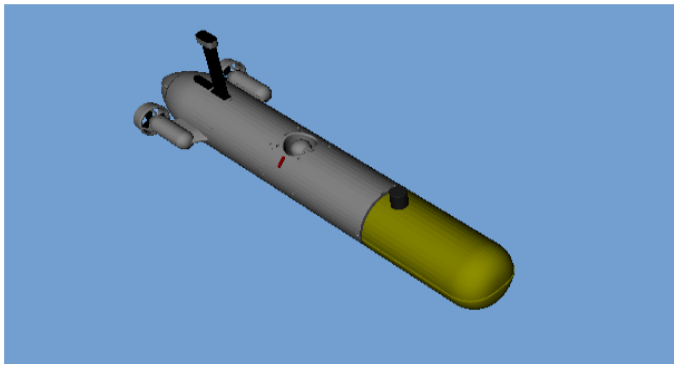


Figure 8: 3-D model of the Sparus II in RViz.

be located at the vertices of the cube. From this position they will travel to the opposite vertex, generating a collision risk situation in the center of the cube. For this same scenario, simulations will be performed with four and eight AUVs.

Figure 9 shows how the four AUVs are located along with the navigation paths to force a collision risk situation in the center of the cube. In this case two of the AUVs will be placed at the opposite lower vertices of the cube and the other two at the opposite upper vertices. Figure 10 illustrates the same situation when eight AUVs are simulated.

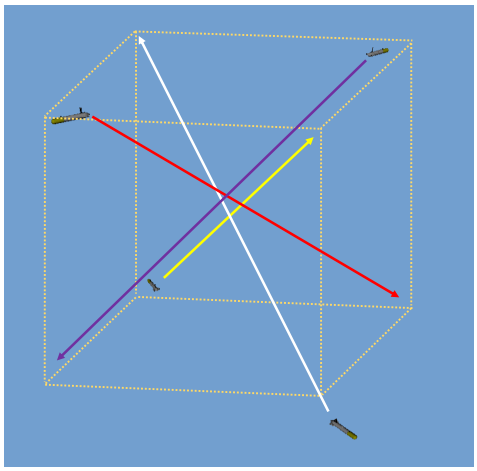


Figure 9: Navigation paths when four AUVs located at the opposite vertices of a 20-meters edge cube.

The simulations that have been carried out in the scenario described above, have also been performed in a larger scenario, in which the AUVs are also positioned forming a cube, which edges are 60 meters long. TableII resumes the different scenarios that will be simulated.

At the beginning of the simulation, the different AUVs that make up the MRS are always located at the global origin of coordinates, called NED (North-East-Down) origin. Therefore, it has been decided that initially all the AUVs will navigate to an initial position (the cube vertices), and once they have arrived, they will maintain their position

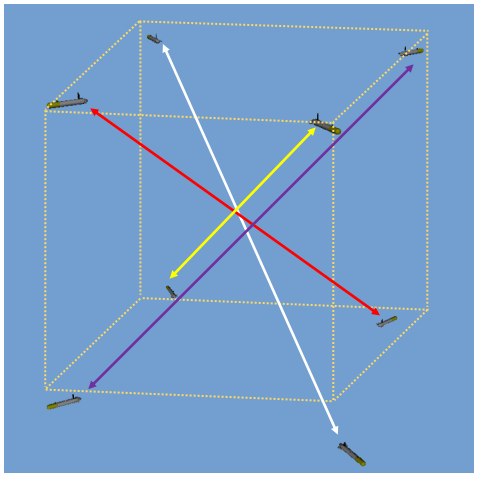


Figure 10: Navigation paths when eight AUVs located at the vertices of a 20-meters edge cube.

Formation Configuration	Number of AUVs
20-meter Edge Cube	4
20-meter Edge Cube	8
60-meter Edge Cube	4
60-meter Edge Cube	8

Table II: Simulation Scenarios with AUVs Arranged at the Vertices of Cubes.

while waiting for the rest of the AUVs to reach their respective initial positions. Therefore, in order to carry out the simulation tests in a correct manner, it is necessary to make use of the navigation strategies presented in Section V-A.

Since this work focuses on the simulation of collision avoidance algorithms, only two out of the four navigation strategies presented will be implemented. The Goto strategy will be used to make the AUVs move to their initial position, although it is not necessary since collision avoidance algorithms are capable of doing so. The only reason why the Goto strategy has been used is to emphasize that the collision avoidance simulation test does not start until each AUV has reached its initial position. For this reason, due to the tolerance range of this navigation strategy, the AUVs will not be positioned exactly at the cube vertices, as can be observed in Figures 9 and 10. Figure 11 shows 4 AUVs using the Goto navigation strategy to reach their initial positions.

On the other hand, the Keep position strategy will be used when an AUV waits for the rest of the AUVs to reach their initial positions and when it detects another AUV with a higher priority when the SAW strategy is applied. Figure 12 shows an AUV waiting (using the keep position navigation strategy) for the rest of the AUVs to reach their initial positions.

As mentioned previously, once all AUVs have reached their initial positions, collision avoidance is activated. Subsequently, they navigate towards the next position, forcing a collision risk situation. From this point on, data is collected

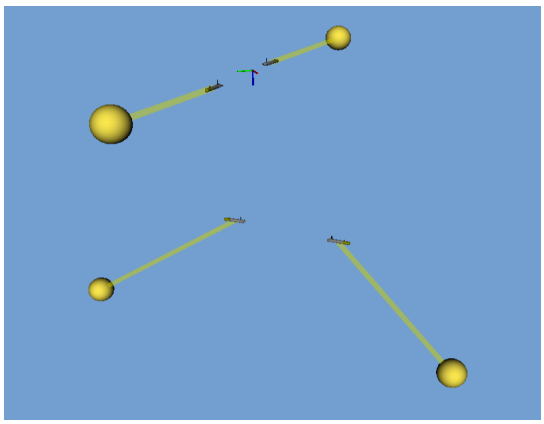


Figure 11: AUVs navigating to their initial positions, represented by the yellow balls.

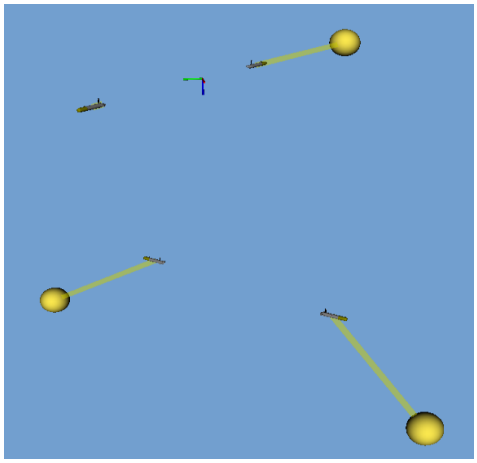


Figure 12: AUV waiting for the rest of AUVs to reach their initial positions.

to facilitate the analysis of the collision avoidance algorithms and the SAW strategy. The collected data is defined as follows:

1) Avoided collisions: Number of times the outer spheres of two AUVs, defined by the CRIT_DIST parameter, overlap and separate without the inner spheres having overlapped, defined by the safety_sphere parameter.
2) Not avoided collisions: Number of times the inner spheres of two AUVs have overlapped.
3) Navigation time: Time elapsed from the time an AUV leaves its initial position until it reaches its final position.
4) Collision avoidance time: The elapsed time in which AUVs are detecting obstacles. Therefore, its value must not be greater than the navigation time.
5) Time limit: The navigation time limit is determined by the time_coefficient parameter and the minimum navigation time, which represents the duration it would take for the AUV to reach its goal in an obstacle-free trajectory. This minimum navigation time is calculated dividing the distance to the goal by the maximum speed DES_VEL. Therefore, the navigation time limit is calculated multiplying the minimum navigation time by the time_coefficient parameter. In case this time limit is exceeded the AUV will be considered to have entered in a trapping situation.
6) Distance travelled: The Distance travelled from the AUV's initial position until the final position is reached.

