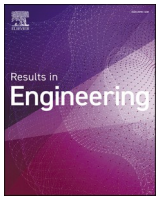
Results in Engineering 23 (2024) 102625

Contents lists available at ScienceDirect

Results in Engineering

journal homepage: www.sciencedirect.com/journal/results-in-engineering



Optimizing autonomous navigation in unknown environments: A novel trap avoiding vector field histogram algorithm VFH+T

Husam A. Neamah [a,b,c,*], Elek Donát [a], Péter Korondi [a]

[a] Department of Electrical Engineering and Mechatronics, Faculty of Engineering, University of Debrecen, Ótemető Utca 2-4, 4028, Debrecen, Hungary
[b] Technical Engineering College, Al-Ayen University, Thi-Qar 64001, Iraq
[c] Department of Business Management, Al-imam University College, Balad, Iraq

ARTICLE INFO

Keywords:
Autonomous systems navigation
Gap following
Obstacle avoidance and mobile robot

ABSTRACT

The Vector Field Histogram Plus (VFH+) algorithm is a cornerstone in robotic navigation, renowned for its efficiency and straightforward implementation across a multitude of environments. Despite its widespread utility, the algorithm's inherent limitations in handling complex obstacle entrapments necessitate refinement. This paper presents an advanced iteration, designated as VFH + T, which incorporates sophisticated memory-based trap recognition and avoidance mechanisms. This enhancement facilitates dynamic adjustment of navigation strategies through the integration of geometrical rules that retrospectively inform path planning decisions. Moreover, the VFH + T algorithm intricately melds the robotic platform's kinematic and dynamic constraints, optimizing real-time navigational commands based on both current sensory input and historical environmental interactions. Empirical simulations validate the enhanced algorithm's proficiency in circumventing navigational traps, improving operational safety and efficiency. Comparative analysis with VFH+ and VFH* algorithms show up to 17 % reduction in traveling distance due to the trap-avoidance technique during navigation. This advancement holds significant implications for enhancing autonomous navigation technologies in various practical applications, from self-driving vehicles to robotic aids in logistics and service industries.

1. Introduction

Autonomous navigation technologies are rapidly becoming more prevalent in various industries and everyday life, thus positioning them as a critical area of research. A prime example is the widely popular development of self-driving cars. Additionally, simpler navigation systems employed in autonomous underwater vehicles (AUVs) and unmanned aerial vehicles (UAVs) are increasingly utilized for tasks like rescue operations and exploration in hazardous environments [1–4]. The field of mobile robotics, where testing is comparatively less costly, less risky, and simpler, often serves as a testing ground for new navigation methods. Moreover, mobile robots are now more frequently found in academic settings, warehouses, and delivery services, necessitating a diverse range of navigation algorithms tailored to their specific designs and roles [5–7].

Mobile robot navigation comprises two fundamental components: global path planning and local obstacle avoidance [8]. Global path planners employ various optimization techniques to define the ideal route to a target, optimizing based on criteria like path length, travel time, and energy requirements. This optimization, however, requires comprehensive knowledge of the environment to be fully effective, which also demands significant computational resources. Conversely, local obstacle avoidance algorithms focus on immediate navigational responses rather than planning a complete trajectory. These systems rapidly compute motion commands directly from real-time sensor data, allowing them to operate at high frequencies and adapt quickly to environmental changes. However, their reliance solely on immediate sensor inputs predisposes them to become ensnared in dead-end situations, known in the field as "local minima".

Among global path planners, the heuristic A* algorithm [9] as well as iterative sampling methods such as the Rapidly Exploring Random Tree (RRT) [10] and Probabilistic Roadmap (PRM) [11] are most commonly employed [12]. These techniques effectively generate a planned path by evaluating multiple possible routes to determine the most optimal. More recent examples of global path planning methods with metaheuristic algorithms are showcased in Ref. [13].

