applied sciences

MDPI

Article

Long-Range Navigation in Complex and Dynamic Environments with Full-Stack S-DOVS †

Diego Martinez-Baselga *, Luis Riazuelo and Luis Montano

Engineering Research Institute of Aragon (I3A), University of Zaragoza, 50018 Zaragoza, Spain; riazuelo@unizar.es (L.R.); montano@unizar.es (L.M.)

* Correspondence: diegomartinez@unizar.es

† This paper is an extended version of our paper published in ROBOT2022: Fifth Iberian Robotics Conference: Advances in Robotics, Zaragoza, Spain, 23–25 November 2022.

Abstract: Robotic autonomous navigation in dynamic environments is a complex problem, as traditional planners may fail to take dynamic obstacles and their variables into account. The Strategy-based Dynamic Object Velocity Space (S-DOVS) planner has been proposed as a solution to navigate in such scenarios. However, it has a number of limitations, such as inability to reach a goal in a large known map, avoid convex objects, or handle trap situations. In this article, we present a modified version of the S-DOVS planner that is integrated into a full navigation stack, which includes a localization system, obstacle tracker, and novel waypoint generator. The complete system takes into account robot kinodynamic constraints and is capable of navigating through large scenarios with known map information in the presence of dynamic obstacles. Extensive simulation and ground robot experiments demonstrate the effectiveness of our system even in environments with dynamic obstacles and replanning requirements, and show that our waypoint generator outperforms other approaches in terms of success rate and time to reach the goal when combined with the S-DOVS planner. Overall, our work represents a step forward in the development of robust and reliable autonomous navigation systems for real-world scenarios.

Keywords: autonomous navigation; dynamic environment; waypoint generation; real-world navigation

check for updates

Citation: Martinez-Baselga, D.; Riazuelo, L.; Montano, L. Long-Range Navigation in Complex and Dynamic Environments with Full-Stack S-DOVS. *Appl. Sci.* **2023**, *13*, 8925. https://doi.org/10.3390/app13158925

Academic Editors: Juan Jesús Roldán-Gómez and Mario Andrei Garzón Oviedo

Received: 8 June 2023
Revised: 25 July 2023
Accepted: 29 July 2023
Published: 3 August 2023

1. Introduction

Autonomous navigation in dynamic environments is a challenging problem that has received significant attention in robotics research. Traditional planners have been successful in navigating in static environments where the robot is aware of the location of all obstacles and can plan a collision-free path to its destination. However, in dynamic environments where obstacles can move, the task becomes much more difficult. Dynamic obstacles have non-zero velocity and can change their position and orientation, which means that the robot needs to take into account their trajectory and predict their future movements to avoid collisions. Failure to do so can result in suboptimal behavior or even catastrophic outcomes.

One approach for solving the problem of autonomous navigation in dynamic environments is the use of the Dynamic Object Velocity Space (DOVS) [1], which provides a representation of the environment in terms of the position and velocity of objects. The Strategy-based Dynamic Object Velocity Space (S-DOVS) planner uses DOVS to plan the robot's trajectory taking into account its kinodynamic constraints and the movement of dynamic obstacles. While S-DOVS has shown promise in aiding robot navigating in dynamic environments, it has limitations in navigating through large maps with unknown environments, avoiding convex objects, and handling trap situations, and is reliant on perfect knowledge of the environment.

To address these limitations, we propose an improved version of the S-DOVS planner which is integrated into a full navigation stack that includes an existing localization system and global planner, a modified obstacle tracker, and a new waypoint generator. Our waypoint generator connects the global planner with S-DOVS, enabling the robot to navigate through large scenarios using known map information while considering robot kinodynamic constraints and the movements of dynamic obstacles. This provides greater freedom to the local motion planner to avoid obstacles while following the global path, and has replanning capability. We compare the waypoint generator to other state-of-the-art approaches and evaluate our system through extensive quantitative and qualitative experiments in simulation and with a ground robot, as seen in Figure 1, to demonstrate its effectiveness in real-world scenarios.

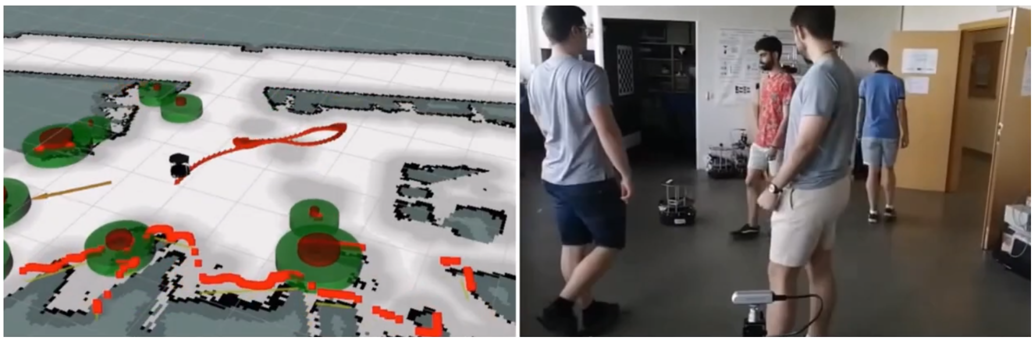
Figure 1. Scenario in which a ground robot must avoid pedestrians while using the map to navigate and reach a goal that has been sent. RVIZ shows what the robot senses, with the tracked obstacles as red and green circles, laser measurements in red, the previous trajectory of the robot as a red line, the costmap in shaded gray and the next waypoint as the yellow arrow.

The contributions of this work are:

- Modification of a state-of-the-art local motion planner used in dynamic environments (S-DOVS) to improve it and enable it to work in real-world scenarios.
- A novel intermediate waypoint trajectory generator designed to be used with motion planners in dynamic environments to enhance their capabilities and enable long-range navigation.
- Adaptation of an existing obstacle tracker for use with S-DOVS, along with a new waypoint generator, a localization system, and an ROS adaptation of all of the components into a full navigation stack.
- Extensive qualitative and quantitative navigation results that show the system's performance in both simulated and real situations.

This paper is an extended version of [2]. Compared to the previous version, the definition of the system and its theoretical development are more comprehensive. The proposed novel waypoint generator is greatly improved here by introducing a spatial horizon to publish the goals that the S-DOVS planner must follow. Moreover, this spatial horizon is variable depending on the importance of the points of the global path. Furthermore, we have refined and improved the criteria necessary for replanning and to avoid situations in which the robot is prevented from reaching its goal. In addition, we have conducted new experiments. In the simulations, we carried out new replanning experiments and used a more challenging scenario. We performed new quantitative experiments to demonstrate our waypoint generator, then compared it with other the state-of-the-art approaches and with our previous generator presented in [2]. Finally, we included a new real-world experiment that explicitly required long-range navigation to be combined with dynamic obstacle avoidance. The discussion section is expanded to include these new results. The repository for this the work can be accessed at https://github.com/dmartinezbaselga/dynamic_waypoint_generator (accessed on 29 July 2023).

The paper is organised as follows. Section 2 details the background of the work, including the related work in Section 2.1 and the basis of the S-DOVS planner in Section 2.2.

