RADAR vs LIDAR for obstacle detection and collision avoidance

Tiago Filipe Félix de Andrade
tiago.andrade@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal

October 2020

Abstract

The field of autonomous vehicles is a dynamic and ever-growing field of research which has seen a rise in popularity in recent years. Research in this area has mostly shifted towards development of increasingly accurate and versatile sensing systems to be integrated in robust autonomous navigation algorithms, culminating in highly autonomous vehicles. Diversity in the field of sensing is significant, with a high number of developed sensing methods to acquire a variety of environment data used in autonomous navigation. This thesis focuses on two of the most commonly used sensors, LIDAR and Doppler RADAR, exploring the technology behind these devices. The main goal of this work is the development of an obstacle detection and collision avoidance system incorporating these sensors and its implementation in a small-scale robot capable of autonomously navigating unknown environments to reach a single or multiple waypoints. A physical prototype is developed as well as its virtual counterpart to simulate its behaviour in a controlled, risk-free environment intended for prior testing. Modularity is a significant factor in this work, with the intent of easing the integration of the prototype in any Wi-Fi network. The capabilities of the ROS framework are explored with its integration in the assembly. Finally, the sensing system is put to test in a set of trials, evaluating the performance of the standalone sensors as well as its coupling (through sensor fusion) in static and dynamic environments.
Keywords: LIDAR, Doppler RADAR, Obstacle detection, Collision avoidance, ROS

1. Introduction

Mechatronics systems in general have been burdened with human dependence when operating, although to a limited degree, due to the unpredictable nature of the environment they are operating in.

With the current trend of lowering costs of sensory components and processing units coupled with the rapid development of hardware and software integrated in these sensing systems, autonomous systems have seen an increase in popularity across the manufacturing industry as well as services and domestic applications. Nowadays, software implementations of obstacle detection algorithms are quite advanced and robust to a point in which their performance is mostly limited by the hardware performing the readings and computations. As a result, research has mostly shifted towards improving the hardware supporting the computations behind these complex algorithms as well as the sensing hardware needed to obtain environment data, since these are the determining factor of an algorithm's precision.

Among the many options of sensing systems available to perform this task, some of the most accurate, each with their own advantages and drawbacks, are sensors which make use of electromagnetic waves for surveying the surrounding area. Often coupled with computer vision-based algorithms to complement their readings, these are the most commonly used sensors due to their accuracy and precision.

1.1. State-of-the-art

The field of autonomous robotics is a widely researched topic with diverse approaches due to the versatility of the subject itself. Among systems solely capable of obstacle detection and collision avoidance, a number of relevant implementations are worth special mention due to similarities with this work.

In [6], the implementation of a short-range obstacle detection and collision avoidance algorithm applied to LIDAR sensing is explored. The robot is equipped with a set of 4 mechanum wheels, granting the ability to move in any direction without shifting its pose (omnidirectional motion). Utilizing basic strafing motions, the robot is set to follow the outline of a wall, avoiding any protrusions it may encounter. Although sensing data is collected, no mention of clustering or real time monitoring is made. Nonetheless, the robot is fully autonomous along any unknown wall-like paths with computational operations being handled by the robot's own

processing unit, granting a high degree of modularity.

In [7], on the other hand, an implementation of long-range collision avoidance using LIDAR sensing is developed. With an effective median clustering algorithm, the robot is able to segment and reconstruct clusters as similar basic geometries (lines, rectangles and circles). Data obtained by the sensor is imported to a *Matlab* instance where avoidance is computed. However, such an implementation increases the bulk of the algorithm, introducing the necessity of a good processing unit to achieve real-time avoidance. A simulation test is carried out in the imported scenario to validate the algorithm. No mention of omnidirectional motion is made and simulations are limited to imported scenarios.

[3] explores an implementation of LIDAR sensing in short-range obstacle avoidance. A thorough study is made of the performance when sensing objects with various different physical traits such as color and size. Motion control is based on modifying each of the rear motors' angular speeds (tricycle configuration). Obstacle avoidance bulk is minimized, increasing or decreasing motor speed based on proximity to obstacles on either side of the robot. While basic, this approach demonstrates a good performance when navigating narrow obstacle courses. Computational operations are handled by a single Raspberry Pi 3 acting as a master to all the assembly hardware. No mention of real-time data monitoring or simulation features is made.

In [9], the authors opt for an approach of short-range LIDAR sensing applied to a collision avoidance and SLAM algorithm, using the ROS framework as a means of data communication and visualization. A Pioneer 3-DX robot is used for its compatibility with the ROS framework. An improvement to the expanded guide circle method for obstacle avoidance is proposed, improving stability of the avoidance trajectory. A selective decision-making approach is made, allowing the robot to navigate the environment in either entry or bypass mode. These modes allow the robot to either navigate gaps and narrow spaces between detected obstacles (if avoidance is possible) or bypass them altogether, selecting an orbit-like trajectory to avoid dense obstacle zones. This grants a high versatility to the implementation, allowing it to navigate sparse or dense obstacle courses with ease. However, such algorithms pose the need of a high resource processing unit such as a computer. No mention of virtual simulation capabilities is made.