On the local navigation front, methods such as the Dynamic Window Approach (DWA) [14] are frequently utilized. The DWA calculates the

* Corresponding author.
E-mail address: husam@eng.unideb.hu (H.A. Neamah).

https://doi.org/10.1016/j.rineng.2024.102625
Received 22 May 2024; Received in revised form 22 July 2024; Accepted 24 July 2024
Available online 26 July 2024

best travel direction and speed from a set of possible velocities that are kinematically feasible, factoring in a cost function to guide decision-making. However, this method is notably vulnerable to getting stuck in local minima, where the robot faces a dead-end or looping path. Another local method, the Nearness Diagram (ND) [15], categorizes navigation challenges into distinct scenarios, providing specific motion commands tailored to each situation. As a method designed to deal with extremely cluttered environments, however, it has a tendency to deflect towards free areas. The Time Elastic Band (TEB) method [16] which is designed for dual-layer navigation systems, slightly adjusts the pre-determined global plan to achieve a kinematically feasible trajectory for mobile robots. However, this reliance on the global planner reduces its flexibility, particularly in unpredictable situations. The Artificial Potential Field (APF) [17] method is one of the earliest local navigation algorithms, using sensor data to detect obstacles which then exert a repulsive force, while the target exerts an attractive force. The vector sum of these forces determines the robot's travel direction. Despite its simplicity and widespread adoption, the APF method has notable limitations. These include susceptibility to getting trapped in local minima, oscillations near obstacles, difficulties navigating narrow passages, and challenges in navigating closely spaced obstacles or reaching targets located near obstacles [18].

To address the shortcomings of the APF, the Vector Field Histogram (VFH) [19] was developed by Borenstein and Koren. The VFH operates within an angular coordinate system, constructing a polar histogram based on the surrounding obstacle density. It then identifies valleys — areas devoid of obstacles — and selects the one closest to the intended direction. The centre of this selected valley becomes the designated travel path. This two-stage data reduction technique mitigates issues such as oscillations and difficulties navigating through narrow spaces, like doorframes. Following the development of VFH, an enhanced version named VFH + [20] was introduced, refining the process for selecting travel directions. It incorporates a cost function that accounts for deviations from the target direction, previous steering direction, and the robot's current velocity. This modification ensures more decisive and consistent directional commitment during navigation. Additionally, the algorithm adjusts for the robot's physical dimensions and a safety buffer by enlarging the representation of obstacles, thus improving navigation around non-point obstacles. It also simplifies the consideration of the robot's kinematics, using circular trajectory arcs to represent potential travel paths. Another development, the VFH* algorithm, integrates the foundational VFH framework with an A* look-ahead planner [21]. VFH*, transcending its local navigation roots, anticipates the effects of each potential steering decision by projecting them into the future and building histograms at these forecasted locations. This proactive analysis helps circumvent simple trap scenarios that pose challenges for VFH+, enhancing the robot's ability to navigate complex environments effectively. On the other hand, the effectiveness of the VFH* algorithm is significantly influenced by the setting of the projection distance parameter. Extending this parameter increases computational demands, and even with optimal adjustments, the variety of trap situations that can be avoided remains limited. To address these limitations, a variant of VFH* featuring time-dependent trees has been introduced [22].

Overall, the VFH+ is a widely used local navigation method, favoured for its flexibility, minimal parameter tuning requirements [23], simple structure, and capability for high-frequency operation. This popularity has spurred various enhancements to the original model. For instance, VFH + D was developed to introduce decaying occupancy values, enhancing navigation in dynamic environments [24,25], Additional enhancements include the integration of Bezier curves for path generation [26], and the incorporation of neural networks [27], fuzzy logic [28] or neuro-fuzzy algorithm [29] to refine the system's decision-making processes. Given that the original VFH and VFH+ were designed primarily for ultrasonic sensors, which are prone to high measurement errors, subsequent modifications have been proposed to adapt the system for use with more precise laser scanners [30,31]. A novel adaptation, the Vector Polar Histogram (VPH) [32], combines the VFH framework with the potential field method, utilizing laser range sensor measurements. This approach was further refined in the Enhanced Vector Polar Histogram (VPH+) [33], which organizes detected obstacles into groups to boost detection efficiency. VPH + has been improved further by integrating model predictive control to incorporate vehicle kinematics into the navigation process [34]. Additional developments include the integration of the Velocity Obstacle (VO) method and polar scan matching for enhanced dynamic obstacle avoidance [35], and the application of the VFH algorithm with a Kalman predictor for improved dynamic obstacle navigation [36]. These enhancements generally aim to overcome static and dynamic trap avoidance challenges, often caused by line-of-sight obstructions from certain positions. Typically, these require reliance on either global navigational information or historical data to effectively circumvent detected obstacles.