Section 3 analyzes the methodology followed to address the problem and the main components of the proposed system. Section 4 presents a complete evaluation of the proposed method, with quantitative and qualitative results in both simulation and the real world, and discusses the obtained results. Finally, Section 5 concludes the work.

2. Background

In this section, we provide an overview of the current state of navigation in dynamic environments, the need for further study to address the challenges it faces, and a brief summary of the model that is the basis of this work.

2.1. Related Work

The complexity of autonomous navigation increases when the environment changes. Conventional global planning algorithms such as RRT [3] or A* [4] and their subsequent improvements [5,6] compute a plan that the robot should follow regarding the map information. Then, a local planner uses the information from the robot's sensors to consider the changes in the environment and follow the global plan. An example of a local planner is the Dynamic Window Approach (DWA) [7], which iteratively computes the velocity of the robot considering its kinodynamic constraints, the sensed distance to obstacles, and the position of the goal. This approach was extended in [8] to consider high-velocity motion. The problem with the DWA is that it does not consider the velocity and trajectories of the obstacles, only taking into account their current position. Alternative methods to control the movement of the robot have been proposed. For instance, the robot's movement can be represented as the outcome of potential forces, such as the artificial potential fields proposed in [9]. Other methods model the environment as a spring–mass system (i.e., an elastic band), as in [10].

Dynamic environments pose a significant challenge for navigation planning, as they require real-time adaptation to constantly changing surroundings. Planners have been developed to address this issue, such as those presented in [11–13]. Several of the previously mentioned approaches for navigating in static environments have been extended to adapt to changing environments. The DWA can be improved by predicting the future trajectories of obstacles using holonomic controllers [14] or by adapting the DWA parameters for each situation using a deep neural network [15]. The Time Elastic Band (TEB) approach [16] proposed modeling the changes in the environment over time as elastic bands.

Including dynamic obstacles in the model requires a more complex representation than simply using their positions, as in the static case. A wide range of methods use the velocity of the obstacles in the model to achieve better performance. Two models that have been widely used in the literature are the inevitable collision states (ICS) and the Velocity Obstacle (VO). ICS, introduced in [17], uses the dynamics of the system and the obstacles to compute future states where a collision is inevitable. This model considers the time it would take for the robot to stop or avoid the obstacle, which can be essential in fast-changing environments. On the other hand, VO, introduced in [18], is defined by the trajectories that the dynamic obstacles have in time. It computes those velocities that could cause a collision with the robot, after which the velocities that are not inside the velocity obstacles are available to be chosen by the robot. The VO model ensures that the robot dynamics are respected, making it a suitable choice for scenarios with complex obstacle dynamics.

The VO was extended in [1] with Dynamic Object Velocity Space (DOVS), representing the dynamism and future of the environment. A local planner called S-DOVS that includes the robot's kinodynamic constraints was developed. The S-DOVS strategy-based planner identifies various situations based on the relative variables between the robot and moving objects, then applies a different action for each situation in order to balance the time to goal and robot safety.

In recent years, motion planning in dynamic environments has been tackled by learning approaches such as reinforcement learning, as demonstrated in various works, including [19–21], as well as by [22,23], which also incorporated the DOVS model. However, these methods face limitations due to the complexity of the real world and the large number of variables involved. Although they attempt to address the limitations of model-based approaches, there remains a large gap between the simulators used for training and real-world scenarios. Simulators do not precisely reproduce the physics and randomness of real environments, resulting in biased interactions. Furthermore, such approaches assume that robots do not have acceleration constraints and may choose any velocity at any time, which is not practical in real life, and often assume that the robot is holonomic or has perfect knowledge of the environment, which is rarely the case in reality. Several works have analyzed and explained the limitations of Deep Reinforcement Learning (DRL) in the real world [24,25], including inability to deal with environmental constraints, the limitations involved in gathering real-world training data, and partial observability. In this work, we use a modified version of S-DOVS as the local planner to avoid the limitations of reinforcement learning.

Navigating in highly dynamic environments requires a local planner with the flexibility to reactively avoid obstacles, which may cause deviations from the original path. To enable long-range navigation, a global planner must be integrated into the system. The work of [26] has demonstrated promising results by subsampling the original path generated by the global planner. In Refs. [27,28], the authors further improved upon this approach by introducing spatial horizon subgoals and replanning capabilities. Our proposed work presents a waypoint generator that utilizes subsampling and other criteria to decrease the path's density, as well as replanning capabilities and spatial horizon subgoal generation for the areas where the intermediate plan is sparse.

2.2. Dynamic Object Velocity Space (DOVS)

The Dynamic Object Velocity Space (DOVS) [1] is a model used in robotic autonomous navigation to compute safe motion commands within a time horizon. It is based on the velocity space (VS), which is defined as the set of velocities a robot can reach while respecting the constraints of the maximum and minimum velocities. DOVS abstracts the dynamic environment by representing safe and unsafe velocities that the robot can choose in every sampling period. The model defines the Dynamic Object Velocity (DOV) as the set of velocities that can lead to a collision with moving obstacles within a time horizon. The DOVS uses this information to compute safe motion commands for the robot.

To represent moving obstacles, the robot's size is reduced to a point and the obstacles are enlarged by the robot radius. The collision band is then defined as the area swept by the obstacle while it moves. The intersection of the robot's trajectory with the collision band defines the possible collision points, as seen in Figure 2. From these points, the maximum and minimum velocities that lead to collision are computed for every obstacle, then a dynamic window is used to select safe velocities within the robot's kinodynamic constraints for the next step. The DOVS is graphically represented in Figure 3a, with safe velocities shown in white, unsafe velocities in dark, the dynamic window represented with a green rhombus, velocities that lead to the goal following a circular path with a magenta line, and safe velocities that lead to the goal with a green line. The robot can choose from a set of only eight possible velocities, which include: the current velocity of the robot (1); the velocities of the extremes of the rhombus, which are the velocities reachable in the next control period at maximum linear or angular acceleration from the current velocity (2, 3, 4, 5); the minimum and maximum velocities that lead to the goal (6, 7); and the velocity that leads to the goal when using the current linear velocity (8). These actions are represented in Figure 3b, with the numbers having the meanings noted above. In our original work, a planner (S-DOVS) was developed by which the robot selects a velocity based on predefined strategies designed to balance safety and maximum robot velocity.

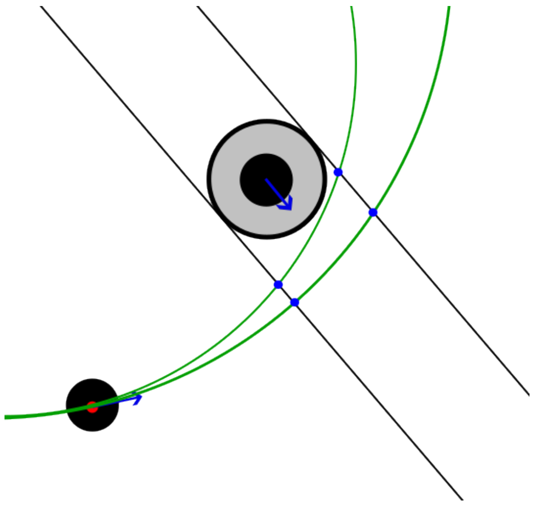
Figure 2. The figure shows how the robot is reduced to a point (red point); the obstacle is enlarged with the robot radius (gray), its collision band is drawn for its current trajectory, and the intersection points between two possible trajectories (green) and the collision band are computed in blue.