RADAR sensing implementations in small-scale collision avoidance implementations are uncommon, denoting a slight bias or favouring LIDAR sensing in these conditions. Nevertheless, these are not non-existant, with a few pieces of research worth mention. In [8], a RADAR-based approach to this problem is presented, with a focus on exploring the limitations of RADAR sensing applied to less controlled scenarios such as roads and low visibility environments. It is shown that RADAR sensing is mostly impervious to low visibility conditions, being able to navigate long sections of a gravel road with sparse vegetation in either side as obstacles. Despite the good performance, this implementation's bulk demands a good processing unit, with a laptop being used for computations.

All aforementioned research works culminate in a common conclusion, rendering either LIDAR or RADAR to be effective sensing methods for collision avoidance implementations when complemented with a variety of decision-making algorithms, each with their own advantages and drawbacks. However, the majority of academic work focuses on the application of each of these devices as standalone data collection platforms to the writer's knowledge, not providing a means of comparison between the two or advantages of sensor fusion. Simulation capabilities are scarce, limiting the range of possible trial scenarios and providing no means of testing in controlled, risk-free scenarios prior to real world trials.

Prototypes are mostly designed with no computational resource limitations, resulting in the necessity of a bulkier processing unit. While such an option often offers the possibility of real-time data monitorization, it hinders the implementation in terms of modularity.

2. Background

2.1. Sensor performance

A theoretical approach to the expected performance of the sensors in this work can be made. Both of these sensors perform differently when working under different environments and subjected to noise from various sources. A few significant environmental factors, as well as constraints of the technology behind these sensing systems may significantly impact their performance:

- Due to health regulations, the **scanning range** of LIDAR systems is indirectly constrained by the maximum allowed radiating power. Due to being harmful to the human eye, the maximum radiating power of any LIDAR sensor is limited by law, hindering its sensing capabilities. RADAR systems however are not constrained, as the emitted waves are not harmful. Furthermore, LIDAR sensor waves are attenuated by the earth's atmosphere to a slightly higher degree than RADAR waves;
- **Lighting conditions** may also influence the performance of these sensing systems. Both sensors are expected to perform at their peak

during night-time. However, during the day, LIDAR sensor readings can be influenced by solar radiation as some of these waves fall under the same wavelength as the electromagnetic wave range used in LIDAR sensors. RADAR sensors, however, are unnafected by solar radiation;

- **Wave penetration** can play a significant part in sensor performance. RADAR sensor waves are characterized by longer wavelengths as opposed to the LIDAR sensor's shorter wavelengths. Thus, RADAR waves are capable of penetrating or passing around detected obstacles, being able to detect objects located behind said obstacles. LIDAR sensor waves are unable to detect obscured obstacles;

- **Resolution** is a significant factor in the comparison of these sensing systems. Range resolution is heavily dependant on wavelength, favoring LIDAR sensing due to its shorter wavelength. Angular resolution follows the same trend, favouring LIDAR sensing. While LIDAR sensors require a single emitter to perform radial scanning with a high resolution due to wave propagation speed, RADAR require multiple fixed emitters to perform readings at a significantly lower resolution;

- **Weather conditions** are an impactful factor in the performance of both sensors. Low visibility due to fog or rain contribute to the attenuation of electromagnetic waves, reaching up to a 25% performance decrease in the case of LIDAR sensors[5].

- **Reflectivity** is a key property of electromagnetic waves which greatly influences the performance of electromagnetic wave-based sensors. Due to the increased wavelength and wider beam form of radio waves used in RADAR systems, these are characterized by a low absorptivity when in contact with obstacles. For this reason, RADAR waves have a high reflectivity, often leading to multipath reflections where an emitted wave may be reflected multiple times and then received in a different direction of arrival (different wave direction from when it was emitted by the source), generating "ghost targets" (detections from non-existent obstacles) [4]

3. Implementation

A virtual simulator as well as a physical implementation were developed in this work. Both models' motion systems were based on the Omni-ANT platform developed in [1]. The Omni-ANT platform is an omnidirectional vehicle equipped with a set of 3 58mm omnidirectional wheels, each powered by an EMG30 motor controlled by two MD25 motor controller boards.

3.1. Vehicle Kinematics

With the established assembly of the Omni-Ant platform, the determination of the kinematic model of the prototype follows. The wheel configuration diagram for the Omni-Ant platform is shown in figure 1. The vehicle's wheels are paired with the corresponding numbered motor (wheel n being powered by motor n). Due to the omnidirectional nature of the wheels and their configuration, the prototype is able to move in any direction without having to change its orientation. This translates into a holonomic behaviour, where the only constraints it is subjected to are positional constraints of the form (the robot is a rigid body, thus nondeformable and, as such, the relative position of its components are constrained to their initial state). Thus, the translational and rotational movement relations in the local reference frame of the prototype are given by:

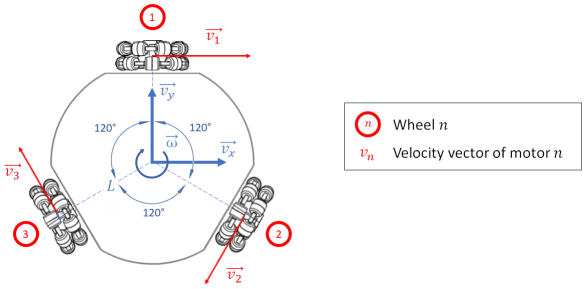

Figure 1: Omni-ANT platform wheel configuration (top view)

