IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 8 November 2023, accepted 7 January 2024, date of publication 26 January 2024, date of current version 5 February 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3359156

RESEARCH ARTICLE

Motion Planning for Mobile Robots Using Uncertain Obstacle Estimation

ZOLTÁN GYENES[1,2], BARNABÁS PAJKOS[1], LADISLAU BÖLÖNI[2], (Senior Member, IEEE), AND EMESE GINCSAINÉ SZÁDECZKY-KARDOSS[1]

[1]Department of Control Engineering and Information Technology, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics, 1111 Budapest, Hungary
[2]Department of Computer Science, University of Central Florida, Orlando, FL 32816, USA

Corresponding author: Zoltán Gyenes (zgyenes@iit.bme.hu)

The research was supported by the Hungarian Eötvös State Scholarship and by the Rosztoczy Foundation Scholarship. The research was also supported by the National Science Foundation under the grant CPS-1931767.

ABSTRACT The collision-free movement of a mobile robot in the presence of dynamic obstacles remains a significant challenge. In addition to self-localization, we also need to worry about the location of the moving obstacles, taking into account the noise in the sensors and the uncertainty in the movement of these obstacles. In this paper, we propose an approach for omnidirectional robot maneuvering in a 2D workspace that combines a particle filter for the estimation of the obstacles from LiDAR laser sensor data and a variation of the Velocity Obstacles (VO) reactive motion planning method. The position and the velocity vector of the obstacles, as well as the uncertainty degree is estimated by the particle filter. These outputs are combined with the VO algorithm to achieve motion planning that takes into account the current level of uncertainty as well as a cost function that expresses the risk tolerance of the user. We validate the approach in simulation and in experiments with a physical robot.

INDEX TERMS Collision avoidance, motion planning, particle filter, state estimation, velocity obstacles.

I. INTRODUCTION

Collision-free navigation of a robot in a dynamic environment requires not only self-localization and path planning but also the ability to track and avoid mobile obstacles. This is complicated by the noise and limited accuracy of sensors, and obstacles being occluded by landmarks or other obstacles. Most reactive motion planning algorithms focus on finding the fastest trajectory to the goal [1], [2], [3], while the primary objective of the state estimation techniques is the self-localization of the robot [4], [5], [6]. In contrast, estimating the uncertain state of mobile obstacles and accordingly adapting the path received significantly less attention.

In this paper we develop a technique that estimates the position and velocity of mobile obstacles, tracks the uncertainty of this estimate and uses this information to plan a path with a risk level that can be specified by the user. The technique builds on a combination of particle filters and an extension of the velocity obstacle method.

The contributions of this paper can be summarized as follows:

- We successfully developed and implemented a novel state perception method that accurately estimates the position and velocity vectors of obstacles that occur within the workspace of an agent, utilizing data received from LiDAR sensor measurements with a particle filter.
- We developed and validated a cost-function-based velocity selection method that uses a calculated uncertainty degree from the estimates of the particle filter to achieve collision-free motion planning.
- We compared the introduced algorithm with other motion planning methods and presented that our method outperforms the baseline approaches.
- A real environmental test is also executed on an omnidirectional robot.

The structure of this paper can be described as: Section II reviews existing methodologies in state estimation, global

The associate editor coordinating the review of this manuscript and approving it for publication was Okyay Kaynak.

© 2024 The Authors. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

and reactive motion planning methods. Following that, Section III outlines the main algorithms such as the particle filter and the Velocity Obstacles method, which serve as the basis of our novel approach. Subsequently, Section IV introduces the proposed Particle Filter Velocity Obstacles method. Section V presents both the simulation and experimental results. Finally, Section VI summarizes the main contributions of the study.

II. RELATED WORK

This section presents the state of the art state estimation methods and motion planning algorithms in the field of Robotics.

A. STATE ESTIMATION METHODS IN ROBOTICS

Estimating the state variables, such as the position and velocity of the agent is a critical requirement for many tasks in the field of robotics. One of the most popular approaches is variations of Bayesian filtering [7], [8], [9]. The original Kalman filter algorithm [10], [11], [12] is usable under the assumption of a linear system. The Extended Kalman Filter (EKF) linearizes the model at each time step, allowing state estimation for non-linear systems as well [13], [14]. This property made it very useful for the localization problem in mobile robotics [4], [11], [15], [16], [17]. The Unscented Kalman Filter (UKF) offers an alternative to EKF which can offer a more accurate solution by avoiding the linearization step [18], [19], [20]. The Kalman Filter assumes Gaussian noise.

The particle filter algorithm, introduced initially in 1955 [21], estimates system state by simulating numerous particles or ''molecules.'' The algorithm was later renamed as the Bootstrap filter in 1993 [22], upon its implementation as a recursive Bayesian filter. The fundamental premise of this algorithm is to construct a posterior distribution using differently weighted samples, which are calculated and updated at each sampling interval based on measurement data. For nonlinear systems, this form of Bayesian filter has been shown to outperform the EKF and UKF, even when the number of particles is constrained [23].

The Collision Avoidance with Localization Uncertainty (CALU) algorithm, designed for multi-robot collision avoidance, leverages the particle filter to solve the localization problem of mobile robots. This novel method integrates the Optimal Reciprocal Collision Avoidance (ORCA) approach [24] to reach a collision-free solution. The central aim of this method is to constrain the inherent error within the localization process.

The Simultaneous Localization and Mapping (SLAM) methodology has two main steps: revision of an environmental map, and calculation of the position of the mobile agent in the environment [5], [25], [26], [27]. The SLAM methodology has been applied in combination with both the Kalman filter [28] and the particle filter [28], [29]. It is important to note that the original SLAM algorithm did not possess the capability to segment obstacles from the overall environment, and was restricted to defining the position information within the map. Contrasting with this, our proposed approach also enables the computation of velocity vectors considering the obstacles, that occur in the agent's workspace.

The algorithm termed 'Collision Avoidance under Bounded Localization Uncertainty' (COCALU) [30] proposes a novel method of convex hull peeling, with an aim to reduce the error associated with localization. This approach presents superior performance when compared with the preceding 'Multi-robot CALU algorithm [31].

B. MOTION PLANNING METHODS IN ROBOTICS

The motion planning methods can primarily be separated into global and local approaches, considering the depth of environment knowledge.

Global motion planning algorithms, often categorized as offline, assume comprehensive, prior knowledge about the environment. This level of information is usually available in a static environment. Some well-known global motion planning methods are the hybrid A* [32] and the Rapidly-exploring Random Tree (RRT) [33], [34], [35].

Conversely, reactive motion planning methods focus on local environmental information, typically received through the robot's onboard sensors. All of the remaining motion planning algorithms are reactive motion planning methods for mobile robots.

The Dynamic Window Approach (DWA) is an often used motion planning method for mobile robots, capable of generating real-time, collision-free paths, considering non-holonomic constraints like limited turning ability [36], [37]. Recently, DWA has been employed to resolve the motion planning problem for forklift-automated guided vehicles [38].

The Artificial Potential Field method is also a well-known reactive motion planning algorithm, where obstacles generate repulsive virtual forces, and the target generates an attractive virtual force. The sum of these forces results in the computation of the robot's velocity vector at each sampling time [39], [40], [41], [42], [43]. Originally, APF was developed for static environments, the algorithm has also been adapted for dynamic obstacles too, used also by robotic soccer applications [43].

The velocity obstacle (VO) algorithm, another local motion planning technique, computes those velocity vectors, that potentially result in a collision in the future between the agent and obstacles [44]. Every sampling step involves the selection of a collision-free velocity vector following specific strategies. These strategies range from 'To Goal' (TG), which prioritizes the largest velocity vector towards the goal, to maximum velocity (MV) strategy, which identifies the maximum velocity within a certain angle to the path towards the goal. The VO approach has been extended to diverse scenarios, including differential-driven robots [45], [46], UAVs (unmanned aerial vehicles) [47], ships [2], and multi-robot collision avoidance [48], [49], [50]. Although its

primary development is for mobile robots, the VO method can also be implemented for robotic manipulators [51]. Furthermore, an energy-efficient algorithm was developed for UAVs with limited battery capacity in multi-robot contexts [1], [3].