The above discussed navigation algorithms all offer only conditional/partial solutions to the problem of trap avoidance. Motivated to define generally applicable rules, which allow successful navigation and trap avoidance in unknown environments (no access to global information), the focus of this paper is to introduce improvements to the popular VFH + algorithm and extend the capabilities of the above listed VFH-type navigation methods and enhance navigation efficiency without the integration of complex global planning systems.

The paper introduces two novel concepts, applicable independently from each other, each aiming to improve a different aspect of the VFH + algorithm. The first technique focuses on the trap avoidance and involves transforming detected trap situations into simplified geometric shapes, which are stored in the robot's memory. Based on this data, a binary polar histogram is constructed at each position of the robot, mapping out potential paths that lead to known traps. To circumvent these paths, the robot's temporary navigation target is dynamically adjusted according to the histogram, steering it away from potential dead-ends. Additionally, the cost function for selecting the optimal steering direction has been updated to include information regarding detected traps. These features allow the algorithm to proactively modify the robot's route to avoid both current and potential future traps, enhancing both the safety and efficiency of navigation. In contrast to existing approaches that augment VFH+ with high-level global planners, this solution provides a standalone capability that simplifies the navigation system's architecture by eliminating the need for additional planning layers. This streamlined approach not only reduces system complexity but also empowers the robot to adapt dynamically to new or changing environments without the requirement for pre-mapped data. This refined approach represents a significant advancement in providing a more efficient and adaptable solution for navigating complex environments using the VFH + algorithm. This improvement is anticipated to enhance the practical application of robotic navigation systems.

The second enhancement of the original VFH + framework includes an additional histogram layer, named dynamic constraint histogram, which generates a detailed histogram indicating feasible steering directions. This histogram is derived from a comprehensive kinematic and dynamic model of the robot, which offers a more precise navigation process compared to the basic VFH + method, which only employs a minimum turning radius parameter to represent the vehicle's constraints. The feasibility of a steering command is evaluated based on a binary histogram, which disables every angle past the predefined minimum turning radius parameter. Our approach instead considers the current motion state (velocity, acceleration and wheel angular velocities) of the mobile robot, providing a more accurate analog histogram representing the feasibility of each traveling direction. Similarly to how the obstacle distances provide a weight to each direction in the original VFH + histogram, the steering time required to reach each orientation also applies a weight to them through the dynamic constraint histogram. Thus, the optimal traveling direction is determined by considering both

aspects. Furthermore, the values of the dynamic constraint histogram are calculated from a mathematical vehicle model, which can be changed to any arbitrary model. This way different vehicle constructions as well as models with different level of detail can be utilized in the decision making (according to implementation circumstances), unlike the fixed simplification the original VFH + employs with the minimal turning radius parameter.

The paper is structured as follows: Section 2 presents the problem statement and assumptions of the paper. Section 3 outlines the kinematic and dynamic models utilized for the omnidirectional robot. Section 4 details the proposed navigation method. The simulation results are analysed in Section 5, and the conclusions are summarized in Section 6.

2. Problem statement and assumptions

Let us define a general navigation task based on the following. The mobile robot is placed in a random position inside an unknown workspace. We define an $\mathscr{F}_w(x_w;y_w)$ world coordinate system with its centre point fixed to the current position (the starting position of the navigation task) and the x_w axis pointing in the starting orientation of the robot. We define a navigation target $P_{tar}(x_{tar};y_{tar})$ in the world-coordinate system, which the robot must reach without colliding with the objects inside the workspace. The mobile robot is represented by its circumcircle with a radius of r_r. During the operation we assume that the position, velocity and acceleration as well as the wheel angular velocities of the robot can be measured with acceptable deviation through odometry sensors and localization. While the localization method is outside the scope of this article, there are countless state-of-the-art solutions in the literature, such as [37].