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} 1 & -\cos 60° & -\cos 60° \\ 0 & -\tan 30° & \tan 30° \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \tag{1}$$

where v_x is the x direction component of the velocity vector of the platform, v_y the y direction component of the same vector and v_n the linear velocity vector of wheel n. Assuming the no-slip condition, the linear velocity of a single wheel is given by:

$$v_{wheel} = \omega_{motor} r_{wheel} \tag{2}$$

where v_{wheel} is the linear velocity of the wheel, r_{wheel} its radius and ω_{motor} the angular velocity of the motor powering the wheel. By combining equations (1) and (2), the relation between the robot's linear velocity components and the angular velocity of each of its wheels is given by:

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} r_1 & -\frac{r_2}{2} & -\frac{r_3}{2} \\ 0 & -\frac{r_2\sqrt{3}}{2} & \frac{r_2\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \tag{3}$$

where r_n is the radius of wheel n and ω_n is the angular velocity of wheel n, with $n = 1, 2, 3$. The angular velocity of the platform in turn is given by:

$$\omega = \frac{v_1 + v_2 + v_3}{3L} \Leftrightarrow \omega = \frac{\omega_1 r_1 + \omega_2 r_2 + \omega_3 r_3}{3L} \quad (4)$$

where ω is the rotational speed of the platform and L is the distance between the rotational center and each of its wheels. Since the three wheels are identical, the local kinematic model of the robot can be simplified:

$$\begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} = r \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & -\frac{\sqrt{3}}{2} & \frac{\sqrt{3}}{2} \\ \frac{1}{3L} & \frac{1}{3L} & \frac{1}{3L} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \quad (5)$$

where r is the radius of the wheel model. A transformation is required to obtain the linear and angular velocity of the robot in the fixed reference frame, given by:

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} = T' \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \quad , \quad T = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (6)$$

where θ is the rotation of the robot in relation to

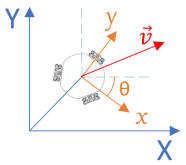

Figure 2: Local and fixed reference frames

the inertial reference frame (vehicle attitude, figure 2). As such, the relation between linear and angular velocities of the robot in the fixed reference frame and the angular velocity of the wheels is given by:

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\theta} \end{bmatrix} = T' r \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & -\frac{\sqrt{3}}{2} & \frac{\sqrt{3}}{2} \\ \frac{1}{3L} & \frac{1}{3L} & \frac{1}{3L} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \quad (7)$$

where $\dot{\theta}$ is the rate of change of the vehicle's attitude in the fixed frame (which is equal to the angular velocity of the robot in its local frame)

3.2. Physical Implementation

As previously mentioned, the Omni-ANT platform was used as the prototype's motion system. The processing unit of the robot is a Raspberry Pi 3B running the *Raspbian* operating system. The sensing system is composed of a URG-04LX-UG01 Lidar sensor as well as a TI mmWave AWR1642BOOST RADAR sensor. No changes to the firmware of these devices were made. Figure 3 depicts the final assembly of the autonomous robot prototype. The Robot Operating System (ROS)

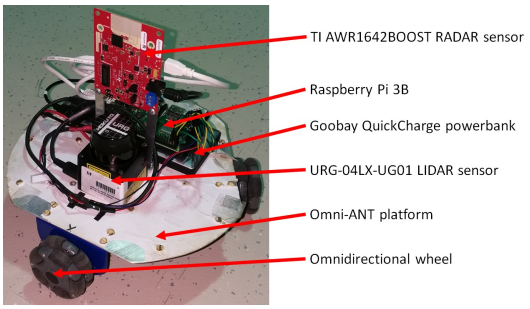

Figure 3: Autonomous robot prototype

was used as the communication network between all the devices in the assembly as well as an external master device to issue commands to the robot. The master node was set to execute in the external master device (a desktop computer running ROS through *Matlab* 2018) as well as a command publishing node. This master node can however be set to run in the robot's processing unit without compromising performance. Several publisher and/or subscriber nodes were developed to run in the robot's processing unit, publishing and/or subscribing to sensor data, odometry data, obstacle detections as well as obstacle avoidance commands and waypoint commands. At any point during robot activity, any of the data topics of the ROS network can be subscribed to and published in by any external ROS terminals connected to the same network. This ensures a high degree of modularity to the system, being able to be connected to and controlled from any ROS network (when running the ROS master node on the robot's processing unit).

3.3. Virtual Simulator

A virtual implementation was developed with the goal of simulating the three-wheeled robot traversing an obstacle course defined by the user. This simulator was programmed in *Matlab*'s *Simulink*, making use of its virtual world capabilities.

Based on the identification of the EMG30 motor in [2], a state-space model model of each motor was obtained:

$$\begin{bmatrix} \dot{i} \\ \dot{\omega} \end{bmatrix} = A \cdot \begin{bmatrix} i \\ \omega \end{bmatrix} + B \cdot U, y = C \cdot \begin{bmatrix} i \\ \omega \end{bmatrix} + D \cdot U$$

with

$$A = \begin{bmatrix} -2088.5 & -149.7059 \\ 89.7707 & -0.1642 \end{bmatrix}, \quad B = \begin{bmatrix} 294.1176 \\ 0 \end{bmatrix},$$
$$C = \begin{bmatrix} 0 & 1 \end{bmatrix}, \quad D = 0 \quad (8)$$

where i is the current applied to the motor, U is the voltage input and ω is the motor's angular speed. Non-linearities are modelled as a saturation in the

voltage input range, a quantization of the voltage input signal and a dead zone to simulate static friction. A linear relation between desired angular velocity and corresponding supply voltage is obtained [2]:

$$\omega_{desired} = \frac{K_s U_{command}}{K_s^2 + RB} = 1,9158 \cdot U_{command} \quad (9)$$

As velocity references are provided as cartesian components, the kinematic model of the robot is required to convert these references into desired wheel angular velocity inputs and convert the response of the motor models to these inputs into cartesian velocity response. The velocity response is then integrated to obtain the absolute robot position and the loop is closed for position control. The robot's speed is limited to a maximum value to ensure the correct functioning of the robot's algorithm.