To comprehend the validation of the implementation of the chosen collision avoidance algorithms and ensure precise data collection, three figures are provided. These figures encapsulate moments within a simulation, depicting a scenario involving the risk of collision. Specifically, Figure 13 shows the moment when collision avoidance is enabled. As can be observed, the safety spheres and the collision avoidance spheres are visualized in green, since they are not detecting any obstacle. On the other hand, Figure 14 represents the moment where two AUVs have detected each other as an obstacle. Therefore, the collision avoidance spheres are represented in orange to indicate so. And finally, Figure 15 shows a case in which the spheres of the AUVs are represented in red, meaning the collision could not be avoided.

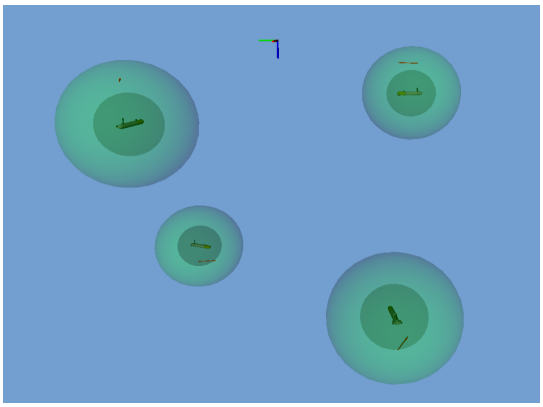


Figure 13: AUVs when they all have reached their initial positions, enabling collision avoidance. The NED origin is also visible.

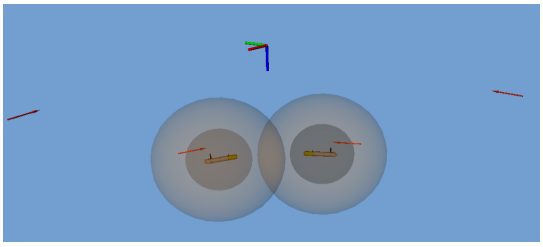


Figure 14: AUVs when they detect each other as an obstacle.

VI. Experimental Results

This Section presents the phases carried out during the simulation process to obtain the necessary data to analyze the efficiency of the navigation algorithms and the SAW strategy. In this context, this Section describes the adjustment of parameters for collision avoidance algorithms and subsequently presents the achieved results derived from their application.

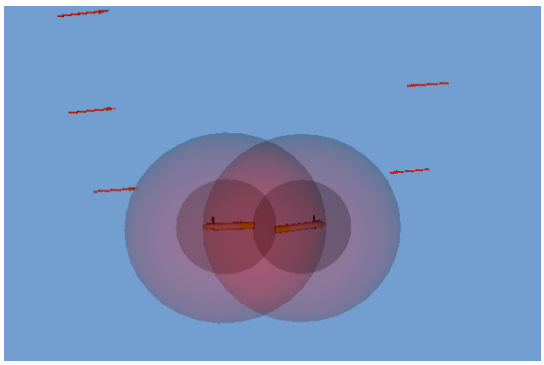


Figure 15: AUVs when the collision has not been avoided.

A. *Parameter setting phase*

The first phase is a parameter setting phase in which only simulations with the obstacle avoidance algorithms will be performed, since the SAW strategy has no parameters that directly condition its efficiency. Therefore, it is necessary to perform numerous simulations with different parameter values, in order to obtain the combination that provides optimal results. For both the APF and the ORCA algorithms, the values of the parameters CRIT_DIST and ORI_ERROR will be modified. However, in the case of the APF algorithm, the modification extends to include the parameter W2, since what is truly relevant is to analyze its relationship with W1, thus it is only necessary to adjust one of them. On the other hand, for the ORCA algorithm, in addition to CRIT_DIST and ORI_ERROR, different values for time_step and time_horizon will be analyzed.

These values will always be within the range specified in Table I previously presented. And this phase will be carried out using the small scenario described in Section V-B, simulating a total of four AUVs to speed up the simulation process. It is also important to highlight that the value of the parameter DES_VEL will not be modified, since the simulation setup is not designed to simulate higher speeds. For this reason, it will be maintained at the value specified in Table I.

However, it is essential to note that non-deterministic behavior has been observed in the simulation results. This is attributed to the inherent characteristics of the COLA2 module[5], which allows simulating the dynamics of an AUV and its sensors, replicating the behavior it would have in a real environment. Therefore, to ensure the validity and reliability of the results, each parameter combination will be simulated five times. Thus, the obtained data represents the average of all AUVs involved in the simulation.

Therefore, for the APF algorithm, 36 parameter combinations have been simulated five times (180 simulations in total). Figure 16 represents the results obtained depending on the different values of the parameters CRIT_DIST, ORI_ERROR and W2.

During the 180 simulations, no collisions have ever been detected. However, as can be observed, in some simulations, AUVs have been detected entering a trapping situation.

The presented data suggests a clear correlation: a higher CRIT_DIST value corresponds to a greater number of successfully avoided collisions. This relationship is attributed to the expanded obstacle detection range, which leads to longer navigation and collision avoidance times, as well as an increased overall distance traveled. Notably, for CRIT_DIST values exceeding 10 meters, there's a risk of navigation time reaching the time limit (depicted by the red line), signaling that the AUVs may have entered in a trapping situation.

Additionally, Figure 16 shows that a higher value of ORI_ERROR yields superior results. This is because the AUVs come to a halt while rotating the required angle for adopting the new velocity, consequently prolonging navigation and collision avoidance times. Concerning the parameter W2, it suggests that, in general, lower values of repulsion force intensity lead to improved results. This can be attributed to the fact that a reduced intensity of the repulsion force results in smoother navigation trajectories for the AUVs.

Regarding the ORCA algorithm, Figures 17a and 17b show the results obtained depending on the different values of the parameters CRIT_DIST, ORI_ERROR, time_step and time_horizon. For this algorithm, 108 parameter combinations have been simulated 5 times (540 simulations in total) and no collisions have ever been detected. Figure 17 shows the results obtained during these simulations.

Specifically, in Figure 17a, contrasting with the APF algorithm, it is evident that better results are achieved with higher values of CRIT_DIST. This behavior is attributed to the increased obstacle detection range, allowing the AUV to identify a greater number of obstacles and subsequently determine the most efficient optimal velocity.

Regarding the parameter ORI_ERROR, at first glance, it seems that the higher the value, the better the results obtained. Although the difference is not very noticeable, since, in general, the results obtained with this algorithm do not show a great variability. This could be partly due to the simplicity of the environment (the AUV density).

On the other hand, in Figure 17b the results obtained depending on the different values of the parameters time_step and time_horizon are shown. Notably, these parameters exhibit minimal variation in their results, with performance consistently close to optimal levels.

Based on the presented data, optimal parameter combinations have been identified for both algorithms. For the APF algorithm, the most favorable results were obtained

[5]https://bitbucket.org/iquarobotics/cola2_wiki/src/5e46035647fd34f18105b58629ab2d89da04eea6/cola2_sim.md