The environment is perceived through a LiDAR scan, which provides an array of depth points in the form of measured distance $R_{mes}(\psi_i)$ at angle ψ_i (where $i \in N \mid i \leq n_{scan}$ and n_{scan} is the number of measured depth points). The accuracy of the laser scan ranges between 0.1 and 2 cm, which is satisfying for applications in mobile robotics. Since the proposed trap-avoiding method links the measured depth points into groups, the crucial factor of the measurement accuracy lies in the resolution of the laser scan. If the measured points are too far from each other the system can't distinguish free areas between close obstacle (e.g. doors) from solid obstacles. Consequently, the proposed algorithm is functional up to a maximum range of r_{max}, depending on the resolution of the scan (the number of measured points) and calculated based on eq. (1).

$$r_{max} = \frac{r_r}{tan^{-1}(2\pi/n_{scan})} \tag{1}$$

It should be noted that measurement and processing errors can occur near walls almost parallel to the laser beams. However, because the surrounding environment is represented by polar obstacle blocks in the algorithm, the proposed method is highly compatible with camera detections. Employing a camera-LiDAR sensor fusion can greatly reduce any measurement error and improve the robustness of the proposed algorithm, but even a single LiDAR scan is enough for the appropriate functioning of the algorithm as presented in the results of this paper. In the scope of this article the operation $\Delta_A(.)$ denotes the angles difference between 2 given angle values ϕ_1, ϕ_2 based on (2).

$$\Delta_A(\phi_1;\phi_2) = \begin{cases} 2\pi - |\phi_1 - \phi_2|, if\ |\phi_1 - \phi_2| > \pi \\ |\phi_1 - \phi_2|, otherwise \end{cases} \tag{2}$$

3. Omnidirectional robot model and movement control

3.1. Kinematic and dynamic model

As mentioned in the introduction, the proposed dynamic constraint histogram method requires the mathematical model of the chosen vehicle. In this paper we will present the technique through the example of a three-wheeled omnidirectional robot. This model is grounded in established studies that explore the kinematic and dynamic modelling of such robots [38,39]. The design features a symmetrically aligned circular shape, known as a Kiwi drive, equipped with three omnidirectional wheels. The wheels are numbered counterclockwise with an angle of $2/3\ \pi$ between each adjacent wheel. The radius and mass of the robot is r_r and m_r respectively. Let us define an $\mathscr{F}_m(x_m;y_m)$ moving coordinate system parallel to $\mathscr{F}_w$ with its origo at the centre of gravity (CoG) of the robot, and an $\mathscr{F}_b(x_b;y_b)$ body coordinate system with its origo identical to $\mathscr{F}_m$, and the axis x_b pointing to the contact point of the first wheel. The angle between the axes x_b and x_m represents the ϕ_r orientation of the robot. Angles $\theta_1 = \phi_r$; $\theta_2 = \phi_r + 2/3\ \pi$ and $\theta_3 = \phi_r - 2/3\ \pi$ denote the direction of each wheel in $\mathscr{F}_m$ in respect to x_m. Besides, 3 individual wheel coordinate systems are defined as $\mathscr{F}_{wh_i}(x_{wh_i};y_{wh_i}) \mid i = 1,2,3$ based on the conversion defined by eq. (3).

$$\begin{pmatrix} x_{wh_i} \\ y_{wh_i} \\ 0 \end{pmatrix} = \begin{bmatrix} cos(\theta_i) & sin(\theta_i) & 0 \\ -sin(\theta_i) & cos(\theta_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_m \\ y_m \\ 0 \end{bmatrix} = R_{m\to wh_i} \times \begin{pmatrix} x_m \\ y_m \\ 0 \end{pmatrix} \tag{3}$$

The vectors $\vec{P}_r = [x_r, y_r, \phi_r]^T$; $\vec{v}_r = [v_x, v_y, \omega_r]^T$ and $\vec{a}_r = [a_x, a_y, \beta_r]^T$ represent the position, velocity and acceleration of the robot in the world coordinate system respectively ($\vec{v}_{lin} = [v_x, v_y]^T$ is the linear velocity vector, ω_r is the angular velocity, $\vec{a}_{lin} = [a_x, a_y]^T$ is the linear acceleration and β_r is the angular acceleration). The modelled system consists of 4 rigid bodies: 3 wheels and the main robot body linked together by bearing joints. The velocity of each wheel contact point in $\mathscr{F}_m$ and $\mathscr{F}_{wh_i}$ coordinate systems is calculated based on eqs. (4) and (5) respectively.