The virtual world was designed using *Matlab*'s 3D world editor.

3.4. Data processing algorithms

3.4.1 Obstacle detection

The obstacle detection process is initialized when sensor data is received. A range of 1m was defined as the maximim scanning range. Upon receiving the detection data, this message is scanned detections within the defined scanning range. Dynamic detections obtained by the RADAR sensor are extracted regardless of proximity. Any detections which verify this condition are extracted and the remaining data is discarded. With the raw detection data from both sensors within the maximum scanning range, it is then transmitted to the clustering algorithm.

3.4.2 Clustering

The clustering algorithm is responsible for grouping the relevant raw range and velocity readings into groups to minimize processing time. Since the prototype is intended to detect obstacles on an unknown environment, the clustering algorithm is based on the proximity between detections. As such, detections within a minimum threshold distance from each other are grouped into the same cluster, extracting the centroid, velocity and radius of said cluster in the case of RADAR data or centroid and radius information in the case of LIDAR data.

3.4.3 Sensor fusion

Following the clustering of the cluster data, if both sensors are selected to be used simultaneously, moving objects characterized by the RADAR velocity readings are paired by proximity to their LIDAR cluster counterparts. Any velocity readings within a distance threshold from a cluster's centroid are grouped and the average velocity of said readings is associated with the same cluster.

3.4.4 Collision avoidance algorithm

The collision avoidance algorithm used in this implementation is denominated **minimal deviation velocity obstacles method**. Based on the standard velocity obstacles approach, this algorithm builds the instantaneous collision cones based on the obstacles present in the current scene. Only obstacles within the robot's warning zone (robot-centered radius in which collision avoidance is meant to be active) and at least one of the sensor's field of view (FOV) are considered for this step. This zone is defined as **active avoidance zone**.

In an initialization phase, a set of feasible controls is defined, containing a discrete set of possible velocity vectors with a fixed norm, defined by the robot's intended cruise speed. Then, using the processed sensor data, the collision cones of each obstacle are determined, containing every possible trajectory which would lead to a collision, assuming static obstacles. Figure 4 (left) is an example of a scene where the Omni-ANT is navigating an environment containing both static and moving obstacles within its active avoidance zone. The collision cones relative to each of the obstacles are represented, as well as the velocity vectors of the robot and obstacles.

The robot's dimensions must to be taken into account when determining the **real collision cones**. Therefore, the collision cones must be expanded to accomodate the passage of the robot. Two examples of robot trajectories are presented in figure 4 (right). As evidenced, avoidance controls (robot

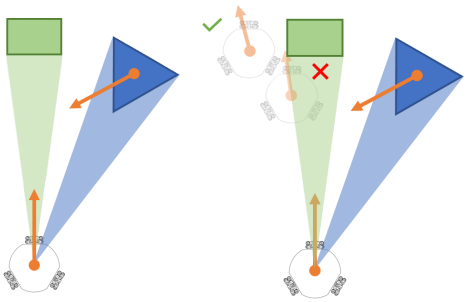

Figure 4: Velocity obstacles scene (left) and avoidance controls example (right)

commands that ensure avoidance of obstacles) in the vicinity of the collision cone thresholds may result in collisions if theses are built based on a point-particle model of the robot (rigid body defined by its mass concentrated in a single dimensionless point). Thus, a transformation must be applied beforehand expand the collision cones and correctly build the collision velocity space. The ap-

plication of a safety factor is also recommended due to the limited precision and noise of the sensing system. Following this correction and, contrary to the standard velocity obstacles approach in which the collision cones of non-static obstacles are displaced by the associated velocity vector, this algorithm will analyze each of these collision cones in separate, thus reducing computational load. However, the velocity vector of each of the obstacles must be subtracted to the feasible control prior to being tested. The orientation of the resulting vector is then compared with the angular thresholds of the detected obstacles. If a feasible control is found to result in an impending collision with any obstacle, it will be flagged as a non-avoidance control and removed from further comparisons. Applying the collision cone correction to the example in figure 4, figure 5 depicts a comparison between the standard velocity obstacles method and the method developed in this work. As evidenced, both methods are similar

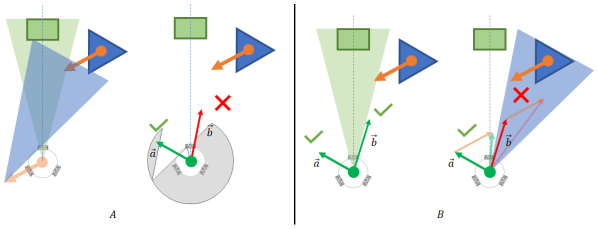

Figure 5: Feasible avoidance control test. Standard velocity obstacles (A) and minimal deviation method (B)

and effective at determining which feasible controls enable the robot to avoid collisions with any obstacles in the near future (command $\overrightarrow{a}$ is found to be a feasible avoidance command and $\overrightarrow{b}$ is not suitable for avoidance). Thus, the minimal deviation is used due to the lower computational load. Finally, the set of feasible avoidance controls (robot commands that ensure avoidance of every obstacle in the scene) is scanned for the control that minimizes the deviation from the robot's current path to ensure a smoother trajectory. If, in any case, a feasible avoidance control is not found, the robot will perform a reverse maneuver, moving in the opposite direction of its current trajectory until a feasible avoidance control is found.