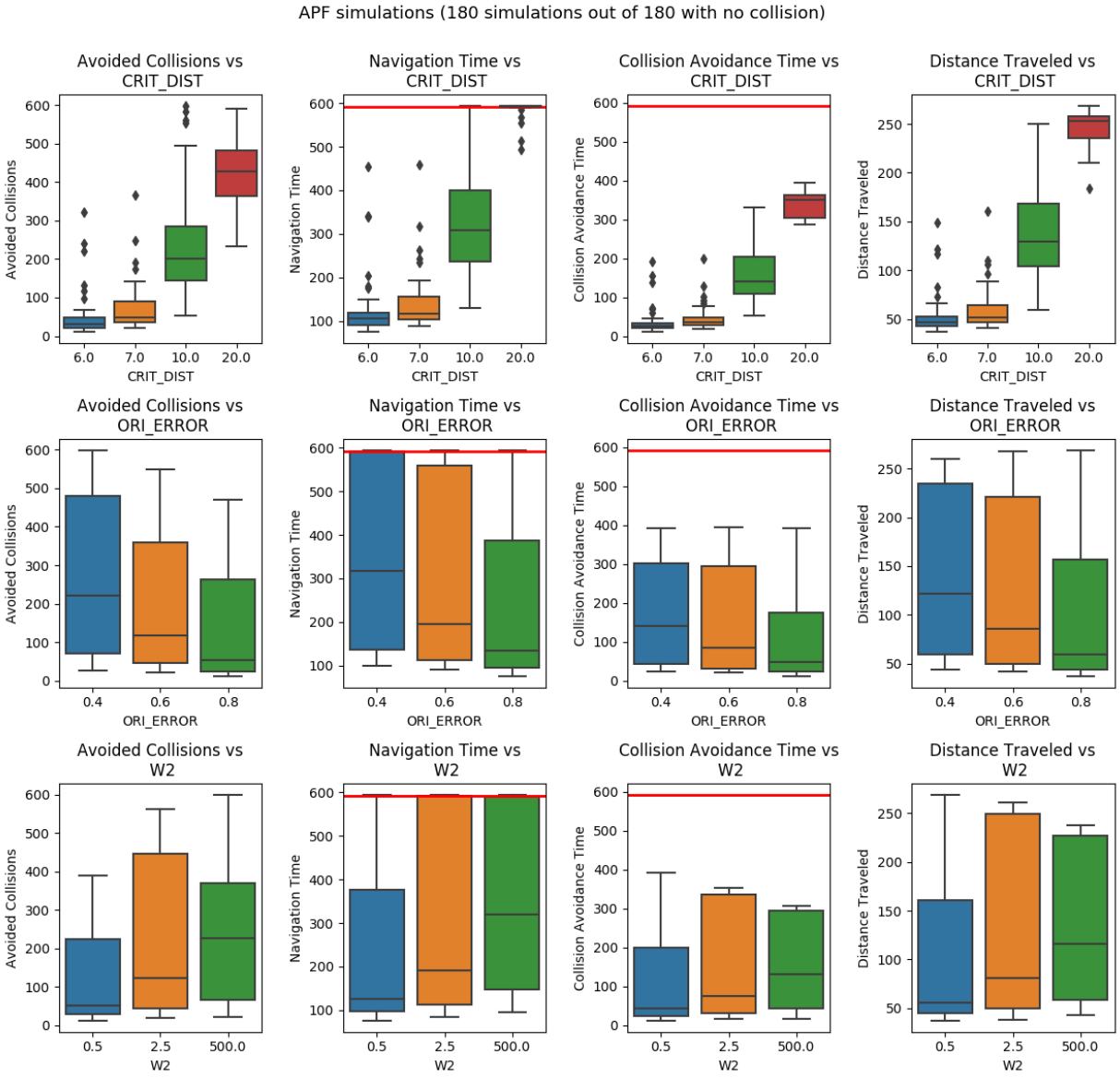


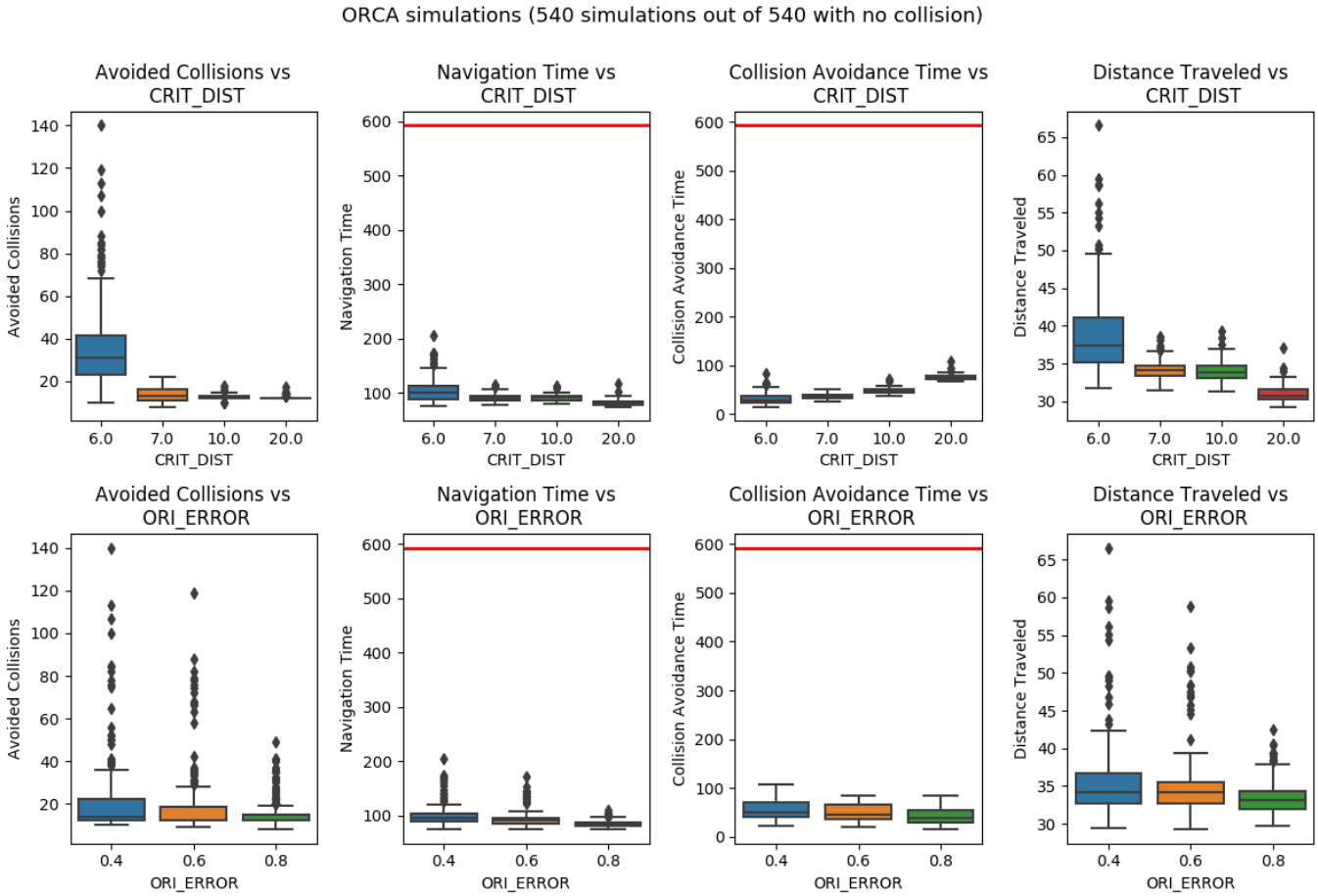
Figure 16: APF results in the parameter setting phase. Where the red line represents the time limit.

with parameter values of 6 meters for CRIT_DIST, 0.8 for ORI_ERROR, and 0.5 for W2. On the other hand, the ORCA algorithm demonstrated optimal performance with parameter values of 20 meters for CRIT_DIST, 0.6 for ORI_ERROR, 1 second for time_step, and 5 seconds for time_horizon.