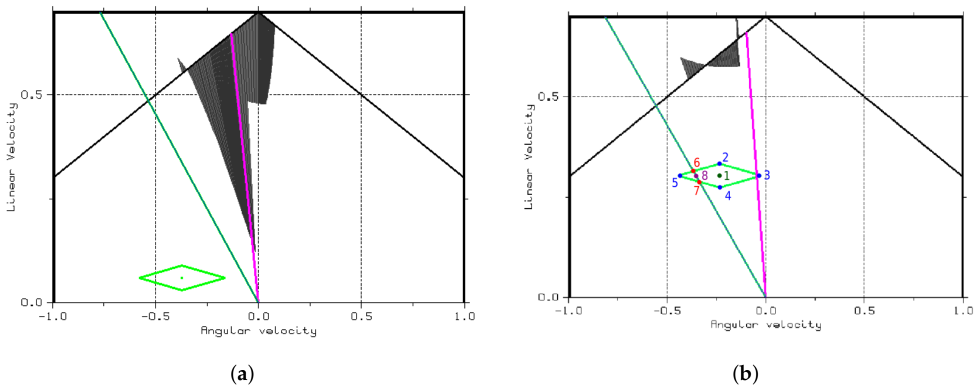

Figure 3. Graphical representation of the DOVS model and its action space: (**a**) DOVS with forbidden velocities and (**b**) representation of the action space.

3. Approach

This section outlines the design and analysis procedures followed in this work, the methodology, and the main components of the designed system.

3.1. Navigation Stack

The algorithm developed in the original program of the work presented in [1] has been adapted in this work to make it usable on a real robot by using the Robotic Operating System (ROS), which is an open-source framework for robotic applications. A representation of the ROS navigation stack is presented in Figure 4, with the main ROS components and the waypoint generator integrated before the motion planner. The stage simulator [29] integrated into the ROS was used, which can realistically simulate the robots' dynamics and kinematics as well as their sensors and the world's physics. Two types of agents were developed: active agents, which use the S-DOVS algorithm for navigation, and passive agents, which act as dynamic obstacles.

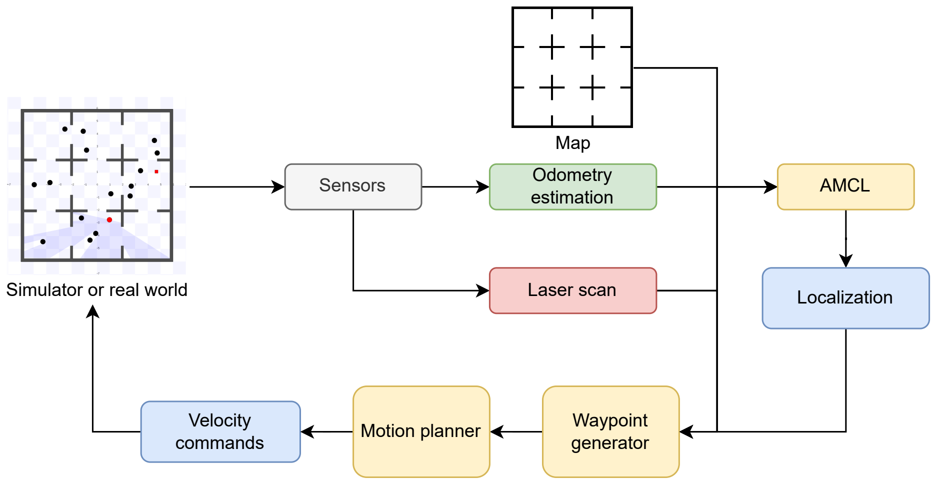

Figure 4. Representation of the ROS navigation stack with a waypoint generator.

The modified simulator detects collisions and allows agents to be teleported to start a new episode without restarting the simulation. The launching of agents and the environment is performed with an automatic script. This makes conducting qualitative experiments and comparative navigation systems possible in maps such as the ones shown in Figure 5a,b. The proposed modifications enabled the program to be usable on a real robot, making it more practical and realistic.

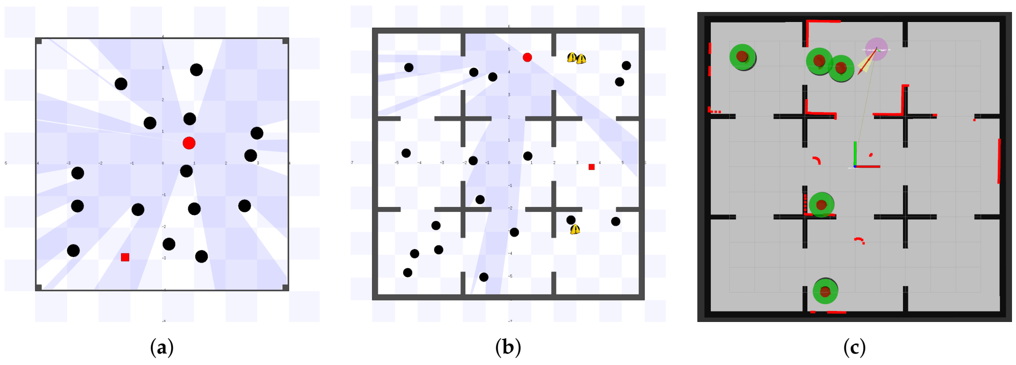

Figure 5. Two maps used for quantitative experiments: (**a**) Single room map, (**b**) Multi-room map. In (**c**), RVIZ shows the AMCL robot estimated pose with a red arrow, the covariance of the estimation in magenta, the laser scans in red, and the tracked obstacles in red and green circles.

To navigate in a scenario, a robot needs to know its location; however, unlike in simulations using a centralized program, in the real world the robot's position is not known. Therefore, the active agents in our study use sensor data such as 2D LIDAR and wheel encoders to estimate their location. To achieve localization, we employ Adaptive Monte Carlo Localization (AMCL) [30] from the ROS navigation stack, which involves updating and sampling a large number of particles based on laser scans and the odometry estimation to converge on the robot's current position. The AMCL requires a static map of the environment, laser scans, an initial position, and encoder measurements as inputs, and outputs the robot's position in the map frame. An example of the AMCL estimated pose of the robot is seen in Figure 5c.

3.2. Obstacle Tracker and S-DOVS Adaptations

The DOVS model uses the current position, radius, and velocity of obstacles to predict their trajectory and collision band. The information must be gathered from the LIDAR sensor. The approach presented in [31] is utilized, which uses laser measurements grouped to form circles or segments, while an Extended Kalman Filter is used to track measurements in time. The approach is extended such the orientation and angular velocity of obstacles are computed using the direction of the obstacle's trajectory and tracking of the change in orientation at a high frequency (25 Hz). Several factors such as spurious measurements and occlusions make these measurements erroneous or unreliable in particular moments, specially in the real world. Assuming that dynamic obstacles as people or robots tend to follow smooth trajectories with no sudden velocity changes, we use the median value over the few last estimations to reduce the perception errors. The time interval used for this median filter (applied for both linear and angular velocities) is configurable and set to 0.4 s during the experiments. In this work, we use this approach to track the size, position, linear velocity, orientation, and angular velocity of obstacles over time, enabling the DOVS model to predict obstacle behavior accurately. The obstacles tracked by the robot and the localization estimated by AMCL may be seen in Figure 5c.