The inability to map the environment and the unavailability of a reference map poses a few limitations to navigation using this algorithm, where despite avoiding collisions with obstacles, it might not be able to reach its intended waypoint and instead become trapped in a trajectory loop. Limitations concerning the sensing system are present as well, where the RADAR sensor will only be able to sense the radial component of a moving obstacle's velocity (the contribution of a moving obstacle's real velocity vector in the direction of the robot). This results in a modified feasible control space which, despite different from the real feasible space, will ensure the avoidance of obstacles.

3.4.5 Odometry

A **dead reckoning** odometry method was developed to estimate the robot's position over time. This method utilizes encoder readings on each wheel to estimate the robot's position based on the kinematics model. Since this method does not accurately reproduce simultaneous rotational and translational movements, a **a fast incremental algorithm** is desireable to minimize these errors. By way of integration of (5) and due to the incremental nature of the algorithm, the odometry model is given by:

$$\begin{bmatrix} X_{t=k} \\ Y_{t=k} \\ \theta_{t=k} \end{bmatrix} = r \begin{bmatrix} \cos\theta_{t=k-1} & \sin\theta_{t=k-1} & 0 \\ -\sin\theta_{t=k-1} & \cos\theta_{t=k-1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & -\frac{\sqrt{3}}{2} & \frac{\sqrt{3}}{2} \\ \frac{1}{3L} & \frac{1}{3L} & \frac{1}{3L} \end{bmatrix} \begin{bmatrix} \Delta\psi_1 \\ \Delta\psi_2 \\ \Delta\psi_3 \end{bmatrix} + \begin{bmatrix} X_{t=k-1} \\ Y_{t=k-1} \\ \theta_{t=k-1} \end{bmatrix} \tag{10}$$

where $X_{t=k}$, $Y_{t=k}$ and $\theta_{t=k}$ is the current robot pose, $\theta_{t=k-1}$ the robot attitude in $t = k-1$, $\Delta\psi_i$ the rotation of wheel i between instants $t = k-1$ and $t = k$, $X_{t=k-1}$, $Y_{t=k-1}$ and $\theta_{t=k-1}$ the previous robot pose.

4. Simulation vs. experimental results

The performance of the simulated and physical implementations were tried in a set of experiments to compare their performances. Since atmospheric conditions of the laboratory as well as the virtual world in which the robot was tried can't be regulated, the sensing system is tried in an enclosed space without the influence of any atmospheric conditions . Static and dynamic trials were designed using simple geometries to simulate both static and moving obstacles. Moving obstacles are constrained to a movement speed lower than the prototype's cruise speed. A *YouTube* playlist was created with demonstrations of the robot navigating test scenarios[1].

4.1. Case study scenarios

A common ground truth is needed to ensure a correct direct comparison of the behaviour of the physical implementation and its virtual counterpart. Two arenas were built in the real world to simulate obstacle courses to be traversed by the robot. These were then scanned using *Qualysis*,

[1] https://www.youtube.com/playlist?list=PLd8pRLUjxQ1GqVXmXKo9Kgd94PkKp8Do

a 3D infrared tracking software and reproduced in *Simulink*'s virtual world. The developed obstacle courses are presented in figure 7 and their virtual counterparts in figure 8.

Figure 6: Qualisys system and mapping environment.

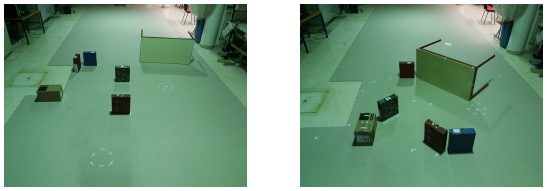

Figure 7: Obstacle configurations for course 1 (left) and course 2 (right) in the real world

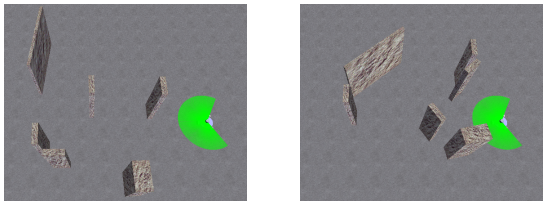

Figure 8: Obstacle configurations for course 1 (left) and course 2 (right) in the virtual world (rotated 90°)

4.2. Simulation vs experimental results

The real and simulated Omni-ANT models were tried in a set of four trials with similar conditions:

- **Trial 1: Course 1 configuration** with the presence of **static** obstacles;
- **Trial 2: Course 1 configuration** with the presence of **static and dynamic** obstacles;
- **Trial 3: Course 2 configuration** with the presence of **static** obstacles;
- **Trial 4: Course 2 configuration** with the presence of **static and dynamic** obstacles.