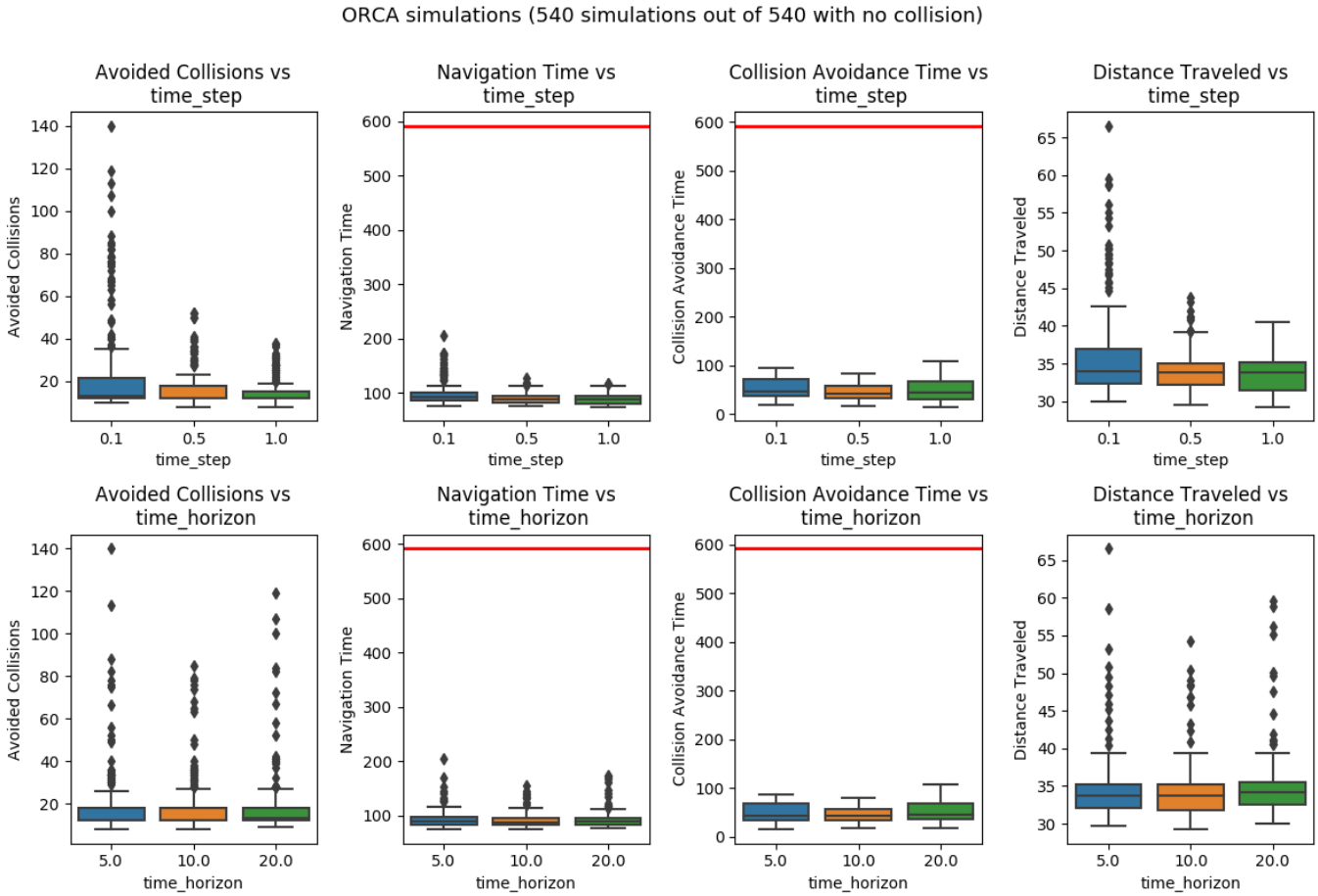
B. Comparison phase

After completing the parameter setting phase and identifying optimal parameter combinations for each algorithm, the next step is to compare the collision avoidance algorithms individually and in combination with the SAW strategy. In the initial test, the same scenario is simulated five times for each simulation type (APF, ORCA, APF with SAW, and ORCA with SAW). If no collisions have ever been detected, the number of AUVs will be increased, and the described process will be repeated. Subsequently, this final phase of the simulation process will be replicated in a larger scenario, ensuring that all scenarios listed in Table II from Section V-B are simulated. However, if collisions are detected, the parameters are adjusted manually until acceptable results are achieved.

These parameter combinations have been assigned to each respective algorithm, and the simulations have been conducted five times using the same scenario employed during the parameter setting phase. In Figure 18 a comparison between the different simulation types (APF, ORCA, APF with SAW, and ORCA with SAW) is presented to perform the comparative analysis.



(a) ORCA results with CRIT_DIST and ORI_ERROR parameters. The red line represents the time limit.



(b) ORCA results with time_step and time_horizon parameters. The red line represents the time limit.

Figure 17: Combined results of ORCA parameter setting phase.

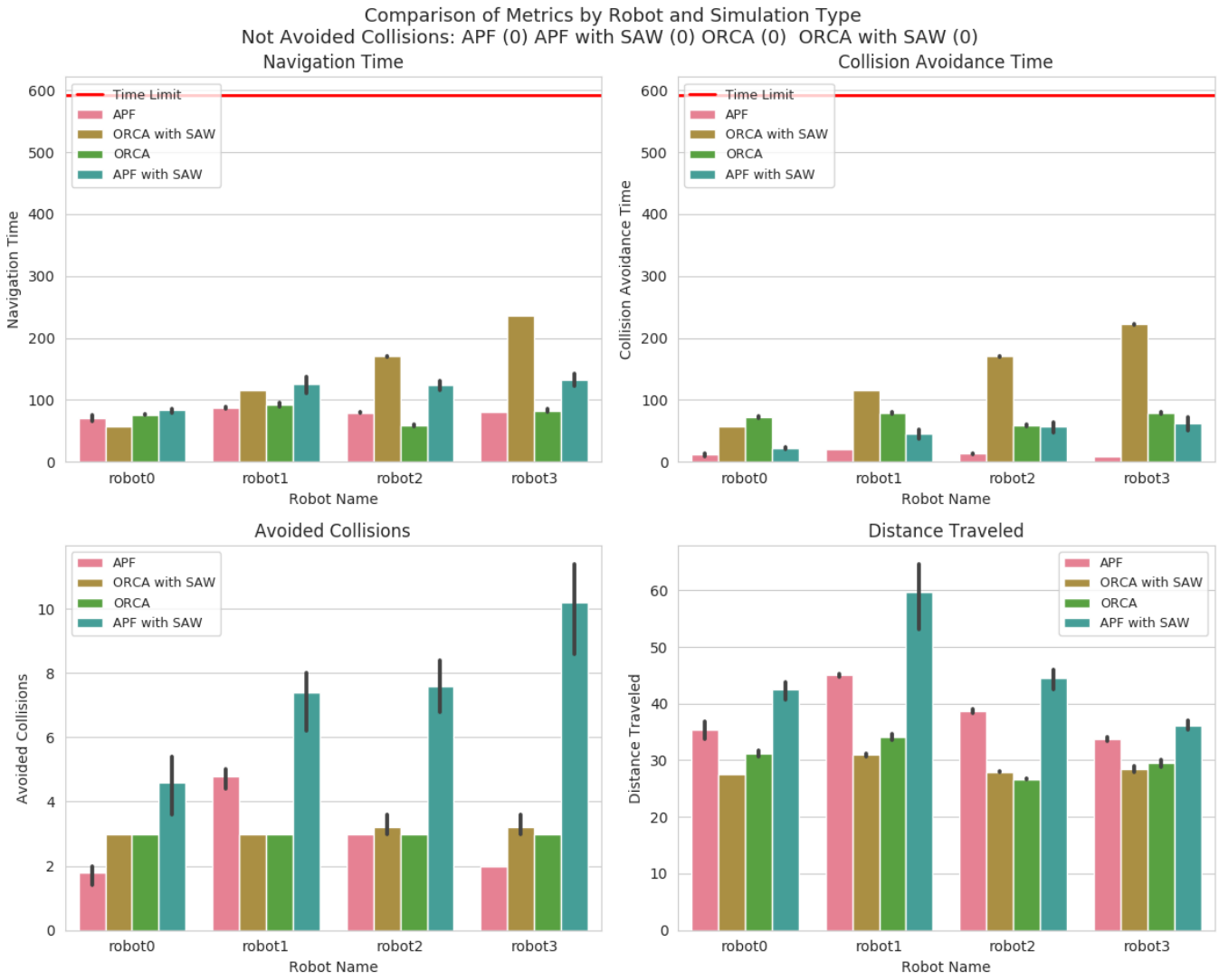


Figure 18: Comparison of Metrics by Robot and Simulation Type with the small scenario and 4 AUVs.

During the simulations, no collisions have ever been detected. Regarding navigation and collision avoidance times, it can be observed that the combination of the ORCA algorithm with the SAW strategy yields worse results. This is attributed to the higher value assigned to CRIT_DIST, leading to the detection of more AUVs. Consequently, AUVs with lower priority spend more time halted, waiting for the AUVs with higher priority to move away, resulting in sub-optimal results. In contrast, the APF algorithm, which uses a smaller value for the parameter CRIT_DIST, experiences the opposite effect, leading to an increased number of avoided collisions and distance traveled, and obtaining better values for the navigation and collision avoidance times. Therefore, it seems that the SAW strategy works better with the APF algorithm in this case, since the primary goal is to minimize the navigation time without detecting collisions. However, both the ORCA algorithm and the APF algorithm, when not combined with the SAW strategy, generally demonstrate better results, and their performance is quite similar. Although the ORCA algorithm presents better results in terms of distance traveled, and worse results in terms of collision avoidance time. Therefore, in this particular scenario, the SAW strategy has not proven to be of great utility

Since no collisions have been detected in the previous simulations, these experiments will be replicated with eight AUVs. Figure 19 shows the results obtained.