We introduced modifications to the original S-DOVS algorithm. The modified S-DOVS model improves on the initial model by including segment obstacles and removing strategies that lead to trapping situations or small margins to react. The modified model calculates forbidden velocities for static obstacles differently, considering only trajectories that have a distance traveled greater than the distance required for the robot to brake multiplied by a safety factor, following the equation

$$v_{safe} \leq \sqrt{2 \cdot a_{max} \cdot dist}, \tag{1}$$

where a_{max} is the maximum linear acceleration of the robot and $dist$ is the distance from the robot to the static obstacle following a particular trajectory radius. Thus, the modified model calculates forbidden velocities for static obstacles less restrictively, allowing the robot to take more velocities, leading to greater maneuverability and easier obstacle avoidance.

The inclusion of segment obstacles in the DOVS model improves accuracy and eliminates the curvy nature of the initial obstacle model, which needs to fuse circle obstacles in order to consider walls. The modified model saves the points of the extremes and forbids any velocity leading to a trajectory that intersects with a point between the extremes. Additionally, the behaviors of certain strategies have been changed. When there are no safe velocities in DOVS, the robot applies maximum deceleration to avoid risky situations that could lead to running over pedestrians; the previous behavior was to accelerate towards the closest previous gap. A conservative velocity threshold is set when considering whether an obstacle is static or dynamic in order to account for estimation errors. Therefore, an obstacle is considered static if its sensed linear velocity is lower than a threshold, instead of expecting it to have a complete null velocity.

3.3. *Trajectory Waypoint Generator*

Reactive planners designed to navigate in dynamic environments use information about obstacles and their trajectories to avoid obstacles and reach a goal. They are typically designed to navigate in scenarios where the initial position and goal are connected by a straight line. However, to navigate in the real world it is useful to have access to a known map. The robot may encounter large walls that could surround it, or it may become trapped. A global planner can be used to overcome such problems, enabling long range navigation. While we used S-DOVS as the local planner here, our approach could be used jointly with other deep reinforcement learning-based or model-based planners designed to navigate in dynamic environments. S-DOVS only requires one point as a goal, which can be far from the initial pose, and plans the trajectory in advance using estimated obstacle velocities. Moreover, the goal should be far enough to ensure that the local planner has the freedom to perform collision avoidance maneuvers. Additionally, the original static map may be updated while the robot navigates. If the robot senses static obstacles, it should take them into account in the new global map. For example, the robot may attempt to enter a room through a door, find the door closed, and replan to find an alternate path.

This work uses the A* implementation of the ROS standard move base as a preliminary global planner, computing an initial global path. The map is kept up to date, with the global cost map taking newly seen static obstacles into account and the A* algorithm planning accordingly. The cost map's inflation ratio is set to twice the robot's size, ensuring that the S-DOVS has adequate space to choose its direction.

In our approach, we consider a spatial horizon d_{ahead} to send the next waypoint to the local motion planner. When the distance from the robot to the current subgoal is less than d_{ahead}, the next subgoal is followed, allowing the motion planner space to maneuver and deviate from the original path in order to avoid dynamic obstacles. Therefore, the waypoint generator saves the initial global plan computed by A* as $path_g$ and keeps sending waypoints regarding the spatial horizon.

Nevertheless, there is a problem with this approach. If d_{ahead} is too low, the motion planner may not maneuver properly, as the robot may not deviate from the global plan. If d_{ahead} is too high, the motion planner may not benefit from the map information and may not avoid convex obstacles. To solve this problem, our approach computes an auxiliary path $path_{aux} \subseteq path_g$, which selects important waypoints that should not be overlooked. The value of d_{ahead} is high for the points of $path_g$ that do not belong to $path_{aux}$ and low for the ones that belong to $path_{aux}$; thus, that the robot explicitly reaches the points that make it navigate through sharp corners or rooms.

To compute $path_{aux}$, the waypoint generator performs a downsampling operation, selecting one point per square meter, ensuring that the path followed is sparse. Intermediate points in a straight line are removed, as they do not provide any extra information for motion planning and reaching them would result in suboptimal performance compared to computing the velocities required to reach the final point. For each set of three points, if they are in the same line, the point in the middle is removed. This is determined by testing whether the slope of the line passing through the first and second points is similar to that of the line passing through the first and third points, using the following equation:

$$\frac{y_1 - y_2}{x_1 - x_2} \approx \frac{y_1 - y_3}{x_1 - x_3} \implies |(y_1 - y_2)(x_1 - x_3) - (y_1 - y_3)(x_1 - x_2)| < \epsilon, \tag{2}$$

The robot keeps track of a costmap with the area occupied by the static obstacles, which is updated with the information received by the sensors. It is initialized using the static map of the environment. This costmap is dinamycally updated as the robot navigates and senses the environment. On the one hand, if the LIDAR measurements sense a static obstacle that had not been previously considered, it is added to the costmap, so that the new proposed trajectories and waypoints do not intersect with the obstacles. On the other hand, if no obstacle is detected in a place where an obstacle was supposed to be, the costmap in that positions is cleared, so that the robot may navigate there. This functionality allows the robot to, for example, detect whether a door is open or closed and to choose that path or an alternative one. If there is not any possible path at all, the waypoint generator does not generate any point. In addition, this algorithm assumes that the static map is known in advance, but it may work when there is a significant difference between the map and the environment. The static map is used to initialize the costmap, but, as discussed before, this costmap is dinamycally updated if the environment is not the same as the costmap. Thus, as the robot navigates in the environment, the costmap will be more precise and the performance will improve. Furthermore, the approach could be easily adapted when the map is unknown, by replacing the map and AMCL localization combination with a Simultaneous Localization and Mapping (SLAM) algorithm, such as Gmapping [32], which is publicly available as a ROS node, to build the whole map while the robot navigates.

We provided replanning capabilities to the waypoint generator to ensure that a new plan is generated if the robot is too far away from the global planner's plan. This is detected by measuring the distance from the robot to the segment $path_{aux}$ joining the last point reached and the next to be reached. In addition, a new plan is produced if the robot is stuck. The criteria followed to decide whether the robot is stuck is to check whether its linear velocity is null for some number of timesteps. Moreover, replanning is conducted the first time the robot reaches a velocity lower than the threshold. If the robot keeps the same low velocity, it does not perform additional replanning, as there are situations in which low velocities are necessary to avoid collisions. This is intended to avoid situations in which the robot is completely stuck before they can occur. It is important to note that the new plan may be similar to the previous one. Therefore, if the robot reaches a low velocity due to dynamic obstacles which are not included in the cost map, the new plan sent will not influence the current trajectory of the robot. Furthermore, if it deviates from the original path due to dynamic obstacles, the new plan recomputed would be similar to the previous one, except now accounting for the new starting position of the robot in order to adapt the waypoints to it.