The probability-based motion planning for mobile robots has been a highly researched topic for a long time. The Probabilistic Velocity Obstacles (PRVO) algorithm [52], an augmented variant of the Reciprocal Velocity Obstacles (RVO) methodology [53], includes probabilistic principles alongside time scaling mechanisms and Bayesian decomposition. This technique outperforms bound-based methods in terms of running times. The efficiency of the algorithm has been validated through a series of simulation results [54]. In [55], a pyramid structure was used that provided conditional probabilities for every pixel considering the uncertainty of the environment. The Probabilistic Foam Method (PFM) [56] was introduced to guarantee a safe path for the agent. In this method, the environment is structured with bubbles and uses a breadth-first search to generate the optimal solution. The probabilistic representation of the uncertainty was used in [57] that planned the future probabilistic distribution of the state of the agent and also the collision's probability. The solution is provided by solving the linear program considering linear chance constraints. A probabilistic method [58] was also introduced to calculate the probability of human motion for the agent's motion planning method. To solve this task, a probability grid was introduced from the observation.

As can be seen, all of the introduced state estimation methods focused on the self-localization problem in the field of Robotics, but the SLAM, and the main goal of the reactive motion planning methods was to reach the goal position as fast as it is possible. In our work, we combined the state estimation for the environment with the motion planning method. The state estimation method provides not only the perceived states of the obstacles but also an uncertainty degree that can be used by the introduced motion planning method.

III. BACKGROUND

Two well-known algorithms, the Velocity Obstacle method, and the particle filter method are used as the basic of our introduced algorithm. This section presents these algorithms in detail.

A. THE PARTICLE FILTER ALGORITHM

In this section, we offer an in-depth explanation of the steps required to execute the particle filter algorithm [23]. The particle filter is an approach for approximating a system's posterior distribution using a discrete density. Among its benefits over other recursive Bayesian filtering techniques is the capacity to manage nonlinear dynamic models, as well as linear ones. Furthermore, the particle filter is applicable to systems that have non-Gaussian noise.

The particle filter algorithm employs a weighted set of points (S_k) to approximate the posterior distribution. This set consists of pairs $< \mathbf{x}_k^{(i)}, w_k^{(i)} >$, where $\mathbf{x}_k^{(i)}$ corresponds to the state of the i^{th} particle at time k, and $w_k^{(i)}$ indicates the probability (weight) of that state. The number of particles, denoted by N, can vary throughout the iteration. The weight vector, represented as $\mathbf{w}_k$, is subject to the constraint that its elements' sum equals 1 ($\sum_{i=1}^{N} w_k^{(i)} = 1$).

Before executing the particle filter algorithm, the filter must be initialized. This involves specifying the state transition function, the resampling approach, and the number of particles, represented as N. The state transition function can be expressed mathematically using equation (4).

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) + \xi_k \tag{1}$$

where f denotes a linear or nonlinear function, $\mathbf{x}_k$ means the actual state at time k, $\mathbf{u}_k$ represents the control input, and ξ_k means the system noise.

During the initialization phase of the particle filter algorithm, a random set of points, represented as $\mathbf{x}_1^{(i)}$, is produced by sampling from the prior distribution P_{x0}. Each particle is initially assigned a weight of $w_1^{(i)} = 1/N$. In the absence of initial measurement data, the weight is set to $1/N$. The particle filter algorithm's steps, as described in [23], include the following:

1) Step of Measurement update
   The crucial task of updating particle weights in the Particle filter algorithm is essential for precise state estimation. This is achieved by integrating sensor measurements (z_k) into the weight update equation for every particle. It is important to note that the measurements are noisy.

$$w_k^{(i)} = \frac{\mathrm{w}_{k-1}^{(i)} P(z_k|x_k^{(i)})}{\sum_{j=1}^{N} \mathrm{w}_{k-1}^{(j)} P(z_k|x_k^{(j)})} \tag{2}$$

2) Estimation
   The estimated state can be determined:

$$\hat{x}_k = \sum_{i=1}^{N} w_k^{(i)} x_k^{(i)} \tag{3}$$

3) Resampling
   Typically, N samples are chosen from the particle set with replacement, taking into account the particle weights. Various resampling techniques have been explored in past research, as highlighted in [59].
4) Time update
   In this step, prediction can be formulated:

$$\mathbf{x}_{k+1}^{(i)} = f(\mathbf{x}_k^{(i)}, \mathbf{u}_k) + \zeta \tag{4}$$

   where ζ means a random variable with a given deviation. Equation (4) can be employed during the time update step for non-linear models. After that, the iteration continues with 1).

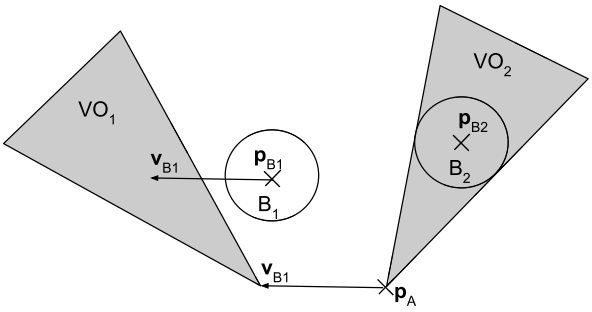

FIGURE 1. An example of the VO method for a robot at position p_A navigating an environment consisting of a moving obstacle B_1 at position p_{B1} with velocity v_{B1} and a static obstacle B_2 at p_{B2}. The gray area illustrates $VO = VO_1 \cup VO_2$.

B. THE VELOCITY OBSTACLES METHOD

Our methodology is based on the Velocity Obstacles (VO) algorithm [44], which aims to find a velocity vector that ensures the agent remains in collision-free states, given the measurement or estimate of the obstacles' position and velocity data during the decision-making process.

The set of velocity vectors for agent A, represented by the velocity obstacle cone VO_i, includes all velocities that would lead to a future collision with obstacle B_i.

$$VO_i = \{\, v_A \mid \exists\, t : A(p_A + v_A t) \,\cap\, B_i(p_{Bi} + v_{Bi} t) \neq 0 \} \quad (5)$$

In the given equation, $A(p)$ and $B_i(p)$ represent positions the robot and obstacle i, respectively. Equation (5) presents that selecting a robot velocity vector $v_A \in VO_i$ will lead to a collision between A and B_i at time t, assuming the velocities of the obstacles and robot remain constant until t. Conversely, choosing a velocity $v_A \notin VO_i$ guarantees that A and B_i will avoid collision as long as their velocities v_A and v_{Bi} are not changed.

Taking into account all obstacles, the final velocity obstacle, denoted as VO, is formed by combining the individual VO_i sets:

$$VO = \bigcup_{i=1}^{m} VO_i \quad (6)$$

where m means the number of obstacles.

The set of reachable velocities, RV, includes all possible velocity vectors v_A that the robot can attain by the sampling time, given its motion capabilities (such as limited inputs). An important aspect of the VO method involves computing the set of reachable avoidance velocities (RAV), which can be achieved by subtracting the VO set from the RV set, resulting in the robot velocities that are both reachable and will not result in collisions. These velocities allow the robot to follow a collision-free path. A common technique involves discretizing the RAV set by introducing a grid on the RAV set.

Figure 1 demonstrates the implementation of the VO method for a robot navigating through an environment containing both a static and a moving obstacle. For the static, disk-shaped obstacle B_2, the VO_2 is a cone in the plane, with its vertex at p_A and its sides tangent to the obstacle's circle.

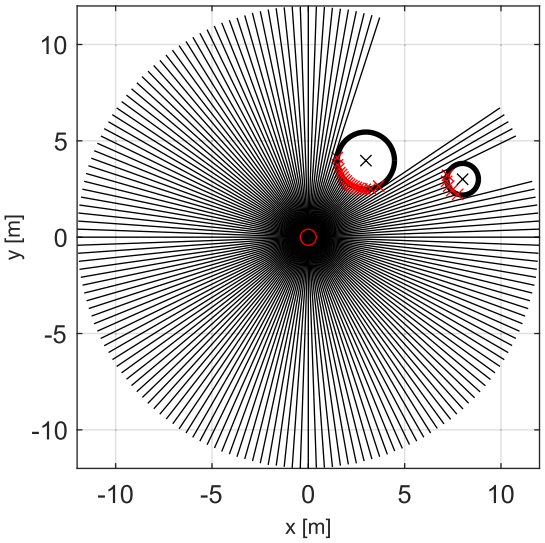

FIGURE 2. Data acquisition via a LiDAR sensor, with the autonomous agent situated at the origin (depicted by a red circle). The sensor's distance measurements are illustrated with black lines. Within the workspace, two obstacles occurred, each denoted by a black circle, and the actual measurement points on the obstacles are presented by red x-s.