Upon increasing the number of AUVs, ten collisions have been detected to have occurred, six when applying the APF algorithm and four when applying the ORCA algorithm. However, no collisions have been detected when these algorithms are combined with the SAW strategy. It is worth noting that overall results have worsened, compared to the results of Figure 18.

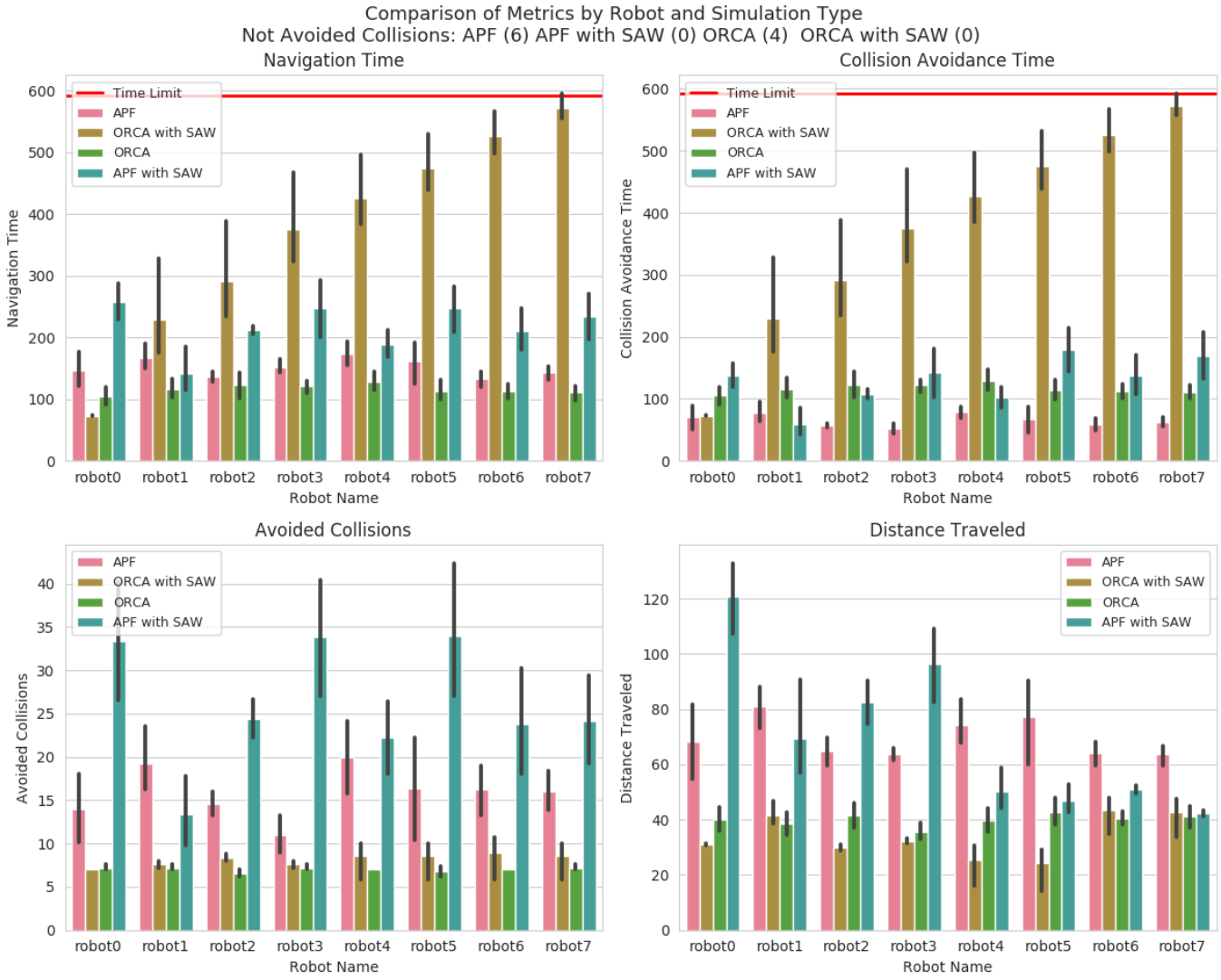


Figure 19: Comparison of metrics by AUV and simulation type with the small scenario and 8 AUVs.

Regarding the simulation type that combines the ORCA algorithm with the SAW strategy, it presents the worst results in terms of navigation time, since the AUV which name is "robot7" has been detected to have entered a trapping situation in some instances. In fact, as can be observed in this Figure 19, the AUVs with the lowest priorities present the worst navigation times, as expected.

Regarding the rest of simulation types, the ORCA algorithm demonstrates superior performance compared to the APF algorithm when not combined with the SAW strategy, in terms of navigation time. However, when combined with the SAW strategy, the APF algorithm yields better results than the ORCA algorithm.

In the light of the collisions detected, a new set of parametres has been simulated. Specifically, for the APF algorithm, the value of CRIT_DIST will be increased from 6 to 7 meters, with the intention of enabling AUVs to detect obstacles earlier to facilitate safer movements. Meanwhile, for the ORCA algorithm, the value of time_step will be reduced from 1.0 to 0.5 seconds. From the concepts described in Sections III-B and IV-C, this reduction of the time_step parameter aims to decrease the discretization period, enabling a more accurate prediction of the movements of neighboring AUVs and obtaining a more precise value for the new velocity vector.

Figure 20 shows the results obtained with the new combinations of parameters. As can be observed, no collisions have been detected. Moreover, no AUV has ever reached the time limit. And, once again, the ORCA algorithm outperforms the APF algorithm, when they are not combined with the SAW strategy. It is also noteworthy that the APF algorithm exhibits results with greater variability in this case.