$$\vec{v}_{c_i}{}^{m} = \begin{pmatrix} v_x \\ v_y \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \omega_r \end{pmatrix} \times \begin{pmatrix} r_r\ cos(\theta_i) \\ r_r\ sin(\theta_i) \\ 0 \end{pmatrix} = \begin{pmatrix} v_x - \omega_r r_r\ sin(\theta_i) \\ v_y + \omega_r r_r\ cos(\theta_i) \\ 0 \end{pmatrix} \tag{4}$$

$$\vec{v}_{c_i}{}^{wh_i} = R_{m\to wh_i} \times \vec{v}_{c_i}{}^{m} = \begin{pmatrix} v_x\ cos(\theta_i) + v_y\ sin(\theta_i) \\ -v_x\ sin(\theta_i) + v_y\ cos(\theta_i) + \omega_r r_r \\ 0 \end{pmatrix} \tag{5}$$

Similarly, the acceleration of each contact point in the moving and wheel coordinate systems can be calculated from eqs. (6) and (7).

$$\vec{a}_{c_i}{}^{m} = \begin{pmatrix} a_x \\ a_y \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \beta_r \end{pmatrix} \times \begin{pmatrix} r_r\ cos\ \theta_i \\ r_r\ sin\ \theta_i \\ 0 \end{pmatrix} - \omega_r{}^2 \begin{pmatrix} r_r\ cos\ \theta_i \\ r_r\ sin\ \theta_i \\ 0 \end{pmatrix} = \begin{pmatrix} a_x - \beta_r r_r\ sin\ \theta_i - \omega_r{}^2 r_r\ cos\ \theta_i \\ a_y + \beta_r r_r\ cos\ \theta_i - \omega_r{}^2 r_r\ sin\ \theta_i \\ 0 \end{pmatrix} \tag{6}$$

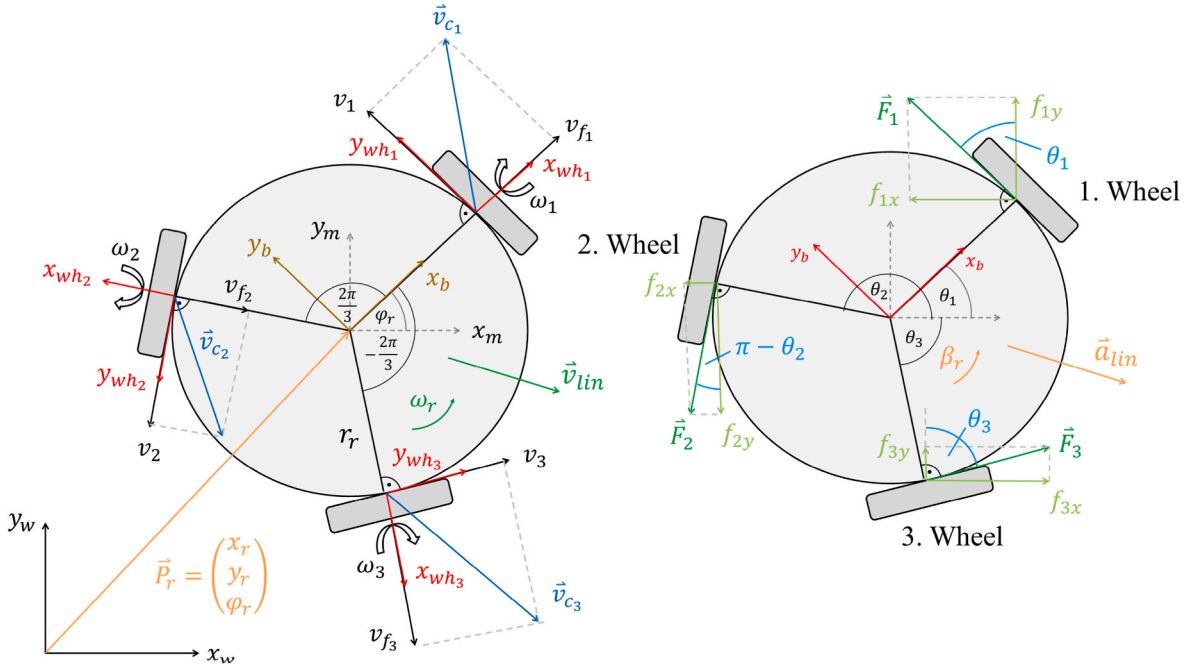


Fig. 1. Kinematic and dynamic model of the 3-wheeled omnidirectional robot.