In the case of the moving obstacle B_1, the cone must be shifted by the velocity vector v_{B1} of the obstacle to define the corresponding VO_1 set. The introduced algorithm assumes disk-shaped robots and obstacles. In the first step, the radii of the obstacles are increased by the radius of the agent resulting in a point as the robot. Using this assumption, the velocity selection and collision avoidance can be calculated better for the autonomous mobile robot. There are two well-known strategies, the To Goal strategy (VOTG) method, which selects always the velocity vector resulting in the fastest solution for the agent, and the Maximum Velocity (MV) strategy which selects the velocity vector from a pre-defined angle. In our scenario, we use the VOTG method as a baseline strategy to compare the results.

IV. THE PARTICLE FILTER VELOCITY OBSTACLES METHOD

In this research, we assume noisy LiDAR measurement data about the disk-shaped obstacles occurring in the workspace of the agent (if the obstacle is not disk-shaped, a circle bounding the obstacle can be substituted). The segmentation must be solved for these obstacles using the Least Square method for circle fitting. Figure 2 presents the results of the measurement of LiDAR sensor in a simulated environment. The sensor has a maximum range of 12 meters and an angular resolution within the interval [0°, 1°], selecting a resolution of 0.5° for the current situation. The LiDAR measurement data is displayed in Figure 2. Obstacle locations in the workspace can be identified by segmenting the acquired data by calculating the center point and the radius of the obstacle using the Least Square estimation. Additionally, the simulation includes measurement noise in the environment. The assumption is made that all obstacles within the

workspace assume a disc-like form. These computations take place within a global coordinate framework. The center points of the obstacles, denoted by black x-s, can be computed through the application of the Least Square (LS) method. The obstacle measurement approach utilized in this research is thoroughly explained in our earlier work [60].

A. PERCEPTION METHOD WITH THE PARTICLE FILTER

This section details the particle filter approach, which employs a weighted collection of particles to present the potential states of a system at a particular sampling interval. These states can be denoted by the **particles** matrix, having dimensions of 2 rows representing the number of the obstacles and N columns, and the weight vector, symbolized as $\mathbf{w}$. Prediction of obstacle states is achieved using measurement data received from the LiDAR sensor, using both the deviation of the system and the measurement noise. The process for implementing the algorithm is summarized in Section III-A, where the primary steps are repeated in an iterative manner.

1) CALCULATING THE WEIGHTS OF THE PARTICLES

Assuming that the workspace only contains disk-shaped obstacles, we can compute the center point using the LiDAR sensor's measurement data. Here, a measured x center (z_{p_x}) and estimated y center (z_{p_y}) are derived through a *Least square error* estimation, accounting for every data point measured by the sensor.

Each particle represents the obstacle's real center point. The deviation of the distance' measurement noise is represented by (MN_d). Each particle contains data on the x position ($\hat{z}_{p_x}^{(i)}$) and the y position ($\hat{z}_{p_y}^{(i)}$).

Initially, the x position is taken into account when computing the weights:

$$w_{p_x}^{(i)} = \frac{1}{\sqrt{2\pi\ MN_d}} exp(\frac{-(z_{p_x} - \hat{z}_{p_x}^{(i)})^2}{2MN_d}) \tag{7}$$

The x position weights should be normalized:

$$w_{p_x}^{(i)} = \frac{w_{p_x}^{(i)}}{\sum_{j=1}^{N} w_{p_x}^{(j)}} \tag{8}$$

In this instance, the total weight of the particles sums up to 1.

Subsequently, we can compute the weights while taking the y position into consideration:

$$w_{p_y}^{(i)} = \frac{1}{\sqrt{2\pi\ MN_d}} exp(\frac{-(z_{p_y} - \hat{z}_{p_y}^{(i)})^2}{2MN_d}) \tag{9}$$

Normalization is also required for the y position weights:

$$w_{p_y}^{(i)} = \frac{w_{p_y}^{(i)}}{\sum_{j=1}^{N} w_{p_y}^{(j)}} \tag{10}$$

The ultimate weights of the particles can be calculated by using the x and y position weights:

$$w^{(i)} = w_{p_x}^{(i)} \cdot w_{p_y}^{(i)} \tag{11}$$

The final weights should also be subjected to normalization:

$$w^{(i)} = \frac{w^{(i)}}{\sum_{j=1}^{N} w^{(j)}} \tag{12}$$

2) THE STATE PERCEPTION ALGORITHM

For the state perception of the obstacles, the calculated weights of the particles can be used. In this scenario, Equation (3) is applied. The velocity estimation of the obstacles is calculated from the estimated positions (because the sensor measurement data is available only for the position of the obstacles).

3) RESAMPLING ALGORITHM AND STATE TRANSITION METHOD

In this step, the systematic resampling algorithm was employed for particle selection [60].

The state of the dynamic obstacle model is composed of two coordinates (p_x, p_y), indicating its position. Following the perception of the position, the velocity vector (v_x, v_y) can be calculated. The state transition model takes both the perceived position and velocity into account.

4) CALCULATION OF THE UNCERTAINTY DEGREE

At every time step, an uncertainty degree (α) can be also calculated which can be useful in the motion planning algorithm for the mobile agent. First, the maximum change of the velocity must be calculated:

$$\Delta v = \max||\mathbf{v}_k(i) - \mathbf{v}_{k-1}(i)|| \tag{13}$$

where k means the actual time step, $k-1$ is the previous time step, and i represents the *ith* obstacle in the workspace of the agent.

Also, the saturated deviation of the particles can be determined:

$$d_p = \min\left(1, \max(std(\textbf{particles}))\right) \tag{14}$$

In this equation, *std* denotes the standard deviation calculated for both the x and y positions of the obstacles, with the larger value being chosen as the uncertainty degree. The maximum value is capped at 1.

Finally, the uncertainty degree (α) can be calculated:

$$\alpha = \max\left(\Delta v, d_p\right) \tag{15}$$

A larger value for the uncertainty degree indicates a higher degree of uncertainty regarding the precision of the current perception.

B. COLLISION AVOIDANCE WITH PARTICLE FILTER VELOCITY OBSTACLES METHOD

In the introduced Particle Filter Velocity Obstacle method, the velocity decision for the agent is focused on two key elements: speed and safety. Both of these factors are simultaneously taken into account by employing a weighted objective function that can be calculated in accordance with the actual uncertainty degree at the specific sampling time.

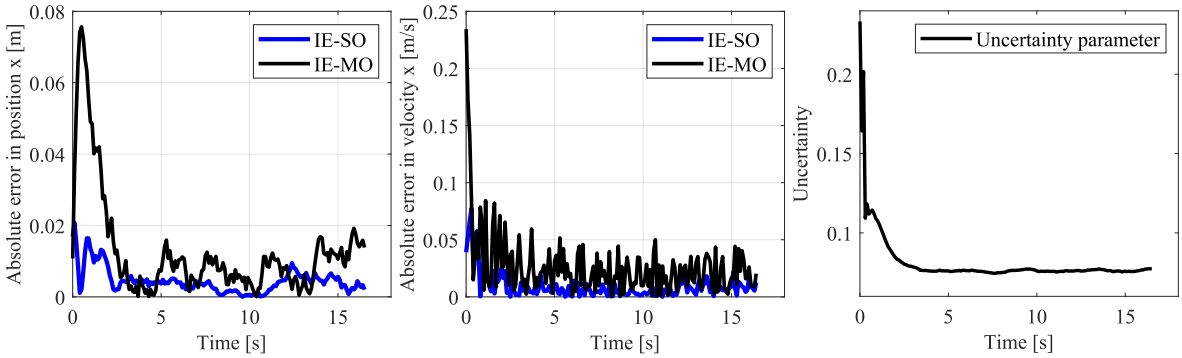

FIGURE 3. Result of the first example. The left and center figures represent the absolute error in the position and the velocity and the right figure shows the uncertainty degree.