With this approach, the robot follows points that adapt to the S-DOVS trajectory and replans when the map is updated with new static obstacles that block its way. A high-level algorithm of the design of the waypoint generator is shown in Algorithm 1.

Algorithm 1: Waypoint generator

```
Input: Global path $path_g$, robot distance to goal $d_{goal}$
Output: Next subgoal $g$
Data: $d_{long}$,$d_{short}$                /* Long and short lookahead distances */
$path_{aux} \leftarrow subsample\_and\_remove\_straight(path_g)$
$g \leftarrow get\_first\_point(path_g)$
while $g \neq get\_last\_point(path_g)$ do
    if $g \in path_{aux}$ then
        $d_{ahead} \leftarrow d_{short}$
    else
        $d_{ahead} \leftarrow d_{long}$
    end
    if $d_{goal} \leq d_{ahead}$ then
        $g \leftarrow get\_next\_point(path_g)$
    end
    if $robot\_is\_stuck() \vee robot\_deviates(path_{aux}) \vee robot\_low\_velocity()$ then
        $path_g, path_{aux} \leftarrow generate\_new\_path()$
    end
end
```

In Algorithm 1, the *robot_is_stuck*() function checks whether the robot is completely stopped due to being trapped. It has a timeout to send a new plan every few time steps until the robot can move again due to a change in the environment. The function *robot_deviates*() checks whether the robot is too far from $path_{aux}$; if this is true, a new plan is generated that better fits the robot's current position. The function *robot_low_velocity*() returns a true value if the robot's velocity is below the threshold, updating its state in an internal variable to check whether this is the first time this has occurred. If it passes the threshold again in subsequent timesteps, the function is returned as false due to the saved internal state. When the robot's linear velocity surpasses the threshold again, the internal state is reset and the function may be returned as true if the condition is fulfilled.

4. Experiments and Discussion

This section analyzes the experiments we conducted to test and validate our system. Videos of every quantitative and qualitative experiment are provided in Supplementary Video S1.

We test and compare the use of the complete integrated system with the S-DOVS planner on its own in different maps, our proposed waypoint generator with others of the state of the art and extensively evaluate the system in both simulation and the real world. Note that we don't compare the S-DOVS planner with other local reactive planners such as RL-DOVS [22], as developing a local planner is not the goal of the paper, and any other local reactive planner could be used in the system instead of S-DOVS. Common reactive planners, by themselves, are limited to handling only a few meters of navigation within a single room in the real world. As a proof of concept, our experiments demonstrated paths of up to 15 meters in length, both in simulations and the real world. However, it is not limited to this distance, as the waypoint generator enables us to generate paths of any desired length.

4.1. Quantitative Experiments

Our quantitative experiments aimed to evaluate the effectiveness of the global planner in helping S-DOVS to avoid collisions by utilizing the previous static map information and adding new static obstacles to the cost map that are avoided with the global plan.

The first experiments compared the use of the waypoint generator in a simple $4 \times 4\ \mathrm{m}^2$ room as the scenario and in a map with multiple rooms, as shown in Figure 5. Static and dynamic obstacles were randomly placed, and the robot had to navigate to a goal. Quantitative experiments were performed to measure the differences between navigating with the S-DOVS by itself and the S-DOVS with the waypoint generator in sets of 200 scenarios for any number of obstacles in the range of 1 to 15.

The results of the experiments are presented in Table 1. The robot can reach the goal in more episodes when using the complete navigation stack, even in the map with only one room. This is because obstacles on the way to the goal can remain static; the global planner adds them to the cost map, assisting the S-DOVS. In the nine-room map the benefits are clear, as the S-DOVS has to reactively navigate through the rooms and is unable to, falling into trap situations from which it is not able to escape. The intersections of the walls are convex obstacles that a reactive planner may not be able to avoid by itself. Nevertheless, the intermediate goals sent by the waypoint generator help S-DOVS to avoid the walls properly. The approaches perform similarly in terms of time on both the single-room map and multi-room map, although the full system is evidently safer and more successful. The time comparison is somewhat unfair, as only the successful scenarios are accounted for in the metrics, and longer scenarios were almost always solved by only the system with the waypoint generator, meaning that its mean time to reach the goal is much higher.

Table 1. Comparison of success rates ($\frac{\text{number of success complete system}}{\text{number of success S-DOVS only}}$) and mean time to reach the goal ($\frac{\text{mean time complete system}}{\text{mean time S-DOVS only}}$) in a map with a single room and a map with nine rooms for the same 200 random scenarios for each number of obstacles.

Map	Metric	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Single	Success	1.02	1.12	0.99	1.05	1.02	1.19	1.19	1.02	1.16	1.06	1.23	1.06	1.06	1.29	1.39
	Time	1.02	1.01	0.97	0.99	0.98	1.09	1.05	1.00	1.06	1.08	0.93	1.03	0.89	1.06	1.11
Multi	Success	3.09	2.89	3.41	2.84	3.09	4.16	4.11	3.08	3.67	3.21	3.73	2.86	3.93	3.52	3.53
	Time	1.09	1.32	1.11	1.07	1.21	1.08	1.13	1.07	0.95	1.11	1.25	1.24	1.10	1.10	1.11

Our approach proposed for the waypoint generator was compared with other state-of-the-art approaches. We randomly generated 200 scenarios with 5, 10, and 15 obstacles in the multi-room environment, and gathered success and time metrics using different planners, plotting the results in Figure 6. The approaches compared here used several alternatives: using only the final goal and no intermediate waypoints; a simple subsampling of the original path, as in [26]; using only space–horizon goals, as in [27]; our previous approach in [2], which did not use importance-varying spacial horizon goals, but included selection of the points of $path_{aux}$ of Algorithm 1; and our improved approach presented in this work. We included replanning with the same criteria in every method; thus, the only difference was in the selection of the waypoints and the look-ahead distance. The scenarios in which every approach led to a collision were discarded, as the randomness involved in scenario generation may have made them unsolvable.

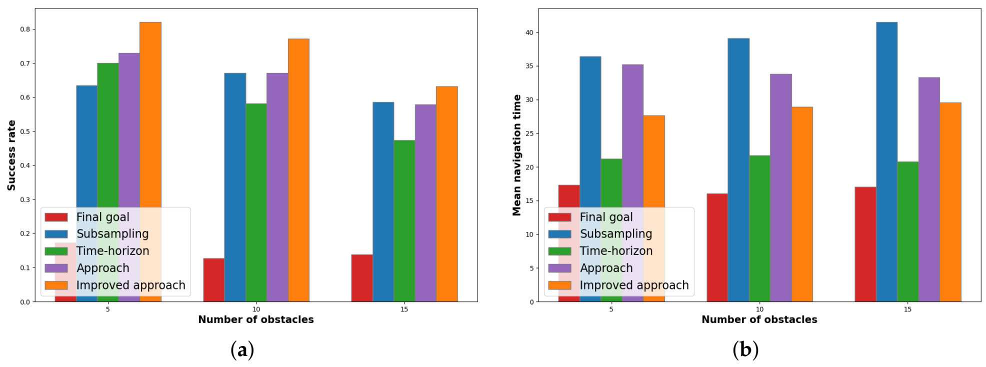

Figure 6. Comparison of the performance of different waypoint generators: (**a**) success rates and (**b**) mean time to reach the goal.