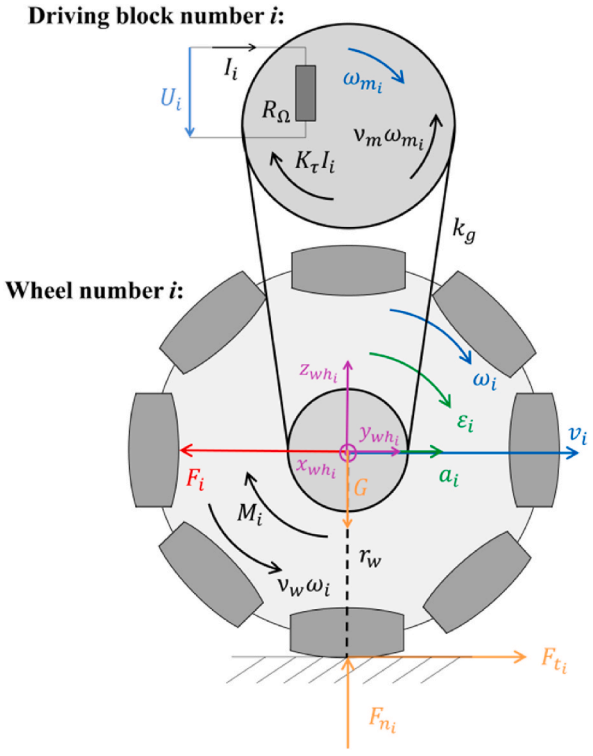


Fig. 2. Dynamic model of the driving blocks and wheels.

$$\vec{a}_{c_i}^{\,wh_i} = R_{m\to wh_i} \times \vec{a}_{c_i}^{\,m} = \begin{pmatrix} a_x\cos(\theta_i) + a_y\sin(\theta_i) - {\omega_r}^2 r_r \\ -a_x\sin(\theta_i) + a_y\cos(\theta_i) + \beta_r r_r \\ 0 \end{pmatrix} \tag{7}$$

The velocity vector $\vec{v}_{c_i}^{\,wh_i}$ of each wheel contact point has a tangential component v_i, which is generated by the rotation of wheel number i and a free sliding radial component v_{fi} generated by the two other wheels. Assuming no slipping at the ground contact point, the $\vec{v}_{c_i}^{\,wh_i}$ velocity of the wheel and the acceleration vector $\vec{a}_{c_i}^{\,wh_i}$ of the contact point can be expressed with eqs. (8) and (9) respectively.

$$\vec{v}_{c_i}^{\,wh_i} = \begin{pmatrix} v_{fi} \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} \omega_i \\ 0 \\ 0 \end{pmatrix} \times \begin{pmatrix} 0 \\ 0 \\ r_w \end{pmatrix} = \begin{pmatrix} v_{fi} \\ -\omega_i r_w \\ 0 \end{pmatrix} = \begin{pmatrix} v_{fi} \\ v_i \\ 0 \end{pmatrix} \tag{8}$$

$$\vec{a}_{c_i}^{\,wh_i} = \begin{pmatrix} a_{fi} \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} \varepsilon_i \\ 0 \\ 0 \end{pmatrix} \times \begin{pmatrix} 0 \\ 0 \\ r_w \end{pmatrix} = \begin{pmatrix} a_{fi} \\ -\varepsilon_i r_w \\ 0 \end{pmatrix} = \begin{pmatrix} a_{fi} \\ a_i \\ 0 \end{pmatrix} \tag{9}$$

Where r_w is the radius and ω_i is the angular velocity of the wheel. Substituting eq. (5) into eq. (8) yields the kinematic connection between the main robot body and the 3 wheels defined by eq. (10).

$$\begin{pmatrix} v_x\cos(\theta_i) + v_y\sin(\theta_i) \\ -v_x\sin(\theta_i) + v_y\cos(\theta_i) + \omega_r r_r \\ 0 \end{pmatrix} = \begin{pmatrix} v_{fi} \\ v_i \\ 0 \end{pmatrix} \tag{10}$$

At each wheel, the tangential component v_i can be controlled by the angular velocity of the wheel. Based on eq. (10) the tangential velocity of each contact point can be calculated from the robot velocity vector by eq. (11).