The speed element, denoted as GO, can be calculated as:

$$GO(\mathbf{v}_\mathrm{i}) = \frac{r_i \cos \Delta\theta_i}{v_{max}} \tag{16}$$

In this context, v_{max} means the maximum velocity achievable by the agent, given the constraints of its kinematic characteristics. The term $\Delta\theta_i = \theta_{rg} - \theta_{rv_i}$ represents the difference between the angle towards the goal (θ_{rg}), the angle of the velocity vector (θ_{rv_i}) and i represents the i^{th} velocity vector on the grid, which is a discretized set of the velocities. Additionally, $r_i = || \mathbf{v}_\mathrm{i} ||$ denotes the magnitude of the velocity vector.

The safety element (SA) in the objective function denotes the safety. It is typically sufficient to consider just the closest Velocity Obstacle (VO). Also, if the robot can't approach the nearest obstacle within a given time span, it's considered entirely safe, and additional distance wouldn't improve this safety status. Thus, we formulate the expression as follows:

$$SA(\mathbf{v}_\mathrm{i}) = \min \left(1, \frac{\min\limits_{v_{VO} \in VO} \| v_i - v_{VO} \|}{v_{max} \cdot T_{max}} \right) \tag{17}$$

In the above expression, T_{max} is a predefined parameter, representing the maximum time interval taken into consideration while the motion of the agent and v_{VO} is the closest point in the VO cone considering the distance between the VO cone and the investigated velocity vector ($\mathbf{v}_\mathrm{i}$).

Additionally, the speed and safety components of the objective function (OF) are combined using a weighted mean, in which the parameter α denotes the previously calculated uncertainty degree. In scenarios where the uncertainty grows, the significance of the safety component within the objective function must be correspondingly increased. The primary objective of the algorithm is to determine the velocity vector for the agent that maximizes the value of the objective function.

$$OF(\mathbf{v}_\mathrm{i}) = \begin{cases} \alpha\ SA(\mathbf{v}_\mathrm{i}) + (1-\alpha)\ GO(\mathbf{v}_\mathrm{i}) & \text{if } \mathbf{v}_\mathrm{i} \in RAV \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

V. EXPERIMENTS AND RESULTS

In this section, the simulation and experimental results are presented using the Particle Filter Velocity Obstacle method.

A. SIMULATION RESULTS

All of the presented examples were compared with the fastest solution (VOTG method), which was presented in Section III-B, and with one of our previously introduced motion planning algorithms (SVO method) [61], where there is a predefined constant safety parameter.

1) FIRST EXAMPLE: OBSERVABLE OBSTACLES

In the initial illustrative case, the agent's workspace consists of both stationary and dynamic obstacles. The robot receives the sensor information at each sampling time. Figure 3 presents the absolute error considering the positions and velocities for both types of obstacles, in addition to the change of the uncertainty degree throughout the movement. In the referenced figure, 'IE-SO' means the Inertial Estimation for stationary obstacle, while 'IE-MO' denotes the Inertial Estimation for the moving obstacle. It is notable that the estimation accuracy considering the obstacle positioning is about 0.02 m and the velocity estimation error is 0.04 m/s using N=10000 particles; velocity estimation maintains a substantial degree of precision and usability. The estimation accuracy for stationary obstacles has a better performance than at the dynamic obstacle.

In Figure 4, the trajectory of the autonomous agent towards the target position is depicted (the start position is at the origin, and the target position is at $[14, -1]$). Under these conditions, the trajectory results in the same solution for both the VOTG and Particle Filter Velocity Obstacle (PFVO) methodologies. The uncertainty degree remains small during the whole motion because the agent can reach the sensor measurement data at every sampling time. Conversely, the Safety Velocity Obstacle (SVO) approach navigates toward the target along a more spacious trajectory, considering a higher safety parameter.

2) SECOND EXAMPLE: VARIABLE VELOCITY MOVING OBSTACLE

In the second experiment, we consider a scenario with two moving obstacles. One obstacle maintains a constant velocity vector, while the other demonstrates a change in velocity during its movement. The variability in the

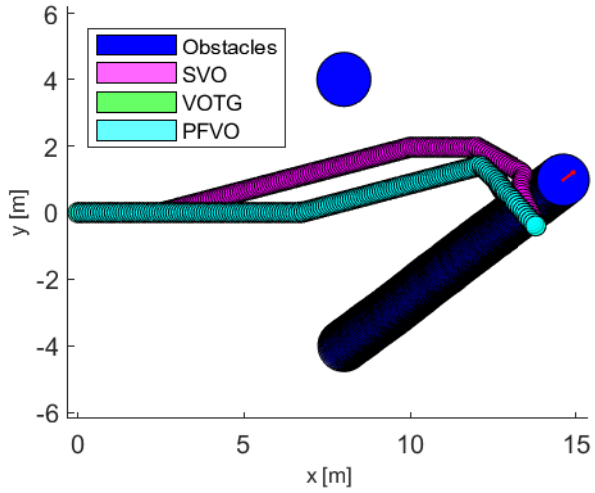

FIGURE 4. Path of the mobile agent to the goal in the first example.

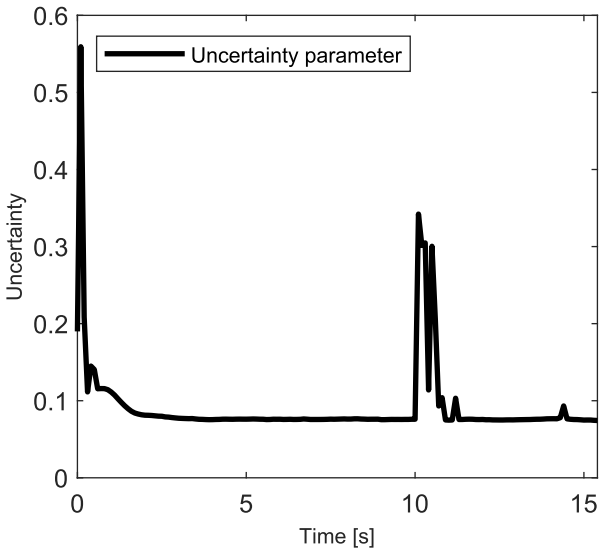

FIGURE 5. The changes of the uncertainty degree in the second example.

uncertainty degree, as presented in Figure 5, is noteworthy. The uncertainty degree experiences an increase when the obstacle changes its velocity vector. However, once the agent perceives this new information over time, the uncertainty degree subsequently decreases.

One obstacle starts its motion in the position of $[4; -3]$ and has a velocity vector of $[0.8; 0]$ m/s until 10 s when it changes the velocity vector to the opposite direction and continues the motion with the velocity of $[-2; 0]$ m/s. The agent's path to the goal position is illustrated in Figure 6 (the start position is at the origin, and the target position is at [14, -1]). Analyzing the agent's paths under different strategies reveals that even if the uncertainty degree is higher, the PFVO and VOTG strategies result in quite similar path. This leads to the fastest goal attainment as none of the moving obstacles impact the robot's trajectory. Conversely, the SVO strategy (with parameter $\alpha = 0.5$), due to its higher constant safety parameter, results in a more evasive path, offering a wider

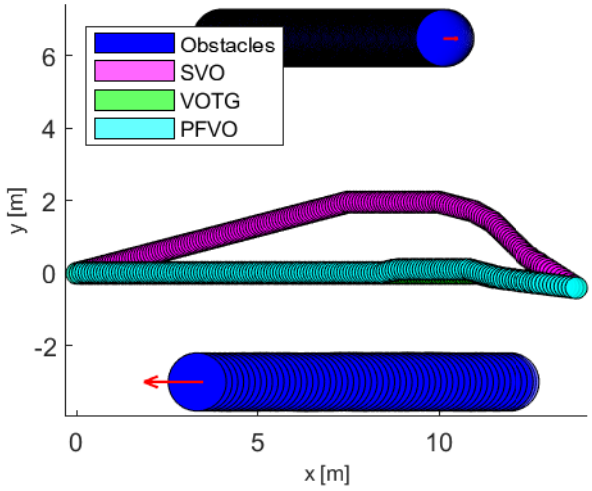

FIGURE 6. Result of the motion of the agent in the second example.

maneuver from the obstacle that exhibits a change in velocity during its motion.

3) THIRD EXAMPLE: OBSTRUCTED OBSTACLE

In the third experiment, the environment contains a static obstacle and a dynamic obstacle that, during its trajectory, becomes covered by the former. As the dynamic obstacle disappears from detection, the autonomous agent cannot receive any real-time sensory input regarding it. Consequently, the agent has to estimate the object's position and velocity vector based on historical data.