The experiments showed that our improved approach outperforms the original version on both metrics, showing the benefits of the added modifications. The proposed approach outperforms the other alternatives as well, as it combines their various attributes with several additional features. In terms of success rate, the improvement is clear. The mean navigation time was only gathered for the successful episodes in every method, meaning that the navigation time metrics are influenced by the success rates. On the one hand, when a method has a low success rate this means that it fails to find a solution in a significant portion of the trials. Thus, the episodes in which it reaches the goal are the simplest ones, which leads to shorter average time to reach the goal. On the other hand, planners with high success rates may spend more time on difficult episodes, leading to higher average time to reach the goal for successful trials. Even with this caveat, our improved approach shows lower navigation times than the two approaches with the most similar success rates. In addition, the results show that the improvements presented in this work clearly improve on the approach presented in our original conference paper [2].

4.2. Qualitative Experiments

We tested the system in a variety of simulated scenarios. In Figure 7, a scenario was used in which the system was forced to replan. The robot was placed in the middle of the central room of a multi-room environment and the goal was placed in a room next to it blocked by three static obstacles. The robot senses the blocked route and adds it to the cost map, then the S-DOVS planner reduces its velocity, as it may not be able to avoid the three obstacles and the walls (i.e., the robot senses that it may become trapped). After the robot reduces its velocity and before it becomes completely stuck, the waypoint generator computes a new plan that the robot must follow to be able to reach the goal. In addition to this scenario, more experiments were performed in a newly designed environment to test the robot's navigation in a scenario with differently shaped rooms, narrow corridors, and moving obstacles, as seen in Figure 8. This scenario poses challenges to the robot that it has not experience before; nonetheless, the system is able to navigate successfully.

The system was tested on a Turtlebot 2 platform equipped with a Hokuyo 2D-LIDAR sensor and an NUC with an Intel Core i5-6260U CPU and 8 GB of RAM. The replanning capability in a real-world scenario was tested in a room connected to a corridor with two doors. The robot was sent to a goal, then encountered a closed door on the original path. The system replanned and navigated by an alternative path through the other door, as shown in Figure 9.

The robot was then tasked to navigate autonomously in a room with human occupants acting as dynamic obstacles. The robot successfully completed the task by using the whole integrated system. The goals were dynamically placed in the corners of the window, and the robot needed to avoid collisions with the humans to reach the goals. An image of this experiment can be seen in the previously shown Figure 1. In addition, an experiment was conducted in which the robot was forced to navigate through the corridor, enter the room, exit the room, and navigate again through the corridor while avoiding pedestrians in order to explicitly show a combination of the global planner with the S-DOVS. The intermediate waypoints sent by the waypoint generator are shown in the visualization in Figure 10. Videos of every described experiment can be found in Supplementary Video S1.

The successful integration of various components, including the local planner, global planner, waypoint generator, localization system, and obstacle tracker, was demonstrated in both simulated and real-world experiments, underscoring the effectiveness of this integrated approach. The S-DOVS local planner efficiently computes safe motions while maintaining high velocities whenever possible, enabling it to navigate through moving obstacles while adhering to the kinodynamic constraints of the robot. This aspect sets it apart from existing state-of-the-art systems, which often overlook such constraints. Moreover, S-DOVS adopts a strategy-based planning approach, significantly reducing the sim2real gap by eliminating the need for real-world data during training, which is a substantial challenge in the field of robotics.

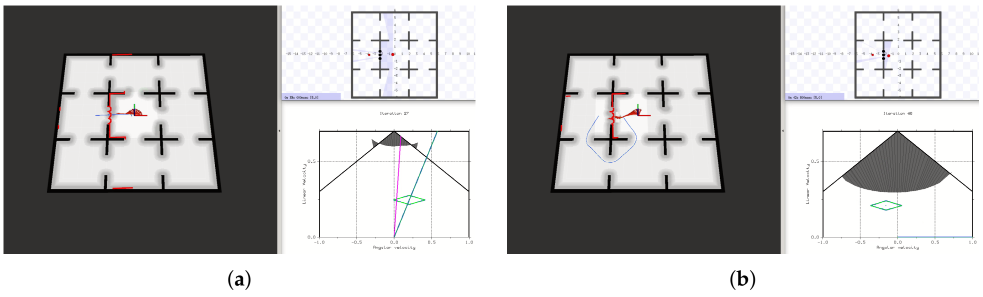

Figure 7. Experiment in which the robot senses that the path is blocked and computes a new path: (**a**) Before replanning and (**b**) after replanning. In (**a**), the robot is accelerating to reach the goal by moving towards the blocking obstacles. The cost map is updated with the three sensed obstacles that are blocking its trajectory. After reducing its velocity to avoid collision with the three static obstacles and the walls, a new path is sent that follows a new trajectory through free space, as seen in (**b**). This is the new plan the robot follows to reach the goal.

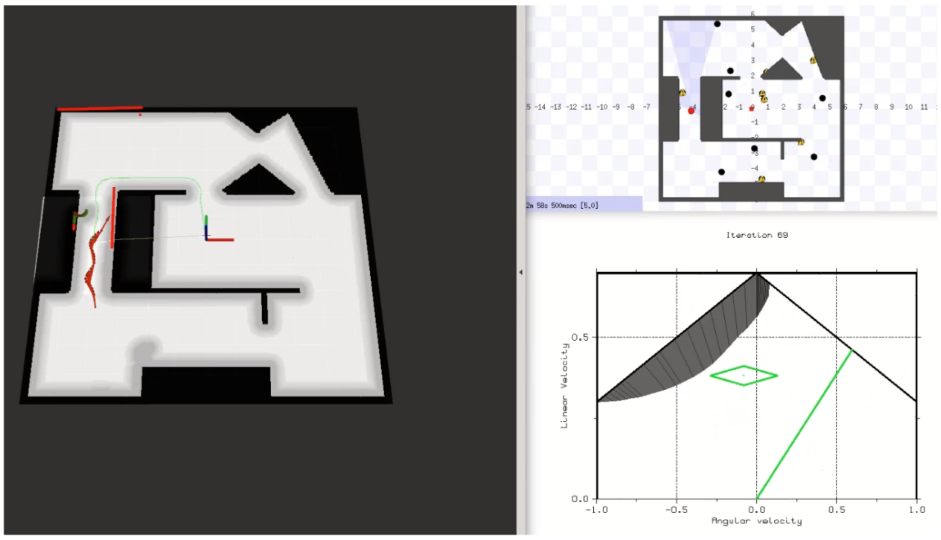

Figure 8. Here, the robot is navigating in a different map among moving obstacles; it must face various challenges to reach the goal in the shortest time possible. The goal is located at the square red point, the green line is the global trajectory planned towards the goal in the workspace, and the velocity space (VS) depicts the DOVS at the instant in which the robot is at the arrow's location in the workspace. The dark area in VS represents the velocities leading to collision with the right wall, and the green line shows the velocities to move the robot towards the next waypoint on the planned trajectory from the current velocity, i.e., the center of the rhombus.