Aditionally, each trial was performed using distinct active sensor configurations for each trial:

- **LIDAR** sensor;
- **RADAR** sensor;
- **LIDAR and RADAR** sensors.

The goal of the robot is to navigate the environment, avoiding collisions and keeping a safe distance of no less than 0.2m from any obstacles. A designated waypoint located $4m$ in front of the robot's starting point is set to be the endpoint of the trial. The trial is considered to be complete once the robot reaches the vicinity of the designated waypoint or is otherwise unable to compute a feasible trajectory to avoid all obstacles, ceasing movement. The Omni-ANT's and dynamic obstacles' real locations were tracked in each trial. The results of each trial are presented in the following sections. The leftmost figure of each trial represents the location over time of the robot and obstacles. The real, tracked location of the robot in blue, the odometry estimated position of the robot in red and the virtual simulated robot in green. Static obstacles are represented by black rectangles while dynamic obstacles are represented in pink. The minimum relative distance to obstacles over time is also presented in the top right plot and the odometry error in the bottom right of each trial.

4.2.1 Course 1 results

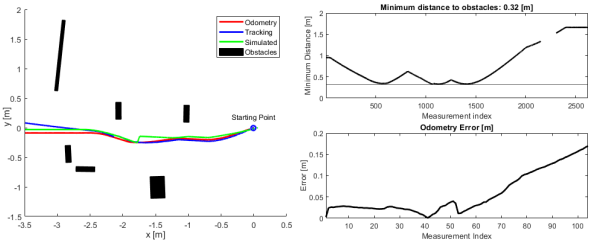

Figure 9: Trial 1, LIDAR and RADAR integration

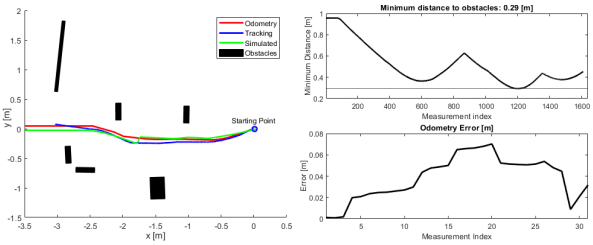

Figure 10: Trial 1, LIDAR sensor active

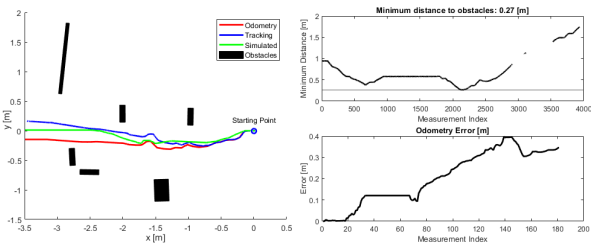

Figure 11: Trial 1, RADAR sensor active

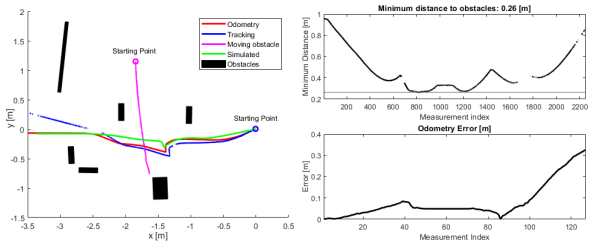

Figure 12: Trial 2, LIDAR and RADAR integration

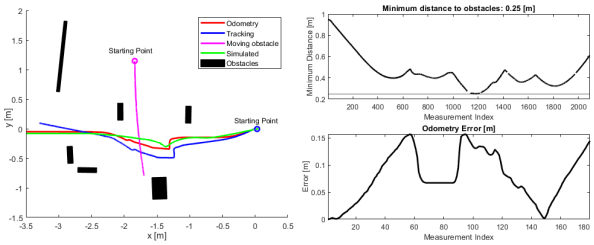

Figure 13: Trial 2, LIDAR sensor active

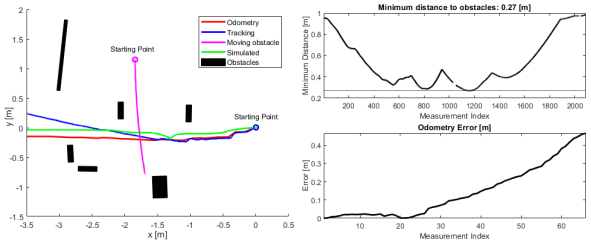

Figure 14: Trial 2, RADAR sensor active

Some observations can be made about the results of trials 1 and 2, run on course 1:

- All the trials were successful, with the robot navigating the course without colliding and keeping a distance above the designated safe distance from all obstacles;

- A number of trials were characterized by the loss of tracking of the robot (figures 10, 12, 13). However, the tracking system failed only in instances where the robot had already exited the course. Since tracking data is most relevant in the obstacle course section of the trial, such irregularities are negligible;

- The sensor fusion approach in trial 2 (figure 12) registered periods of loss of tracking during navigation through the obstacle-ridden section. However, these discontinuities in tracking were brief, with the tracking system being able to quickly localize and track the robot for the remainder of its navigation through the course. No significant data loss was registered, rendering the trial a success;

- The odometry algorithm achieved satisfactory results, estimating the current position accurately as evidenced by its comparison with the real, tracked position. As expected, this method suffered from innacuracies the further the robot travelled (most notably in figures 11-14). This can be attributed to a slight misalignment of the wheels' axes (intended to be mounted at a 120°angle between each pair) and the propagation of these errors in the odometry algorithm;

- The movement of the robot in trials where only the RADAR sensor is active appears to be slightly irregular, with frequent and sudden changes in trajectory (figures 11, 14). This can be attributed to sudden "ghost targets" due to the reflectivity in the environment these trials were performed in, introducing unnecessary and sudden changes in robot trajectory before resuming its previous path. Nonetheless, the robot is able to consistently detect the existing obstacles and avoid them.