In Figure 7, the left part illustrates the absolute error in estimating the position within this scenario (in this scenario, the errors are presented considering the x coordinates but the result is similar in y coordinates). 'IE-SO' and 'IE-MO' retain their meanings as outlined in the preceding examples. It becomes evident that as long as the LiDAR sensor provides the agent with sensor measurement data, the error in estimation decreases. However, when the moving obstacle is covered by the static one (between about 4s and 8s), a rise in the estimation error for the position is observable, and this continues to increase until the agent can receive sensor information again.

This increasing error can largely be connected with the elimination of the resampling phase in the Particle Filter Velocity Obstacle (PFVO) algorithm. In such instances, the particles disperse, influencing the estimation, and the algorithm can only rely on the data from the latest perception. Despite the growing estimation error during the period of sensor information unavailability, it is notable that the error remains comparatively smaller than the scenario where the obstacle disappears, it becomes hidden behind the static obstacle. This is illustrated in the center segment of Figure 7., where 'LS-MO' denotes the moving obstacle as last seen, implying the case when the obstacle halts upon disappearing.

The uncertainty degree (UC) is depicted alongside the absolute error in the right part of Figure 7. A clear correlation

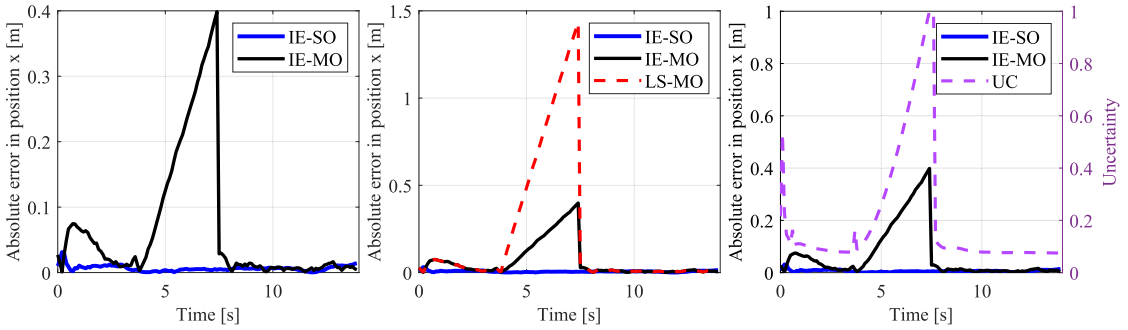

FIGURE 7. Result of the third example. The left and the center figures represent the absolute error of the positions and the right figure shows the absolute errors and uncertainty degree.

is observed between the uncertainty degree and the estimation error. The uncertainty degree reaches its highest value when the sensor information becomes unavailable, and as depicted, it reaches its peak value (1) during this period. It begins to decrease as the sensor information is available again.

Figure 8 visually represents the temporal transformation in the spread of particles. At the beginning of the motion, the particles are evenly distributed (using uniform distribution) across the entire workspace, as depicted in the figure's left segment. The particles are symbolized by red x-marks, while blue x-marks indicate the resampled particles. The black circle marks the estimated current center point of the obstacle.

As the agent begins to receive sensor information regarding the obstacle, a contraction in the particle distribution becomes smaller, as illustrated in the figure's center segment. However, upon the obstacle's disappearance, the particle distribution begins to expand once more, an effect clearly observable in the figure's right segment. During this period, considering the missing of LiDAR measurements, the resampling part is missing in the PFVO algorithm.

The comparative analysis of diverse motion planning strategies is the focus of this section. Figure 9 illustrates the spatial relationship between the agent and obstacles in terms of distance. The VOTG strategy results in the fastest solution as it selects the quickest velocity vector, despite the potential risks of collision this may pose for the agent during motion. As evidenced by the figure, this strategy results in a trajectory that with tangential movement to the obstacles (where R_r represents the radius of the robot), reducing the distance to null. Notably, however, the uncertainty inherent in the sensor measurements makes this tangential motion prone to collision in almost all instances, thereby collision-free motion cannot be always reached.

In the case of the Particle Filter Velocity Obstacle (PFVO) strategy, the agent-obstacle distance diminishes quickly as long as sensor information is available. Yet, during periods of data unavailability (around the 6-second mark), the increasing uncertainty degree prompts a more safer trajectory that distances the agent from the nearest obstacle. Once sensor data is reacquired (around the 9-second mark), the agent quickly converges on the goal, achieving it slightly faster than the Safety Velocity Obstacle (SVO) strategy.

TABLE 1. Running times of the PFVO algorithm considering the number of the particles.

Number of Particles (N)	Running time [s]	Average error in position [m]
100	0.006	1.25
500	0.0077	0.45
1000	0.0078	0.0503
5000	0.0135	0.0385
10000	0.0212	0.0294
50000	0.0647	0.0209
100000	0.1642	0.0119

The SVO strategy, on the other hand, consistently prioritizes safety during motion planning due to its constant safety parameter (0.5). Consequently, it formulates a smoothly safe path, unaffected by the change of sensor data. However, it requires the most time and results in the longest path to the goal.

Figure 10 presents the generated paths for each strategy (the start position of the agent is at the origin, and the target position is at [11, -0.5]).

Table 1 presents the relationship between the number of particles, corresponding running times, and the average position error occurred during execution. It is noteworthy that an increase in particle count corresponds to an increase in running time and a decrease in average error. Notably, with particle numbers 100 and 500, the average error seems extremely high (1.25m and 0.45m respectively). Beyond the particle count of 100000, the running time exceeds 0.1s, thereby exceeding the boundary for real-time feasibility. An appropriate trade-off between running time and the average error is achieved by selecting a particle number within the range of 1000 to 50000. In our presented scenarios, we adopt a particle count of 10000, an approach that effectively reduces the average error while maintaining a fast computational process.

B. ROBOT SETUP

An omnidirectional mobile robot equipped with a LiDAR was built for the demonstration of the introduced algorithm that can be seen in Figure 11.

ROS2 (Robot Operating System) [62] is a flexible and distributed framework for building robotic systems that provides

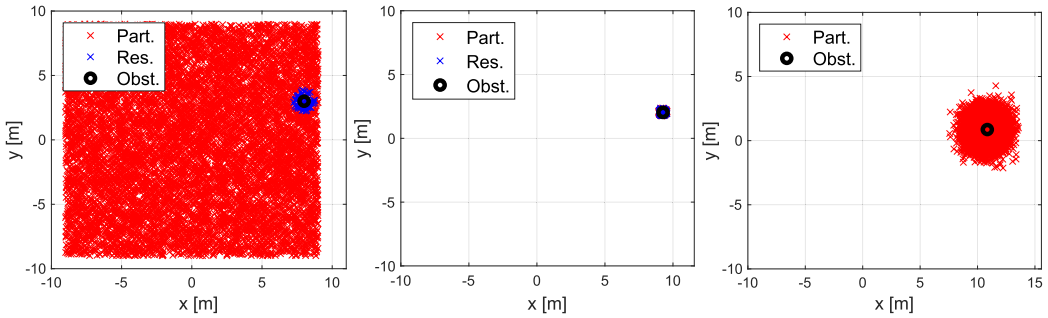

FIGURE 8. Changes in the distribution of the particles during the motion of the agent.

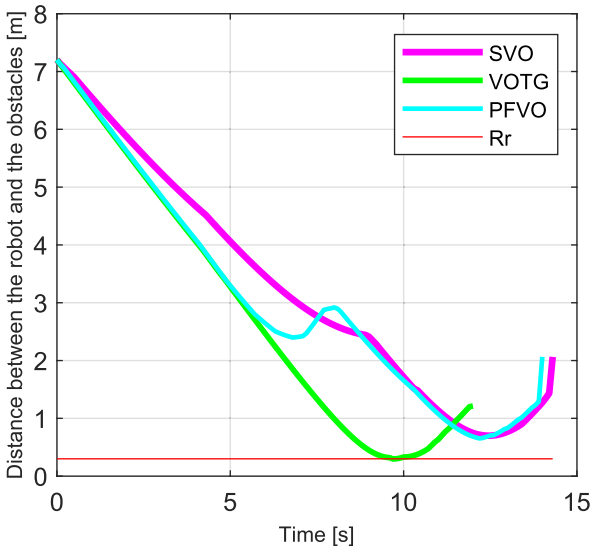

FIGURE 9. Minimum distances between the agent and the obstacles considering the different strategies.