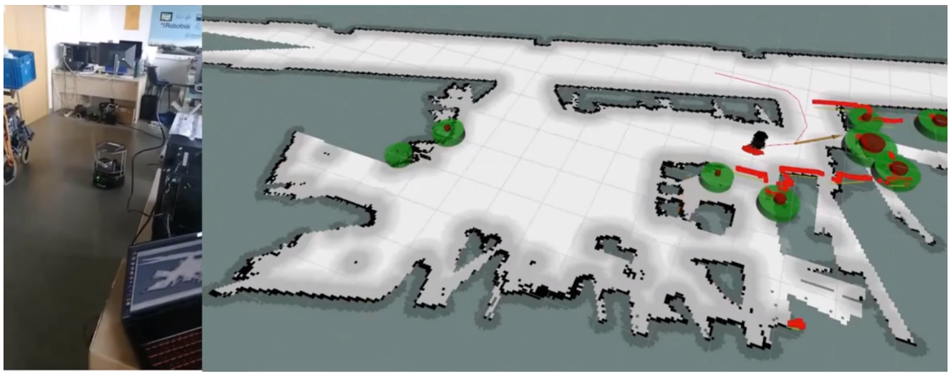

Figure 9. The moment before the robot senses the closed door, adds it to the cost map, and computes a new path. In the subsequent time steps, it detects that the way is blocked and computes a path through the other door, which is open.

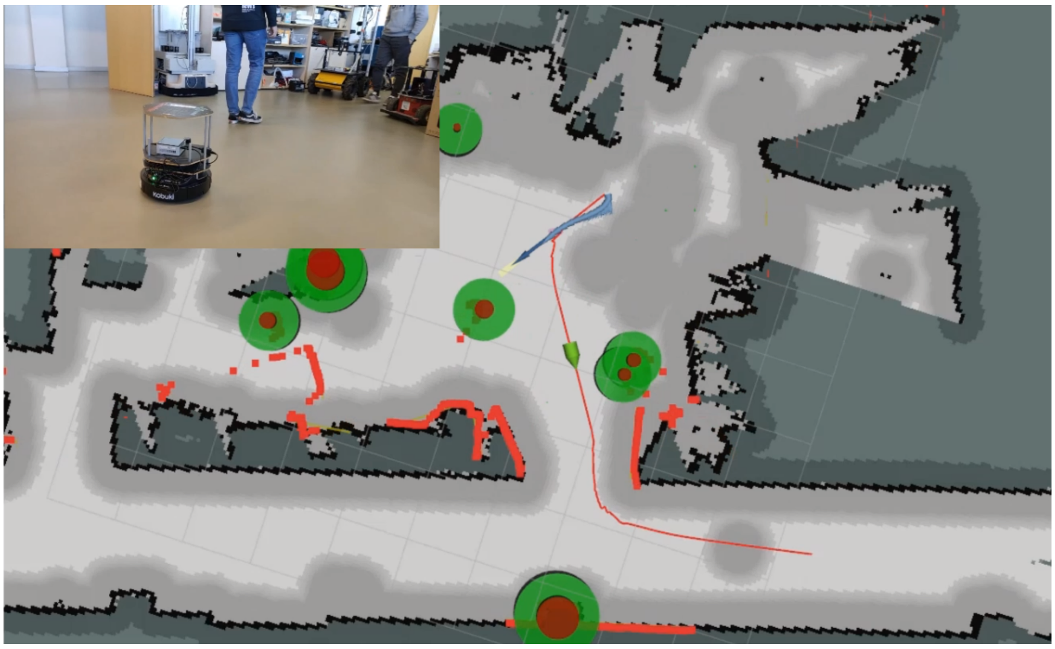

Figure 10. Experiment in which the robot has to avoid collisions with dynamic obstacles and follow intermediate waypoints to reach the goal. The path denoted in Algorithm 1 as $path_{aux}$ is shown in red, while the green point in $path_{aux}$ is the current waypoint that the robot is following.

The waypoint generator is a critical component, as it allows the robot to navigate through multiple rooms and scenarios with different shapes while avoiding static obstacles and blocking situations, which can frequently occur when using only a local planner. Additionally, the proposed design allows the local planner to fully exploit its capabilities by allowing it freedom to maneuver. Our approach outperforms other state-of-the-art approaches in terms of success rate and time to reach the goal, and is able to work correctly in real-world situations.

5. Conclusions

This work presents an approach for autonomous navigation in complex and dynamic environments. The proposed approach integrates the strategies-based S-DOVS method designed for dynamic scenarios as the local planner, and uses a waypoint generator for long-range navigation. S-DOVS takes into account the robot's kinodynamics and the velocity and position of obstacles to plan the robot's motion within a time–space horizon, avoiding dynamic obstacles while balancing safety and ensuring high robot velocities whenever possible. The waypoint generator generates a set of waypoints based on a static environment map, significantly improving the success rate and time to reach the goal with respect to the case in which the generator is not used. Multiple experiments in simulation and real-world scenarios demonstrate the effectiveness of our approach. Our system is able to navigate dense and dynamic scenarios with different shapes while avoiding both static and dynamic obstacles, performing better than the state-of-the-art techniques used for comparison. The results of our experiments demonstrate the effectiveness of this approach for use in real-world applications and populated scenarios.

Future work will include exploring the use of different local planners based on deep reinforcement learning approaches to further improve the performance of the system. Moreover, the work could be extended to 3D navigation for use with autonomous drones that are able to navigate in the presence of people or other drones moving in the vicinity. Furthermore, approaches as [33,34] could be used to have the actual shape of obstacles on the environment and improve the robot navigation and obstacle avoidance.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/app13158925/s1, Video S1: supplementary.mp4.