- This erratic behaviour is significantly attenuated in trials where sensor fusion is performed (figures 9, 12). This indicates that proximity sensing via LIDAR is more accurate and with less noise in this scenario, resulting in smoother avoidance trajectories which support the previous statement;

- While the simulated trajectories in sensor fusion (figures 9, 12)as well as LIDAR-based trials (figures 10, 13) are similar to their real, tracked counterparts, there is less similarity when compared to RADAR-based trial trajectories (figures 11, 14). This indicates a lesser fidelity of the simulator when simulating RADAR-based obstacle avoidance in real scenarios, where the reflectivity of the environment is a significant factor.

4.2.2 Course 2 results

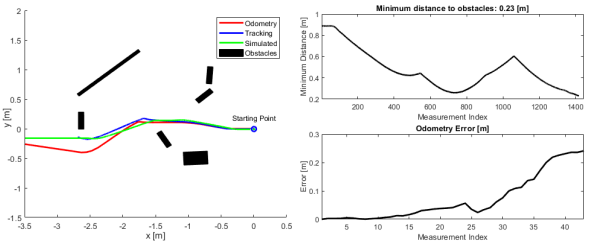

Figure 15: Trial 3, LIDAR and RADAR integration

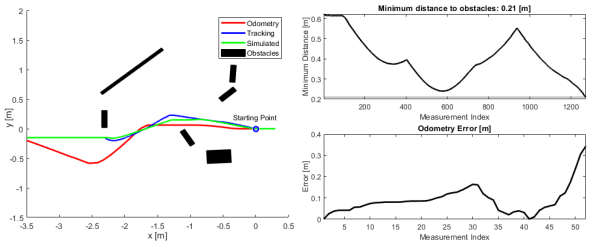

Figure 16: Trial 3, LIDAR sensor active

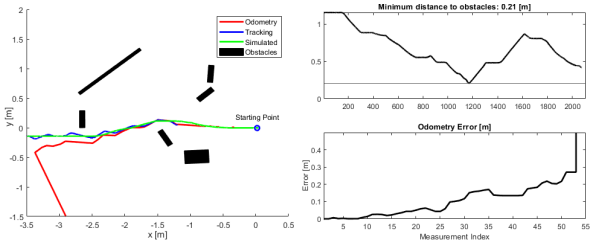

Figure 17: Trial 3, RADAR sensor active

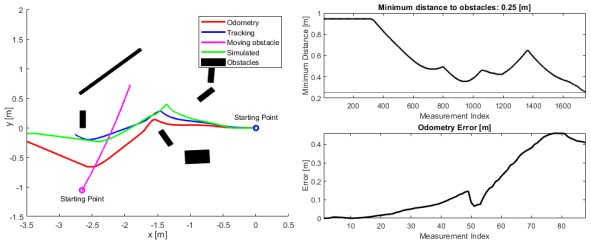

Figure 18: Trial 4, LIDAR and RADAR integration

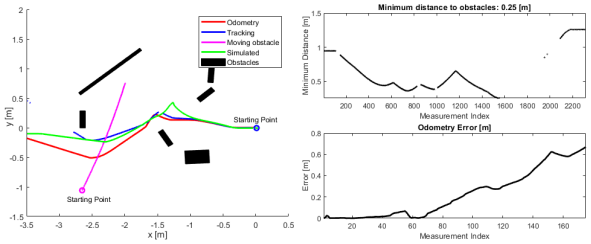

Figure 19: Trial 4, LIDAR sensor active

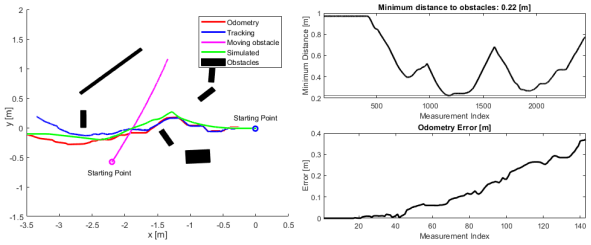

Figure 20: Trial 4, RADAR sensor active

Some observations can be made about the results of trials 3 and 4, run on course 2:

- Once again, all trials were completed successfully, with the robot navigating the obstacle course and arriving at its intended goal;
- Similarly to course 1, course 2's results show that the robot was able to navigate the obstacle course while keeping the minimum safe distance from any obstacles. However, the minimum safe distances registered along the trials on course 2 are lower than those registered on course 1. Such behaviour was expected due to the higher density of obstacles in course 2 when compared to course 1, with the robot having to traverse narrower paths to achieve its goal.
- Loss of tracking of the robot during the obstacle course part of the trials is practically nonexistant, with the tracking system being able to locate the robot during the entirety of its navigation through the maze;
- Unlike the trials run on course 1, some of the trials performed on course 2 registered a loss of tracking during the initial phase of the trial, prior to the robot initiating its navigation during the obstacle course (figures 17, 20). However, this is noticeable prior to the robot initiating avoidance maneuvers, with the system being able to locate the robot during the initial section of the maze. For this reason, the trials are considered a success;
- Once again, the odometry estimation algorithm is accurate at estimating the location of the robot in the initial part of the trials. However, small errors are propagated throughout the robot's path, contributing to an increasing disparity between the real, tracked position and the odometry estimation (most notably in figures 16, 18 and 19). This can be attributed to longer periods of strafing coupled with forward motion (due to the increased degree of avoidance required when compared to course 1) registered inaccurately due to the same inaccuracies in the assembly of the Omni-ANT platform previously mentioned;
- In the case of figure 17, the odometry algorithm fails to estimate the position of the robot towards the end of the trial, with a sudden and significant shift in trend. This phenomenon can be attributed to a failure in the wheel encoders, leading to an incorrect incrementation in the odometry algorithm. However, this failure occurred after the maze portion of the trial, which maintains the trial's validity.

5. Conclusions

A few conclusions can be drawn pertaining the software and hardware implementations and results presented in the previous chapter:

- Velocity sensing (via RADAR) in the context of small-scale autonomous vehicles navigating

static or otherwise non-significantly dynamic environments (dynamic obstacles moving at low velocities) does not offer an improvement in performance when compared to the avoidance system using solely proximity sensing, introducing erratic avoidance trajectories instead;

- Despite the presence of disparities between the tracking and simulated results, the behaviour of the models is similar, where both perform the same general avoidance maneuvers and follow a similar trajectory throughout the obstacle courses in each trial. However, there is a noticeable disparity in behaviour between the solely RADAR-based real and simulated models (due to high reflectivity of real environments). As such, the Omni-ANT simulator model is considered validated for LIDAR-based and sensor fusion trials;

- The virtual simulator can be used as a tool to run experiments in easily generated scenarios (within the validated conditions of sensor fusion or LIDAR-based trials) without risk of damage to any intervenient prior to the real world testing trials. However, it can not be used to predict the absolute outcome of a real trial in similar conditions without thorough testing. The sensors involved in this implementation as well as all the hardware are subject to mechanical failure as well as innacuracies. Analyzing the results from the real world trials, the robot trajectory is not as smooth and continuous as its simulated counterpart. These can be minimized by the implementation of specialized estimation algorithms such as Kalman filtering which, while effective, increase the computational bulk of such a system and is not feasible for implementation in the same conditions as the developed prototype;

- Due to the high degree of modularity that ROS grants to any machine running its software, the prototype can easily be installed in a different ROS network, requiring no additional setup other than setting up the correct IP addresses of the machines involved and ensuring ROS version compatibility between nodes. This framework is an essential part of the system, which not only provided a simplified and efficient baseline for the interaction of the different peripherals of the prototype but also granted a high degree of versatility to the system, allowing for the communication and acquisition of data in several different terminals running ROS, which was a determining factor in the experimental portion of this work.

References

[1] A. Casqueiro, D. Ruivo, A. Moutinho, and J. Martins. Improving Teleoperation with Vibration Force Feedback and Anti-Collision Methods. In L. Reis, A. Moreira, P. Lima, L. Montano, and V. Muñoz-Martinez, editors, *Robot 2015: Second Iberian Robotics Conference. Advances in Intelligent Systems and Computing*, volume 417, pages 269–281. Springer, Cham., 2016.

[2] J. Gonçalves, J. Lima, P. Costa, and A. Moreira. Modeling and Simulation of the EMG30 Geared Motor with Encoder Resorting to SimTwo: The Official Robot@Factory Simulator. *Advances in Sustainable and Competitive Manufacturing Systems*, pages 307–314, 2013.

[3] D. Hutabarat, M. Rivai, D. Purwanto, and H. Hutomo. Lidar-based Obstacle Avoidance for the Autonomous Mobile Robot. In *2019 12th International Conference on Information & Communication Technology and System (ICTS)*, pages 197–202, 2019.

[4] A. Kamann, P. Held, F. Perras, P. Zaumseil, T. Brandmeier, and U. Schwarz. Automotive Radar Multipath Propagation in Uncertain Environments. pages 859–864, 11 2018.

[5] M. Kutila, P. Pyykönen, W. Ritter, O. Sawade, and B. Schäufele. Automotive LIDAR sensor development scenarios for harsh weather conditions. In *IEEE Conference on Intelligent Transportation Systems (ITSC) Proceedings*, pages 265–270, 2016.

[6] S. T. Padgett and A. F. Browne. Vector-based robot obstacle avoidance using LIDAR and mecanum drive. In *SoutheastCon 2017*, pages 1–5, 2017.

[7] Y. Peng, D. Qu, Y. Zhong, S. Xie, J. Luo, and J. Gu. The obstacle detection and obstacle avoidance algorithm based on 2-D lidar. In *2015 IEEE International Conference on Information and Automation*, pages 1648–1653, 2015.

[8] I. Ruiz, D. Aufderheide, and U. Witkowski. Radar Sensor Implementation into a Small Autonomous Vehicle. In U. Rückert, S. Joaquin, and W. Felix, editors, *Advances in Autonomous Mini Robots*, pages 123–132. Springer Berlin Heidelberg, 2012.

[9] Y. Shim and G. Kim. Range Sensor-Based Efficient Obstacle Avoidance through Selective Decision-Making. *Sensors*, 18:1030, 2018.