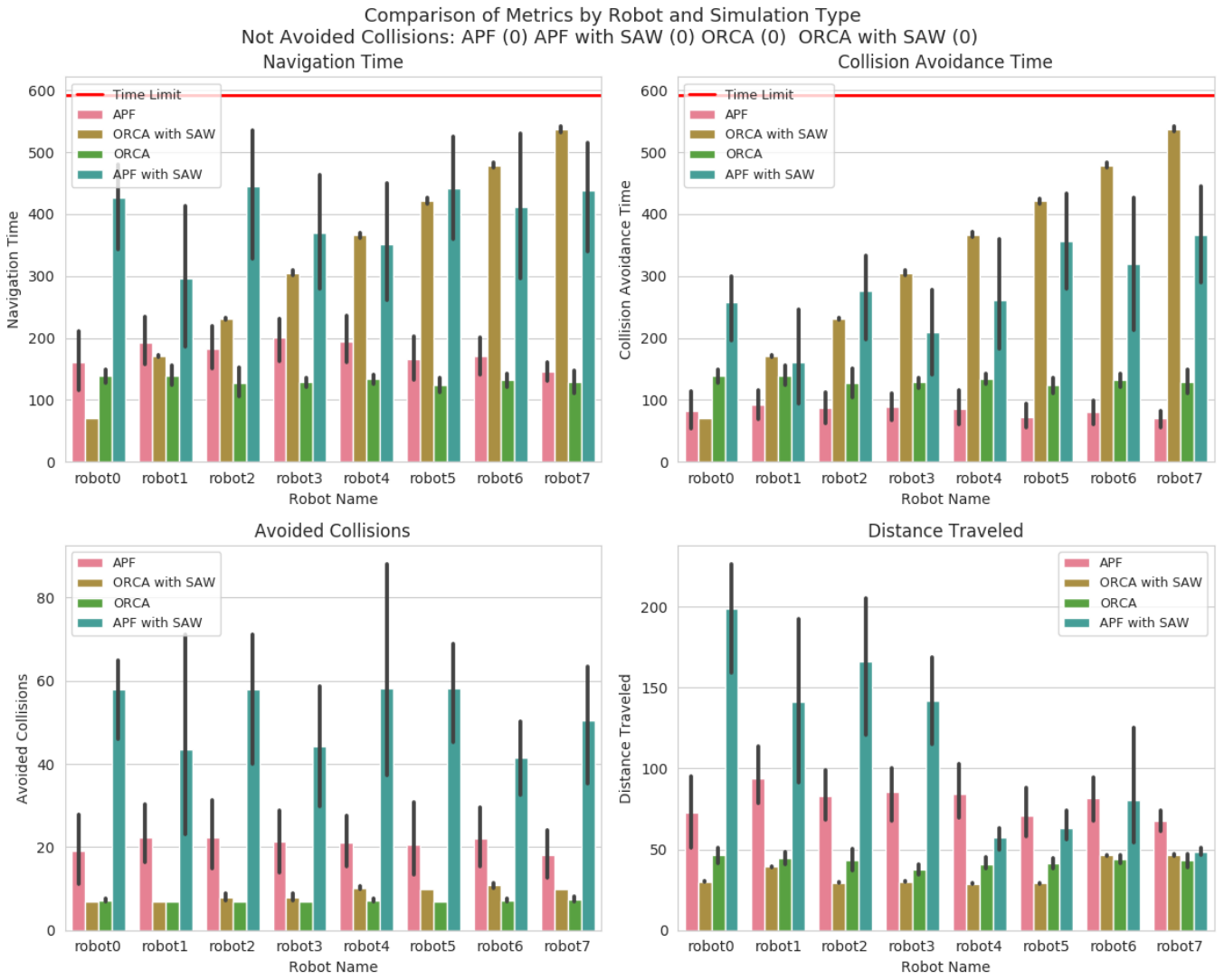


Figure 20: Second comparison of metrics by AUV and simulation type with the small scenario and 8 AUVs.

On the other hand, Figure 20 no longer reveals a clear difference in the behavior of these algorithms when combined with the SAW strategy, in terms of navigation time.

Next, the simulation will be conducted again with four AUVs, but this time in a larger environment. The results are presented in Figure 21

In Figure 21, it can be observed that the results are much more balanced. The navigation and collision avoidance times are significantly distant from the navigation time limit. Both algorithms present similar results in terms of navigation time, when not combined with the SAW strategy. However, the APF algorithm yields the worst results in terms of avoided collisions and distance traveled, when not combined with the SAW strategy. While the ORCA algorithm, yields the worst results in terms of navigation and collision avoidance times, when combined with this strategy.

Once again, the larger scenario will now be simulated with eight AUVs. The results are presented in Figure 22. As can be observed, collisions have been detected again, eight when applying the APF algorithm and four when applying the ORCA algorithm, both without SAW strategy. In this scenario where the AUV density has been increased, navigation times are significantly distant from approaching the time limit. The ORCA algorithm yields the best results in terms of navigation time and avoided collisions, but it produces the worst results in terms of navigation and collision avoidance times when combined with the SAW strategy. In terms of distance traveled, ORCA presents optimal results whether combined with the SAW strategy or not.

Therefore, aiming to enhance the results, the parameters will be adjusted once again for each algorithm, setting CRIT_DIST to 7 meters and time_step to 0.5 seconds, thus

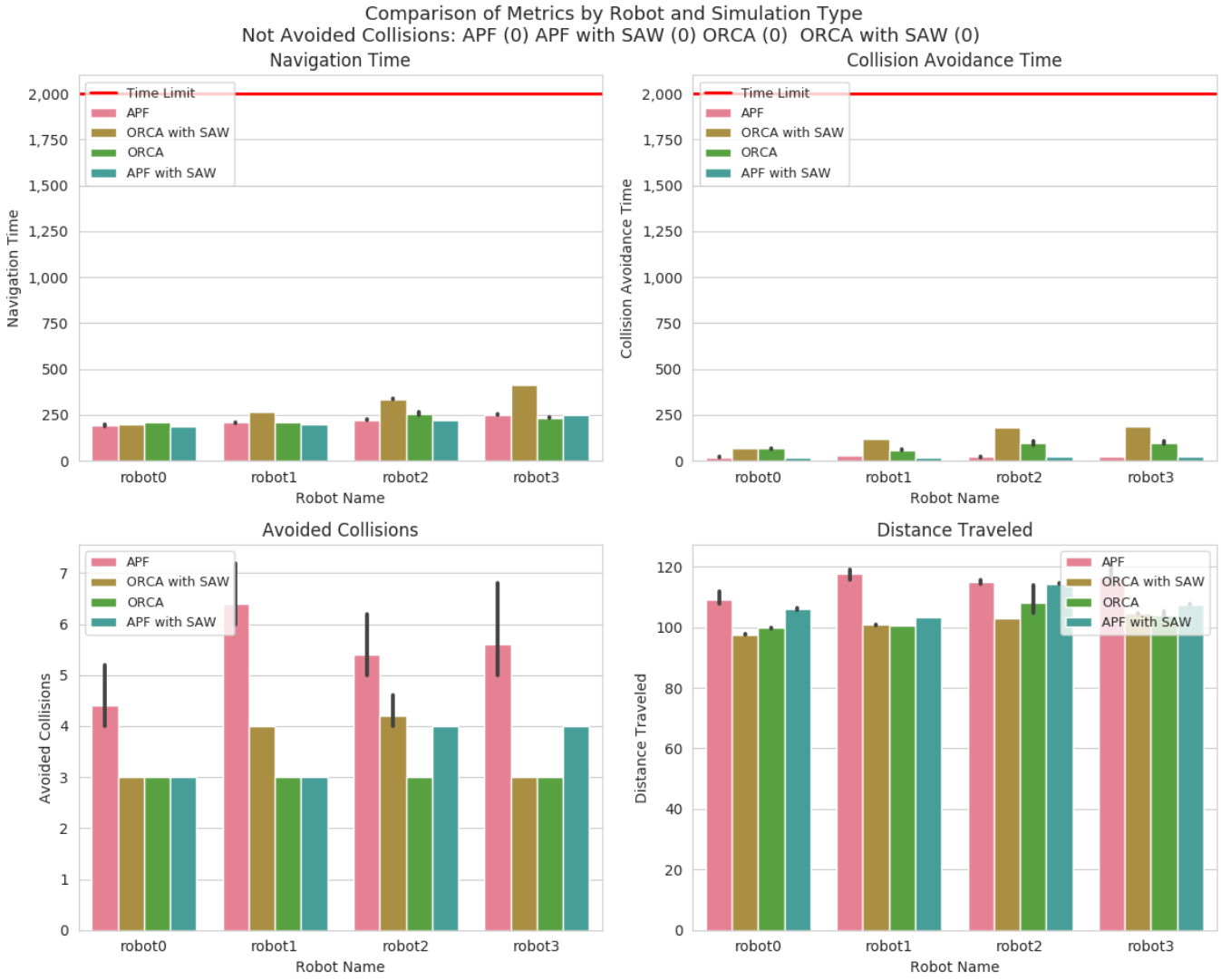


Figure 21: Comparison of metrics by AUV and simulation type with the larger scenario and 4 AUVs.

replicating the approach used on the previous occasion when collisions were detected (Figure 19). The results obtained with these parameter values are presented in Figure 23.

The simulations with these new parameter values show better results than the previous ones, as no collisions were detected in any simulation type. Additionally, no AUV has ever been detected to have entered in a trapping situation, and their navigation times and collision avoidance times remain relatively distant to the time limit compared to the results obtained with the smaller scenario of Figure 20, emphasizing the crucial role of AUV density in this context. Furthermore, the ORCA algorithm still presents the best results in terms of navigation time and avoided collisions but performs worse than the APF algorithm in terms of navigation and collision avoidance times when both are combined with the SAW strategy. Additionally, the results obtained when applying the APF algorithm still exhibit greater variability.

VII. Discussion

From the results presented in Section VI, it can be deduced that the ORCA algorithm is more efficient and precise than the APF algorithm when they are not combined with the SAW strategy. The simplicity of the APF algorithm, not specifically designed for MRSs, generally exhibits results with greater variability and worse navigation times. However, adjusting parameter values is critical for each situation, as increasing the number of AUVs may lead to undesired results.