Author Contributions: Conceptualization, D.M.-B., L.R. and L.M.; methodology, D.M.-B., L.R. and L.M.; software, D.M.-B.; validation, D.M.-B., L.R. and L.M.; formal analysis, D.M.-B., L.R. and L.M.; investigation, D.M.-B., L.R. and L.M.; resources, D.M.-B.; data curation, D.M.-B.; writing—original draft preparation, D.M.-B.; writing—review and editing, D.M.-B., L.R. and L.M.; visualization, D.M.-B.; supervision, L.R. and L.M.; project administration, L.M.; funding acquisition, L.M. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by the Spanish projects MCIN/AEI/PID2019-105390RB-I00 and Aragon Government_FSE-T45_20R.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The ROS package corresponding to the waypoint generator implementations can be accessed at https://github.com/dmartinezbaselga/dynamic_waypoint_generator (accessed on 29 July 2023).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lorente, M.T.; Owen, E.; Montano, L. Model-based robocentric planning and navigation for dynamic environments. *Int. J. Robot. Res.* **2018**, *37*, 867–889. [CrossRef]
2. Martínez, D.; Riazuelo, L.; Montano, L. Full-stack S-DOVS: Autonomous Navigation in Complete Real-World Dynamic Scenarios. In Proceedings of the ROBOT2022: Fifth Iberian Robotics Conference: Advances in Robotics, Zaragoza, Spain, 23–25 November 2022; Volume 2, pp. 14–25.
3. LaValle, S.M.; Kuffner, J.J., Jr. Randomized kinodynamic planning. *Int. J. Robot. Res.* **2001**, *20*, 378–400. [CrossRef]
4. Hart, P.E.; Nilsson, N.J.; Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100–107. [CrossRef]
5. Noreen, I.; Khan, A.; Habib, Z. Optimal path planning using RRT* based approaches: A survey and future directions. *Int. J. Adv. Comput. Sci. Appl.* **2016**, *7*, 97–107. [CrossRef]
6. Fareh, R.; Baziyad, M.; Rahman, M.H.; Rabie, T.; Bettayeb, M. Investigating reduced path planning strategy for differential wheeled mobile robot. *Robotica* **2020**, *38*, 235–255. [CrossRef]
7. Fox, D.; Burgard, W.; Thrun, S. The dynamic window approach to collision avoidance. *IEEE Robot. Autom. Mag.* **1997**, *4*, 23–33. [CrossRef]
8. Brock, O.; Khatib, O. High-speed navigation using the global dynamic window approach. In Proceedings of the 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C), Detroit, MI, USA, 10–15 May 1999; Volume 1, pp. 341–346.
9. Warren, C.W. Global path planning using artificial potential fields. In Proceedings of the 1989 IEEE International Conference on Robotics and Automation, Scottsdale, AZ, USA, 14–19 May 1989; pp. 316–317.
10. Quinlan, S.; Khatib, O. Elastic bands: Connecting path planning and control. In Proceedings of the 1993 IEEE International Conference on Robotics and Automation, Atlanta, GA, USA, 2–6 May 1993; pp. 802–807.

11. Stachniss, C.; Burgard, W. An integrated approach to goal-directed obstacle avoidance under dynamic constraints for dynamic environments. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Lausanne, Switzerland, 30 September–4 October 2002; Volume 1, pp. 508–513.
12. Minguez, J.; Montano, L. Sensor-based robot motion generation in unknown, dynamic and troublesome scenarios. *Robot. Auton. Syst.* **2005**, *52*, 290–311. [CrossRef]
13. Hsu, D.; Kindel, R.; Latombe, J.C.; Rock, S. Randomized kinodynamic motion planning with moving obstacles. *Int. J. Robot. Res.* **2002**, *21*, 233–255. [CrossRef]
14. Missura, M.; Bennewitz, M. Predictive collision avoidance for the dynamic window approach. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 8620–8626.
15. Dobrevski, M.; Skočaj, D. Adaptive dynamic window approach for local navigation. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 October–24 January 2020; pp. 6930–6936.
16. Rösmann, C.; Hoffmann, F.; Bertram, T. Integrated online trajectory planning and optimization in distinctive topologies. *Robot. Auton. Syst.* **2017**, *88*, 142–153. [CrossRef]
17. Fraichard, T.; Asama, H. Inevitable collision states—A step towards safer robots? *Adv. Robot.* **2004**, *18*, 1001–1024. [CrossRef]
18. Fiorini, P.; Shiller, Z. Motion planning in dynamic environments using velocity obstacles. *Int. J. Robot. Res.* **1998**, *17*, 760–772. [CrossRef]
19. Shi, H.; Shi, L.; Xu, M.; Hwang, K.S. End-to-End Navigation Strategy with Deep Reinforcement Learning for Mobile Robots. *IEEE Trans. Ind. Inform.* **2020**, *16*, 2393–2402. [CrossRef]
20. Lei, X.; Zhang, Z.; Dong, P. Dynamic path planning of unknown environment based on deep reinforcement learning. *J. Robot.* **2018**, *2018*, 5781591.
21. Chen, C.; Liu, Y.; Kreiss, S.; Alahi, A. Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 6015–6022.
22. Mackay, A.K.; Riazuelo, L.; Montano, L. RL-DOVS: Reinforcement Learning for Autonomous Robot Navigation in Dynamic Environments. *Sensors* **2022**, *22*, 3847. [CrossRef] [PubMed]
23. Martinez, D.; Riazuelo, L.; Montano, L. Deep reinforcement learning oriented for real world dynamic scenarios. *arXiv* **2022**, arXiv:2210.11392.
24. Ibarz, J.; Tan, J.; Finn, C.; Kalakrishnan, M.; Pastor, P.; Levine, S. How to train your robot with deep reinforcement learning: Lessons we have learned. *Int. J. Robot. Res.* **2021**, *40*, 698–721.
25. Dulac-Arnold, G.; Levine, N.; Mankowitz, D.J.; Li, J.; Paduraru, C.; Gowal, S.; Hester, T. Challenges of real-world reinforcement learning: Definitions, benchmarks and analysis. *Mach. Learn.* **2021**, *110*, 2419–2468. [CrossRef]
26. Guldenring, R.; Görner, M.; Hendrich, N.; Jacobsen, N.J.; Zhang, J. Learning local planners for human-aware navigation in indoor environments. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 October–24 January 2020; pp. 6053–6060.
27. Kästner, L.; Buiyan, T.; Jiao, L.; Le, T.A.; Zhao, X.; Shen, Z.; Lambrecht, J. Arena-Rosnav: Towards deployment of deep-reinforcement-learning-based obstacle avoidance into conventional autonomous navigation systems. In Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 27 September–1 October 2021; pp. 6456–6463.
28. Kästner, L.; Bhuiyan, T.; Le, T.A.; Treis, E.; Cox, J.; Meinardus, B.; Kmiecik, J.; Carstens, R.; Pichel, D.; Fatloun, B.; et al. Arena-bench: A benchmarking suite for obstacle avoidance approaches in highly dynamic environments. *IEEE Robot. Autom. Lett.* **2022**, *7*, 9477–9484. [CrossRef]
29. Gerkey, B.; Vaughan, R.T.; Howard, A. The player/stage project: Tools for multi-robot and distributed sensor systems. In Proceedings of the 11th International Conference on Advanced Robotics, Coimbra, Portugal, 30 June–3 July 2003; Volume 1, pp. 317–323.
30. Fox, D. KLD-sampling: Adaptive particle filters and mobile robot localization. *Adv. Neural Inf. Process. Syst. (NIPS)* **2001**, *14*, 26–32.
31. Przybyła, M. Detection and tracking of 2D geometric obstacles from LRF data. In Proceedings of the 2017 11th International Workshop on Robot Motion and Control (RoMoCo), Wasowo Palace, Poland, 3–5 July 2017; pp. 135–141.
32. Grisetti, G.; Stachniss, C.; Burgard, W. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Trans. Robot.* **2007**, *23*, 34–46. [CrossRef]
33. Khan, A.H.; Li, S.; Luo, X. Obstacle avoidance and tracking control of redundant robotic manipulator: An RNN-based metaheuristic approach. *IEEE Trans. Ind. Inform.* **2019**, *16*, 4670–4680. [CrossRef]
34. Wu, Q.; Shen, X.; Jin, Y.; Chen, Z.; Li, S.; Khan, A.H.; Chen, D. Intelligent beetle antennae search for UAV sensing and avoidance of obstacles. *Sensors* **2019**, *19*, 1758. [CrossRef] [PubMed]