$$\begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} = \begin{bmatrix} -\sin(\theta_1) & \cos(\theta_1) & r_r \\ -\sin(\theta_2) & \cos(\theta_2) & r_r \\ -\sin(\theta_3) & \cos(\theta_3) & r_r \end{bmatrix} \times \begin{bmatrix} v_x \\ v_y \\ \omega_r \end{bmatrix} = R_{v_w\to v_i} \times \begin{bmatrix} v_x \\ v_y \\ \omega_r \end{bmatrix} \tag{11}$$

Similarly, the tangential acceleration a_i of each wheel contact point can be expressed as eqs. (12) and (13) derives.

$$\begin{pmatrix} a_x\cos(\theta_i) + a_y\sin(\theta_i) - {\omega_r}^2 r_r \\ -a_x\sin(\theta_i) + a_y\cos(\theta_i) + \beta_r r_r \\ 0 \end{pmatrix} = \begin{pmatrix} a_{fi} \\ a_i \\ 0 \end{pmatrix} \tag{12}$$

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{bmatrix} -sin(\theta_1) & cos(\theta_1) & r_r \\ -sin(\theta_2) & cos(\theta_2) & r_r \\ -sin(\theta_3) & cos(\theta_3) & r_r \end{bmatrix} \times \begin{bmatrix} a_x \\ a_y \\ \beta_r \end{bmatrix} = R_{v_w \to v_i} \times \begin{bmatrix} a_x \\ a_y \\ \beta_r \end{bmatrix} \quad (13)$$

Each driving block consists of a DC motor with electrical resistance R_Ω and torque constant of K_τ. The output of the motors is transferred to the shafts of the wheels through a toothed belt drive with a gear ratio of k_g. Since the time constant of the electrical system is approximately 2 magnitudes smaller than the time constant of the mechanical system, the transient behavior (inductivity) of the DC motor is neglected, and only the steady state equation of (14) is used.

$$I_i = \frac{U_i}{R_\Omega} - \frac{K_\tau}{R_\Omega} * \omega_{m_i} \quad (14)$$

Where U_i is the input voltage and I_i is the current ω_{m_i} is the angular velocity and ν_m is the viscous friction coefficient of the motor. The equations of an ideal belt drive produce the output torque (M_i) of the driving block defined by eqs. (15) and (16).

$$M_i = k_g\left(K_\tau * I_i - \nu_m * \omega_{m_i}\right) \quad (15)$$

$$\omega_{m_i} = k_g * \omega_i \quad (16)$$

Assuming pure rolling with no slipping the dynamic equations of wheel number i are given by eqs. (17) and (18).

$$m_w * a_i = F_{t_i} - F_i \quad (17)$$

$$J_w * \frac{a_i}{r_w} = M_i - \nu_w * \omega_i - F_{t_i} r_w \quad (18)$$

Where m_w is the mass and J_w is the inertia around the x_{wh_i} axis of the wheel; ν_w is the viscous friction coefficient; F_{t_i} is the static friction force at the ground contact point and F_i is the force acting on the wheel joint (see Fig. 1). The inertia of the DC motor was neglected in the calculation. The model of the driving blocks can be seen on Fig. 2.

The condition of the pure rolling is described by eq. (19).

$$\left|F_{t_i}\right| \le \mu_t F_{n_i} = \mu_t G = \mu_t\left(m_w + \frac{m_r}{3}\right) g \quad (19)$$

Where μ_t is the static friction coefficient between the wheel and the ground. Expressing F_{t_i} from (17) and substituting it along with eq. (15) into (18) results the dynamic equation of the driving block shown in eq. (21). Substituting eq. (16) into (21) and expressing F_i yields eq. (22) defining the magnitude of wheel joint forces based on the current motion state of the mobile robot.

$$\left(\frac{J_w}{r_w} + m_w r_w\right) * a_i = k_g * K_\tau * I_i - k_g * \nu_m * \omega_{m_i} - \nu_w * \omega_i - F_i * r_w \quad (21)$$

$$F_i = k_g \frac{K_\tau}{r_w} * I_i - \left(\frac{J_w}{{r_w}^2} + m_w\right) * a_i - \left(\frac{\nu_w + {k_g}^2 \nu_m}{r_w}\right) * \omega_i \quad (22)$$

3.2. Simulation model and parameters

Eq. (20) describes the effect of the current motion state on the vehicles next feasible action and will be used for building the dynamic constraint histogram. Outside the navigation algorithm, the motion simulations of the virtual experiments are based on the following. The forces acting on the wheel joints (F_i) are controlling the motion of the robot, based on eq. (23). Expressing the acceleration from (23) vector yields eq. (24).