Regarding the SAW strategy, despite yielding suboptimal results in terms of navigation and collision avoidance times, it proves to be a safe approach to avoid collisions. Both the ORCA and the APF algorithms, when combined with this strategy, do not generate any collisions in any simulation. However, this strategy tends to be more efficient with the

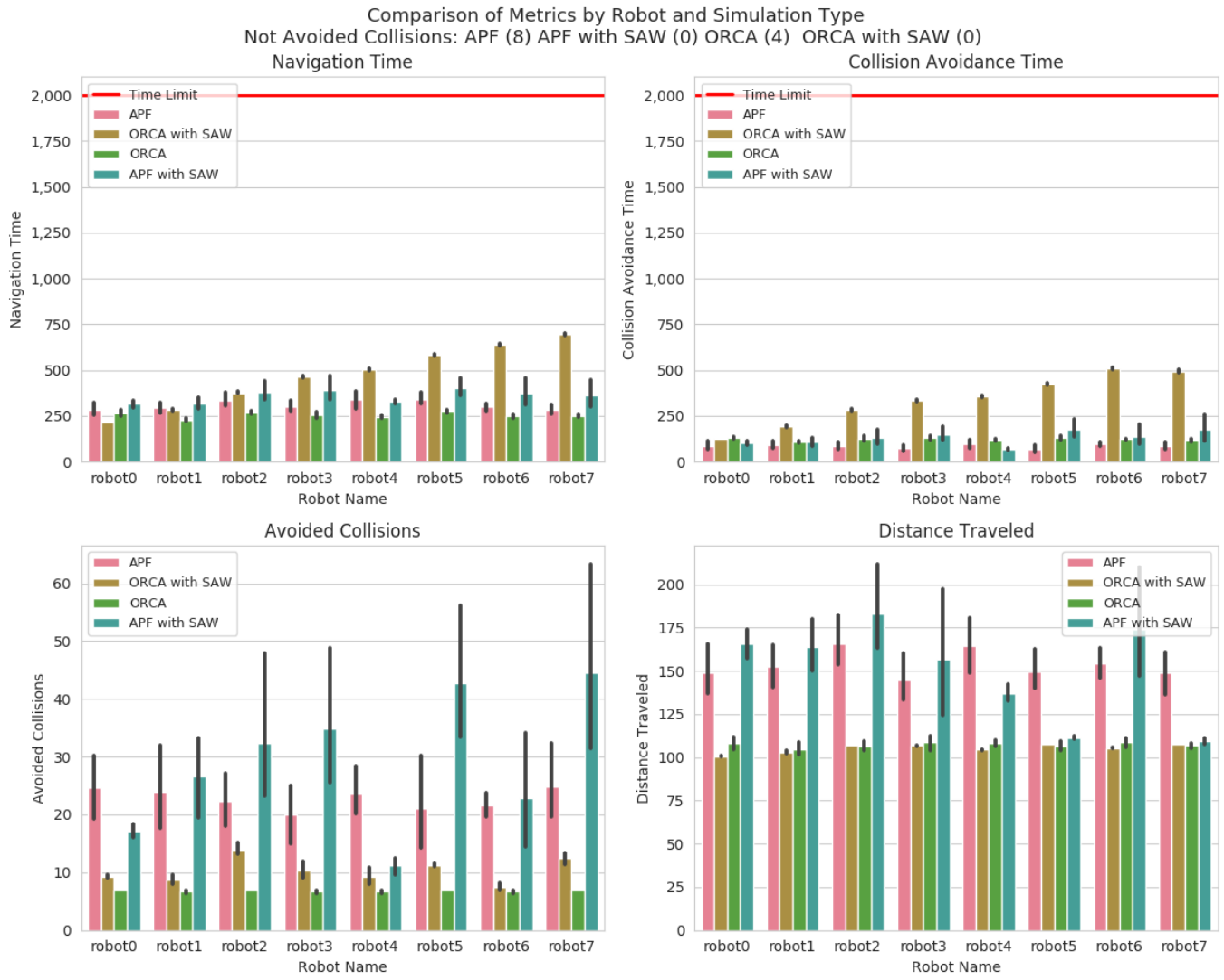


Figure 22: Comparison of metrics by AUV and simulation type with the larger scenario and 8 AUVs.

APF algorithm than with the ORCA algorithm, as the ORCA algorithm requires a greater obstacle detection distance, increasing the waiting time. However, this difference becomes less noticeable as the AUV density is increased.

As was described in Section VI, larger scenarios produce results with greater deviation from the time limit. Recall that this limit is calculated using the time_coefficient parameter, determining how many times the minimum navigation time can be exceeded. This minimum time is obtained from the AUV's maximum speed and the distance to the goal. Therefore, it is essential to consider the value of this parameter based on the AUVs' navigation paths.

From the experimental results, it can be concluded that the ORCA algorithm outperforms the APF algorithm. Nevertheless, if navigation time is considered a secondary factor, and there is a desire to ensure collision avoidance without taking risks, combining either algorithm with the SAW strategy proves to be a viable option. While the APF algorithm may provide better results than the ORCA algorithm when both are combined with this strategy, it should be noted that in some scenarios the APF algorithm generates more collisions between AUVs than the ORCA algorithm. In addition, the APF algorithm already has a very high number of avoided collisions. For this reason, the APF algorithm might detect and avoid more collisions.

In summary, it is crucial to be clear about the objectives to be achieved and conduct necessary simulations with different combinations of parameters to determine the optimal navigation strategy.

VIII. Conclusions

This Master's Thesis is motivated by the research of mechanisms to prevent collisions among AUVs in

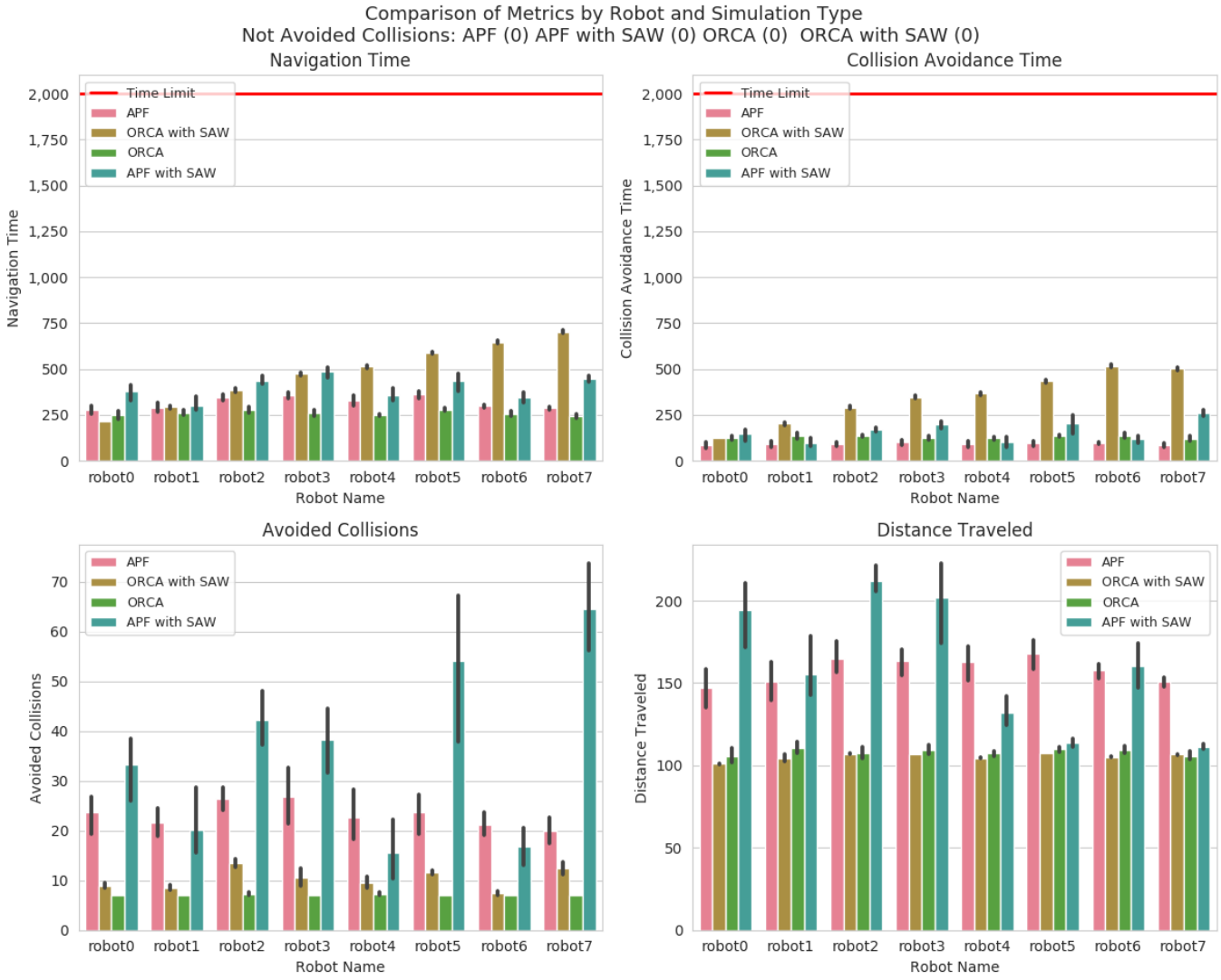


Figure 23: Second comparison of metrics by AUV and simulation type with the larger scenario and 8 AUVs.