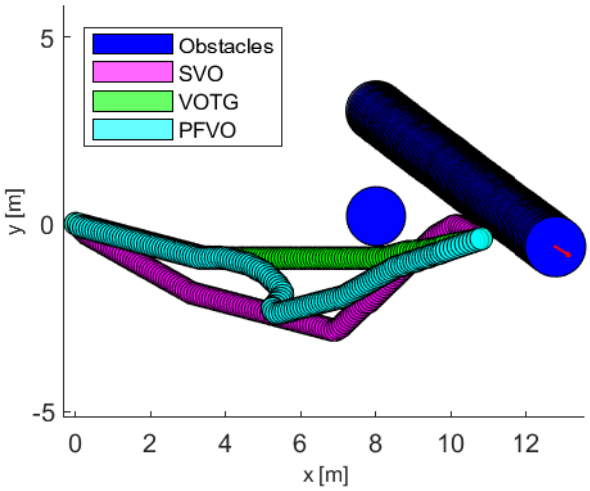

FIGURE 10. Result of the motion of the agent in the third example.

a set of tools and libraries for efficient communication and development of complex robot software. The Slamtec company provides a simplified BSD licensed ROS2 package alongside the purchased LiDAR.

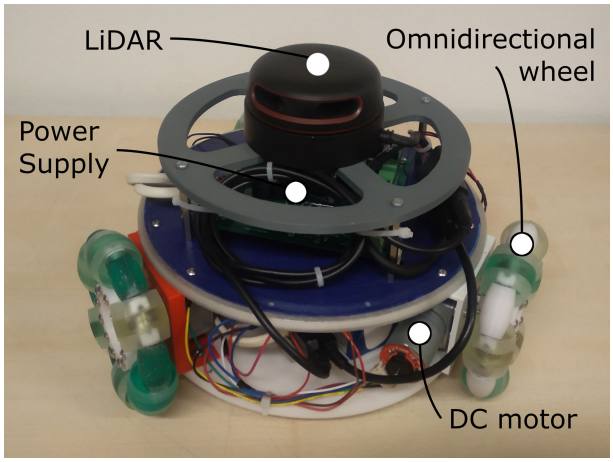

FIGURE 11. Omnidirectional mobile robot.

The ROS2 network consists of three nodes as can be seen in Figure 12. The node named "*sllidar_node*" handles the LiDAR by controlling its motor and forwarding incoming data to the selected communication interface through the "scan" topic. We used the provided C++ file in its original state without modification for this project. The scan topic delivers messages of type "*sensor_msgs/LaserScan*" to the subscribers. It can be observed that the orientations are mapped to the $(-\pi;\pi]$ rad range and stored in a separate array along with the corresponding distance values. The node called "*base_station*" subscribes to the scan topic and is created under the MATLAB-based user interface. The message received from the scan topic contains the minimum and maximum angles, the size of the increments between them, and the distance values. Each distance value corresponds to a specific angle. Using this data, elementary mathematical operations can be used to generate Cartesian coordinates in the Descartes coordinate system. Due to the placement of the laser scanner on the robot, corrections, rotations, and reflections were required to ensure that the resulting image has the appropriate orientation in the coordinate system used in the application, where the forward direction corresponds to the positive direction of the y-axis. The "*base_station*" node sends pulse width values for controlling the motors to the "*control_client*" node through the "control" topic. We transmitted the published data in

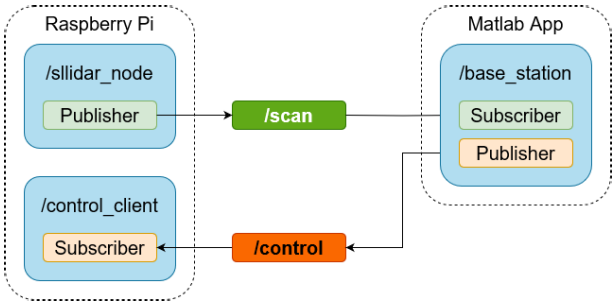

FIGURE 12. Connection of the ROS2 nodes.

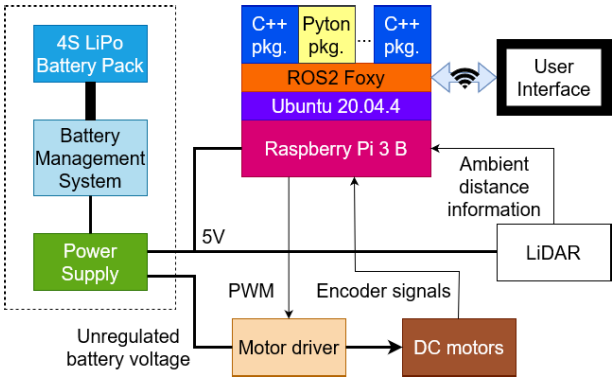

FIGURE 13. Structure of the mobile agent.

the "*std_msgs/UInt8MultiArray*" format. The MultiArray message type is quite flexible, allowing the developer to freely determine the type and amount of data contained in the homogeneous array. In this case, the data package contains an eight-element, unsigned, eight-bit integer array, specifying the movement of the three motors in different directions. The "*control_client*" node receives control signals from the user interface and forwards them to the motor controllers through the Raspberry Pi GPIO connectors. We used the PIGPIO library to access the GPIO pins. It was important to ensure that if the connection between the robot and the controlling computer is interrupted due to any malfunction, the system should stop quickly, within 200ms in this case. We achieved this using software-based timed interrupts. Launch files assist in the configuration and launch of the nodes. These files can be in Python, XML, or YAML format. The configuration file was created for this project based on the project launch files published by Slamtec mentioned earlier. In this file, we parameterized the "*sllidar_node*" to send the LiDAR data to the scan topic via UDP port, and we also initialized the "*control_client*" node.

The different parts and the connections of the mobile agent can be seen in Figure 13.

C. EXPERIMENTAL RESULTS

The Particle Filter Velocity Obstacle (PFVO) algorithm was subjected to real-world testing using the robot specified in Section V-B. As an obstacle within the robot's workspace, a differential-driven robot was employed. This obstacle was programmed to follow a black line on the ground.

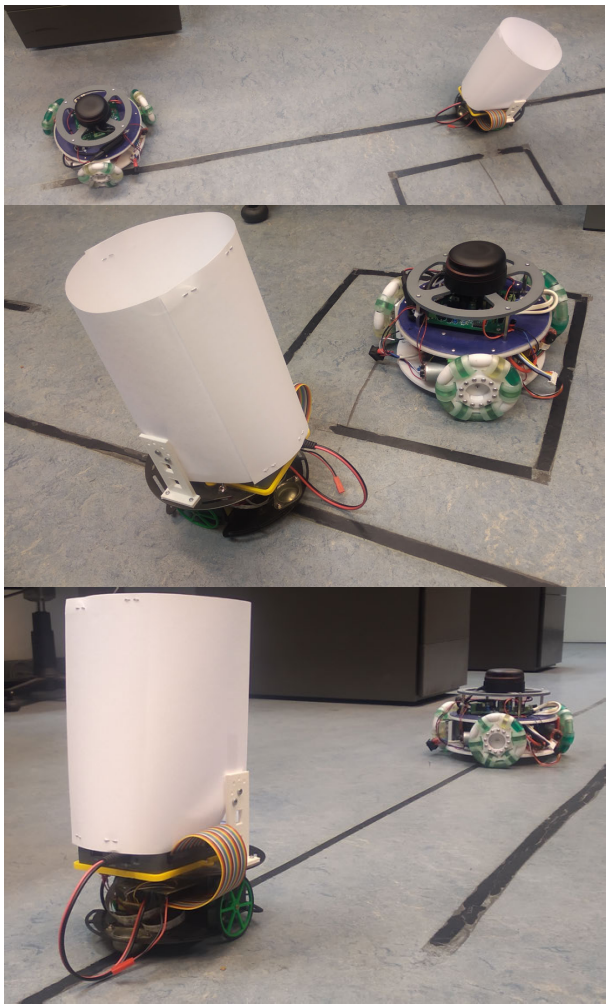

FIGURE 14. Real test.

An extension, termed a 'hat,' was added to the differential-driven robot to ensure the LiDAR sensor's laser measurements were reflected appropriately; initially, the robot's height was insufficient for this purpose. This modification enabled the agent to consistently obtain information about the obstacle at every time step, thus facilitating accurate estimation of the obstacle's position and velocity vector and ensuring collision-free goal attainment. To distinguish the obstacle from the environment, the segmentation method previously described was utilized, which effectively differentiates the obstacle from its surroundings.