challenging underwater environments. The work involves the implementation of two obstacle avoidance algorithms, the APF and ORCA algorithms, along with the SAW strategy, within the framework of a scalable MRS. The characteristics of the simulation environment employed and the decisions taken to force collision between AUVs while evaluating the effectiveness of the algorithms are outlined.

This work details a parameter setting phase, which includes multiple simulations with various parameter combinations for each algorithm to identify the optimal combination. Subsequently, these optimal combinations are used in additional simulations, varying the size of the scenario and the number of AUVs, in order to compare both algorithms and the SAW strategy.

The results obtained indicate that the ORCA algorithm demonstrates greater efficiency than the APF algorithm when applied individually, with APF providing results with greater variability and detecting more collisions. However, when combining both algorithms with the SAW strategy, results related to navigation and collision avoidance times are affected, sometimes significantly increasing, even leading to trapping situations. Remarkably, despite this, the SAW strategy has never detected collisions, and navigation and collision avoidance times tend to improve when combined with the APF algorithm.

In conclusion, the choice of the navigation strategy depends on specific objectives. If prioritizing navigation time, the ORCA algorithm might be the preferred option. On the other hand, if the safety of AUVs is the primary concern, the SAW strategy proves to be a robust tool. However, it is crucial to evaluate each situation and conduct an extensive parameter setting phase to optimize the efficiency of the algorithms.

Furthermore, it is essential to note that the application of these algorithms and strategies in real-world environments is still pending, and their validation in practical scenarios remains a crucial next step.

REFERENCES

[1] M. F. Ali, D. N. K. Jayakody, Y. A. Chursin, S. Affes, and S. Dmitry. Recent advances and future directions on underwater wireless communications. *Archives of Computational Methods in Engineering*, 27:1379–1412, 11 2020.

[2] H. Barki, F. Denis, and F. Dupont. Contributing vertices-based minkowski sum computation of convex polyhedra. *CAD Computer Aided Design*, 41:525–538, 2009.

[3] J. V. D. Berg, S. J. Guy, M. Lin, and D. Manocha. Reciprocal n-body collision avoidance. 2011.

[4] J. V. D. Berg, M. Lin, and D. Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. 2008.

[5] X. Cao, H. Sun, and L. Guo. Potential field hierarchical reinforcement learning approach for target search by multi-auv in 3-d underwater environments. *International Journal of Control*, 93:1677–1683, 7 2020.

[6] M. Carreras, C. Candela, D. Ribas, A. Mallios, L. Magí, E. Vidal, N. Palomeras, and P. Ridao. Sparus ii, design of a lightweight hovering auv m5. 2018.

[7] M. Carreras, J. D. Hernandez, E. Vidal, N. Palomeras, D. Ribas, and P. Ridao. Sparus ii auv - a hovering vehicle for seabed inspection. *IEEE Journal of Oceanic Engineering*, 43:344–355, 4 2018.

[8] C. Cheng, Q. Sha, B. He, and G. Li. Path planning and obstacle avoidance for auv: A review. *Ocean Engineering*, 235, 9 2021.

[9] G. H. D. Alejo, J. A. Cobano and A. Ollero. Optimal reciprocal collision avoidance with mobile and static obstacles for multi-uav systems. *IEEE Robotics and Automation Society and Institute of Electrical and Electronics Engineers*, page 1316.

[10] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *International Journal of Robotics Research*, 17:760–772, 1998.

[11] S. Huang, R. S. H. Teo, and K. K. Tan. Collision avoidance of multi unmanned aerial vehicles: A review. *Annual Reviews in Control*, 48:147–164, 1 2019.

[12] C. Hui, Z. Qiyuan, L. Zhongchang, X. Tianye, and L. Liang. Decentralized navigation of multiple agents based on orca and model predictive control. 2017.

[13] H. R. Kam, S. H. Lee, T. Park, and C. H. Kim. Rviz: a toolkit for real domain data visualization. *Telecommunication Systems*, 60:337–345, 10 2015.

[14] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5, 1986.

[15] J. J. Kuffner. Rrt-connect : An efficient approach to single-query path planning, 2000.

[16] S. Kumar and C. Vats. Underwater communication: A detailed review. 2021.

[17] A. Martorell-Torres, E. Guerrero-Font, J. Guerrero-Sastre, and G. Oliver-Codina. Xiroi ii, an evolved asv platform for marine multirobot operations. *Sensors*, 23, 1 2023.

[18] M. V. Musa Morena Marcusso Manh~aes, Sebastian A. Scherer and L. R. Douat. *UUV Simulator: A Gazebo-based Package for Underwater Intervention and Multi-Robot Simulation*. IEEE, 2016.

[19] E. M. Nebot. Sensors used for autonomous navigation, 1999.

[20] N. Palomeras, A. El-Fakdi, M. Carreras, and P. Ridao. Cola2: A control architecture for auvs. *IEEE Journal of Oceanic Engineering*, 37:695–716, 2012.

[21] T. R. Player, A. Chakravarty, M. M. Zhang, B. Y. Raanan, B. Kieft, Y. Zhang, and B. Hobson. From concept to field tests: Accelerated development of multi-auv missions using a high-fidelity faster-than-real-time simulator. 11 2023.

[22] M. J. D. Powell, S. Lasdon, J. D. Schoeffler, B. P. Dzielinski, R. E. Gomory, P. E. Hart, N. J. Nilsson, and B. Raphael. The gradient projection method for nonlinear programming, pt. i, linear constraints, 1968.

[23] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng. Ros: an open-source robot operating system.

[24] J. Rice. Underwater acoustic measurements: Technologies and results" heraklion.

[25] N. Saeed, A. Celik, T. Y. Al-Naffouri, and M. S. Alouini. Underwater optical wireless communications, networking, and localization: A survey. *Ad Hoc Networks*, 94, 11 2019.

[26] J. Sun, J. Tang, and S. Lao. Collision avoidance for cooperative uavs with optimized artificial potential field algorithm. *IEEE Access*, 5:18382–18390, 8 2017.

[27] J. Susi, F. Moch, N. S. M. Susiki, and H. Mochammad. Crowd navigation using leader-follower algorithm based reciprocal velocity obstacles. 2016.

[28] C. Wang, J. Wang, Y. Shen, and X. Zhang. Autonomous navigation of uavs in large-scale complex environments: A deep reinforcement learning approach. *IEEE Transactions on Vehicular Technology*, 68:2124–2136, 3 2019.

[29] M. Wzorek, C. Berger, and P. Doherty. A framework for safe navigation of unmanned aerial vehicles in unknown environments. volume 2017-January, pages 11–20. Institute of Electrical and Electronics Engineers Inc., 11 2017.

[30] H. Yu and L. Ning. Coordinated obstacle avoidance of multi-auv based on improved artificial potential field method and consistency protocol. *Journal of Marine Science and Engineering*, 11, 6 2023.

[31] W. Yu and Y. Lu. Uav 3d environment obstacle avoidance trajectory planning based on improved artificial potential field method. volume 1885. IOP Publishing Ltd, 4 2021.