Figure 14 illustrates both the obstacle and the agent's trajectory. Of note is the agent's ability to execute an evasive maneuver in close proximity to the obstacle, owing to the continuous reception of sensor measurement data from the obstacle. This capability is highlighted in the central part of Figure 14.

VI. CONCLUSION

In conclusion, this study has successfully introduced a novel state perception technique for mobile robots, using a Particle Filter. This method has shown its capability to accurately

perceive the position and velocity vectors of obstacles within the agent's workspace. The calculated uncertainty degree, derived from the distribution of particles and changes in velocity vectors, has proven effective when applied in an objective-function-based velocity selection method. Testing through various simulations and real-time experiments has validated the robustness and effectiveness of the proposed algorithm. When compared with two alternative motion planning algorithms, the proposed method has demonstrated better performance.

This research opens up new opportunities for further exploration and refinement in the domain of state estimation algorithms and collision-free motion planning for mobile robots in dynamic environments. In the future, we plan to extend the algorithm for differential-driven mobile robots too and the introduced method could also be used in multi-agent scenarios.

REFERENCES

[1] O. M. Gul and A. M. Erkmen, "Energy-efficient cluster-based data collection by a UAV with a limited-capacity battery in robotic wireless sensor networks," *Sensors*, vol. 20, no. 20, p. 5865, Oct. 2020.

[2] Y. Huang, L. Chen, and P. H. A. J. M. van Gelder, "Generalized velocity obstacle algorithm for preventing ship collisions at sea," *Ocean Eng.*, vol. 173, pp. 142–156, Feb. 2019.

[3] O. M. Gul, A. M. Erkmen, and B. Kantarci, "UAV-driven sustainable and quality-aware data collection in robotic wireless sensor networks," *IEEE Internet Things J.*, vol. 9, no. 24, pp. 25150–25164, Dec. 2022.

[4] J. Simanek, M. Reinstein, and V. Kubelka, "Evaluation of the EKF-based estimation architectures for data fusion in mobile robots," *IEEE/ASME Trans. Mechatronics*, vol. 20, no. 2, pp. 985–990, Apr. 2015.

[5] Y. Chen, S. Huang, and R. Fitch, "Active SLAM for mobile robots with area coverage and obstacle avoidance," *IEEE/ASME Trans. Mechatronics*, vol. 25, no. 3, pp. 1182–1192, Jun. 2020.

[6] H. Zhang, J. Wen, Y. Liu, W. Luo, and N. Xiong, "Mobile robot localization based on gradient propagation particle filter network," *IEEE Access*, vol. 8, pp. 188475–188487, 2020.

[7] A. Doucet, S. Godsill, and C. Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering," *Statist. Comput.*, vol. 10, pp. 197–208, Jul. 2000.

[8] Z. Zhong, H. Meng, and X. Wang, "Extended target tracking using an IMM based Rao–Blackwellised unscented Kalman filter," in *Proc. 9th Int. Conf. Signal Process.*, Oct. 2008, pp. 2409–2412.

[9] I. Arasaratnam, S. Haykin, and R. J. Elliott, "Discrete-time nonlinear filtering algorithms using Gauss–Hermite quadrature," *Proc. IEEE*, vol. 95, no. 5, pp. 953–977, May 2007.

[10] S. I. Roumeliotis and G. A. Bekey, "Bayesian estimation and Kalman filtering: A unified framework for mobile robot localization," in *Proc. IEEE Intl. Conf. Robot. Autom.*, vol. 3, Apr. 2000, pp. 2985–2992.

[11] H. Ahmad and T. Namerikawa, "Extended Kalman filter-based mobile robot localization with intermittent measurements," *Syst. Sci. Control Eng.*, vol. 1, no. 1, pp. 113–126, Dec. 2013.

[12] S. Y. Chen, "Kalman filter for robot vision: A survey," *IEEE Trans. Ind. Electron.*, vol. 59, no. 11, pp. 4409–4420, Nov. 2012.

[13] L. Jetto, S. Longhi, and G. Venturini, "Development and experimental validation of an adaptive extended Kalman filter for the localization of mobile robots," *IEEE Trans. Robot. Autom.*, vol. 15, no. 2, pp. 219–229, Apr. 1999.

[14] E. Kiriy and M. Buehler, "Three-state extended Kalman filter for mobile robot localization," McGill Univ., Montreal, QC, Canada, Tech. Rep. TR-CIM, p. 23, 2002, vol. 5.

[15] L. Chen, H. Hu, and K. McDonald-Maier, "EKF based mobile robot localization," in *Proc. 3rd Int. Conf. Emerg. Secur. Technol.*, Sep. 2012, pp. 149–154.

[16] L. Teslic, I. Skrjanc, and G. Klancar, "EKF-based localization of a wheeled mobile robot in structured environments," *J. Intell. Robotic Syst.*, vol. 62, no. 2, pp. 187–203, May 2011.

[17] L. D'Alfonso, W. Lucia, P. Muraca, and P. Pugliese, "Mobile robot localization via EKF and UKF: A comparison based on real data," *Robot. Auto. Syst.*, vol. 74, pp. 122–127, Dec. 2015.

[18] F. Martinelli, "Robot localization: Comparable performance of EKF and UKF in some interesting indoor settings," in *Proc. 16th Medit. Conf. Control Autom.*, Jun. 2008, pp. 499–504.

[19] P. G. Medewar, M. Yadav, and H. G. Patel, "A comparison between nonlinear estimation based algorithms for mobile robot localizations," in *Proc. IEEE 1st Int. Conf. Energy, Syst. Inf. Process. (ICESIP)*, Jul. 2019, pp. 1–6.

[20] N. Ko and T.-Y. Kuc, "Fusing range measurements from ultrasonic beacons and a laser range finder for localization of a mobile robot," *Sensors*, vol. 15, no. 5, pp. 11050–11075, May 2015.

[21] M. N. Rosenbluth and A. W. Rosenbluth, "Monte Carlo calculation of the average extension of molecular chains," *J. Chem. Phys.*, vol. 23, no. 2, pp. 356–359, Feb. 1955.

[22] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear and linear Bayesian state estimation," *IEE Proc. F, Radar Signal Process.*, vol. 140, no. 2, pp. 107–113, 1993.

[23] F. Gustafsson, "Particle filter theory and practice with positioning applications," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 25, no. 7, pp. 53–82, Jul. 2010.

[24] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Proc. 14th Int. Symp.*, vol. 70. Cham, Switzerland: Springer, 2011, pp. 3–19.

[25] S. Kamburugamuve, H. He, G. Fox, and D. Crandall, "Cloud-based parallel implementation of SLAM for mobile robots," in *Proc. Int. Conf. Internet Things Cloud Comput.*, Mar. 2016, pp. 1–7.

[26] Y. K. Tee and Y. C. Han, "LiDAR-based 2D SLAM for mobile robot in an indoor environment: A review," in *Proc. Int. Conf. Green Energy, Comput. Sustain. Technol. (GECOST)*, Jul. 2021, pp. 1–7.

[27] A. Yusefi, A. Durdu, M. F. Aslan, and C. Sungur, "LSTM and filter based comparison analysis for indoor global localization in UAVs," *IEEE Access*, vol. 9, pp. 10054–10069, 2021.

[28] F. Zhang, S. Li, S. Yuan, E. Sun, and L. Zhao, "Algorithms analysis of mobile robot SLAM based on Kalman and particle filter," in *Proc. 9th Int. Conf. Model., Identificat. Control (ICMIC)*, Jul. 2017, pp. 1050–1055.

[29] D. Törnqvist, T. B. Schön, R. Karlsson, and F. Gustafsson, "Particle filter SLAM with high dimensional vehicle model," *J. Intell. Robotic Syst.*, vol. 55, nos. 4–5, pp. 249–266, Aug. 2009.

[30] D. Claes, D. Hennes, K. Tuyls, and W. Meeussen, "Collision avoidance under bounded localization uncertainty," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 1192–1198.

[31] Z. Ai, M. Ma, and B. Zhao, "Distributed multi-robot collision avoidance under localization uncertainty," in *Proc. 3rd Int. Symp. Robot. Intell. Manuf. Technol. (ISRIMT)*, Sep. 2021, pp. 672–679.

[32] J. Petereit, T. Emter, C. W. Frey, T. Kopfstedt, and A. Beutel, "Application of hybrid A* to an autonomous mobile robot for path planning in unstructured outdoor environments," in *Proc. 7th German Conf. Robot.*, Munich, Germany, May 2012, pp. 1–6.

[33] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Dept. Comput. Sci., Iowa State Univ., Ames, IA, USA, Tech. Rep., 98–11, 1998.

[34] I. Noreen, A. Khan, H. Ryu, N. L. Doh, and Z. Habib, "Optimal path planning in cluttered environment using RRT-AB," *Intell. Service Robot.*, vol. 11, no. 1, pp. 41–52, Jan. 2018.

[35] Y. Dong, E. Camci, and E. Kayacan, "Faster RRT-based nonholonomic path planning in 2D building environments using skeleton-constrained path biasing," *J. Intell. Robotic Syst.*, vol. 89, nos. 3–4, pp. 387–401, Mar. 2018.

[36] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robot. Autom. Mag.*, vol. 4, no. 1, pp. 23–33, Mar. 1997.

[37] O. Brock and O. Khatib, "High-speed navigation using the global dynamic window approach," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 1999, pp. 341–346.

[38] B. Wu, X. Chi, C. Zhao, W. Zhang, Y. Lu, and D. Jiang, "Dynamic path planning for forklift AGV based on smoothing A* and improved DWA hybrid algorithm," *Sensors*, vol. 22, no. 18, p. 7079, Sep. 2022.

[39] J. H. Reif and H. Wang, "Social potential fields: A distributed behavioral control for autonomous robots," *Robot. Auto. Syst.*, vol. 27, no. 3, pp. 171–194, May 1999.

[40] G. Li, Y. Tamura, A. Yamashita, and H. Asama, "Effective improved artificial potential field-based regression search method for autonomous mobile robot path planning," *Int. J. Mechatronics Autom.*, vol. 3, no. 3, pp. 141–170, 2013.
[41] B. Kovács, G. Szayer, F. Tajti, M. Burdelis, and P. Korondi, "A novel potential field method for path planning of mobile robots by adapting animal motion attributes," *Robot. Auto. Syst.*, vol. 82, pp. 24–34, Aug. 2016.
[42] H.-T. Chiang, N. Malone, K. Lesser, M. Oishi, and L. Tapia, "Path-guided artificial potential fields with stochastic reachable sets for motion planning in highly dynamic environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 2347–2354.
[43] C. Qixin, H. Yanwen, and Z. Jingliang, "An evolutionary artificial potential field algorithm for dynamic path planning of mobile robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2006, pp. 3331–3336.
[44] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *Int. J. Robot. Res.*, vol. 17, no. 7, pp. 760–772, Jul. 1998.
[45] D. Wilkie, J. van den Berg, and D. Manocha, "Generalized velocity obstacles," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2009, pp. 5573–5578.
[46] J. Snape, J. van den Berg, S. J. Guy, and D. Manocha, "Smooth and collision-free navigation for multiple robots under differential-drive constraints," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2010, pp. 4584–4589.
[47] T. Battisti and R. Muradore, "A velocity obstacles approach for autonomous landing and teleoperated robots," *Auto. Robots*, vol. 44, no. 2, pp. 217–232, Jan. 2020.
[48] J. Van Den Berg, J. Snape, S. J. Guy, and D. Manocha, "Reciprocal collision avoidance with acceleration-velocity obstacles," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 3475–3482.
[49] J. Snape, J. V. D. Berg, S. J. Guy, and D. Manocha, "The hybrid reciprocal velocity obstacle," *IEEE Trans. Robot.*, vol. 27, no. 4, pp. 696–706, Aug. 2011.
[50] J. A. Douthwaite, S. Zhao, and L. S. Mihaylova, "Velocity obstacle approaches for multi-agent collision avoidance," *Unmanned Syst.*, vol. 7, no. 1, pp. 55–64, Jan. 2019.
[51] F. Vesentini and R. Muradore, "Velocity obstacle-based trajectory planner for two-link planar manipulators," in *Proc. Eur. Control Conf. (ECC)*, Jun. 2021, pp. 690–695.
[52] S. N. J. Poonganam, B. Gopalakrishnan, V. S. B. K. Avula, A. K. Singh, K. M. Krishna, and D. Manocha, "Reactive navigation under non-parametric uncertainty through Hilbert space embedding of probabilistic velocity obstacles," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 2690–2697, Apr. 2020.
[53] J. van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2008, pp. 1928–1935.
[54] B. Gopalakrishnan, A. K. Singh, M. Kaushik, K. M. Krishna, and D. Manocha, "PRVO: Probabilistic reciprocal velocity obstacle for multi robot navigation under uncertainty," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 1089–1096.
[55] A. Timcenko and P. Allen, "Probability-driven motion planning for mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 1994, pp. 2784–2789.
[56] L. B. P. Nascimento, D. Barrios-Aranibar, V. G. Santos, D. S. Pereira, W. C. Ribeiro, and P. J. Alsina, "Safe path planning algorithms for mobile robots based on probabilistic foam," *Sensors*, vol. 21, no. 12, p. 4156, Jun. 2021.
[57] L. Blackmore, H. Li, and B. Williams, "A probabilistic approach to optimal robust path planning with obstacles," in *Proc. Amer. Control Conf.*, 2006, p. 7.
[58] S. Thompson, T. Horiuchi, and S. Kagami, "A probabilistic model of human motion and navigation intent for mobile robot path planning," in *Proc. 4th Int. Conf. Auto. Robots Agents*, Feb. 2009, pp. 663–668.
[59] Z. Gyenes and E. G. Szádeczky-Kardoss, "Particle filter based obstacle's position estimation using LiDAR measurement data," in *Proc. Workshop Adv. Inf. Technol.*, 2021, pp. 97–102.
[60] Z. Gyenes and E. G. Szádeczky-Kardoss, "Particle filter-based perception method for obstacles in dynamic environment of a mobile robot," in *Proc. 25th Int. Conf. Methods Models Autom. Robot. (MMAR)*, Aug. 2021, pp. 97–102.
[61] Z. Gyenes and E. G. Szadeczky-Kardoss, "Motion planning for mobile robots using the safety velocity obstacles method," in *Proc. 19th Int. Carpathian Control Conf. (ICCC)*, May 2018, pp. 389–394.
[62] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Sci. Robot.*, vol. 7, no. 66, May 2022, Art. no. eabm6074.

ZOLTÁN GYENES received the B.Sc. and M.Sc. degrees in electrical engineering and the Ph.D. degree in robotics from the Budapest University of Technology and Economics, in 2017, 2019, and 2023, respectively. During the Ph.D. studies, he introduced several reactive motion planning methods for mobile agents, focusing on the safety of the agent and the environment. He is currently working on state estimation and AI methods in the field of robotics. His research interest includes autonomous motion planning for automated guided vehicles in a dynamic environment. He was a recipient of the DAAD Short-Term Research Scholarship, in 2021, and the Fulbright Visiting Researcher Scholarship, in 2022. He received the Best Student Paper Award from the ISMCR Conference, in 2020.

BARNABÁS PAJKOS received the B.Sc. degree in electrical engineering from the Budapest University of Technology and Economics, in 2023, where he is currently pursuing the M.Sc. degree in electrical engineering. During the bachelor's studies, he focused on the development of an omnidirectional mobile robot, which served as the subject of his paper. His current academic pursuits revolve around power electronics design and electronic control systems.

LADISLAU BÖLÖNI (Senior Member, IEEE) received the B.Sc. degree from the Technical University of Cluj-Napoca, Romania, and the Ph.D. degree from Purdue University. He was a Visiting Researcher with the Computer and Automation Research Institute, the Hungarian Academy of Sciences, the University of Rome La Sapienza, Imperial College, and the KTH Royal Institute of Technology. He is currently a Professor of computer science with the University of Central Florida. His research interest includes AI technologies with applications in robotics and the IoT.

EMESE GINCSAINÉ SZÁDECZKY-KARDOSS received the Ph.D. degree in electrical engineering from the Budapest University of Technology and Economics. She is currently an Associate Professor with the Department of Control Engineering and Information Technology, Budapest University of Technology and Economics. Her research interests include trajectory planning and control of mechatronic systems, robots, and autonomous vehicles.