Table 1
Parameter values used in the simulation.

Parameter	Value	Unit	Parameter	Value	Unit
R_Ω	6	$[\Omega]$	K_τ	0.035	$\left[\frac{Nm}{A}\right]$
ν_w	$8.5*10^{-4}$	$[Nms]$	ν_m	$1.163*10^{-4}$	$[Nms]$
k_g	3	–	μ_t	0.4	–
m_w	0.2	$[kg]$	m_r	10	$[kg]$
r_w	0.05	$[m]$	r_r	0.2	$[m]$
J_w	$2.5*10^{-4}$	$[kgm^2]$	J_r	0.2	$[kgm^2]$

$$\begin{pmatrix} m_r a_x \\ m_r a_y \\ J_r \beta_r \end{pmatrix} = {R_{v_w \to v_i}}^{-1} \times \begin{bmatrix} F_1 \\ F_2 \\ F_3 \end{bmatrix} = \begin{bmatrix} -sin(\theta_1) & -sin(\theta_2) & -sin(\theta_3) \\ cos(\theta_1) & cos(\theta_2) & cos(\theta_3) \\ r_r & r_r & r_r \end{bmatrix} \times \begin{bmatrix} F_1 \\ F_2 \\ F_3 \end{bmatrix} \quad (23)$$

$$\begin{pmatrix} a_x \\ a_y \\ \beta_r \end{pmatrix} = \frac{1}{m_r} * \begin{bmatrix} -sin(\theta_1) & -sin(\theta_2) & -sin(\theta_3) \\ cos(\theta_1) & cos(\theta_2) & cos(\theta_3) \\ \frac{m_r r_r}{J_r} & \frac{m_r r_r}{J_r} & \frac{m_r r_r}{J_r} \end{bmatrix} \times \begin{bmatrix} F_1 \\ F_2 \\ F_3 \end{bmatrix} \quad (24)$$

Where J_r is the inertia of the robot around the z_m axis. The velocity and position of the robot in the subsequent time step are calculated based on eqs. (25) and (26).

$$\begin{pmatrix} v_x[k] \\ v_y[k] \\ \omega_r[k] \end{pmatrix} = \begin{pmatrix} v_x[k-1] \\ v_y[k-1] \\ \omega_r[k-1] \end{pmatrix} + t \begin{pmatrix} a_x[k] \\ a_y[k] \\ \beta_r[k] \end{pmatrix} \quad (25)$$

$$\begin{pmatrix} x_r[k] \\ y_r[k] \\ \phi_r[k] \end{pmatrix} = \begin{pmatrix} x_r[k-1] \\ y_r[k-1] \\ \phi_r[k-1] \end{pmatrix} + \frac{t}{2} \begin{pmatrix} v_x[k-1] + v_x[k] \\ v_y[k-1] + v_y[k] \\ \omega_r[k-1] + \omega_r[k] \end{pmatrix} \quad (26)$$

Where k is the index of the current time step and t is the elapsed time between two subsequent time steps. The parameter values used in the simulations can be seen in Table 1. The values presented in Table 1 were selected based on established parameters from the kinematic and dynamic modeling of omnidirectional mobile robots as detailed in Ref. [40]. This reference provides a comprehensive analysis of drivetrain parameters, ensuring that our simulation model is grounded in validated and reliable data.

3.3. The individual wheel commands

The proposed navigation algorithm issues desired traveling direction (δ_{des}) and speed (u_{des}) commands. To achieve the motion, these need to be converted into motor voltage inputs by a low-level controller. The design of this controller is outside of the scope of this article, so in the simulations a simple PI controller was deployed to link the navigation algorithm and the robot model. The structure of the implemented low-level controller is as follows: first the desired velocity vector is calculated based on eq. (27).

$$\vec{v}_{des} = \begin{pmatrix} u_{des} * cos(\delta_{des}) \\ u_{des} * sin(\delta_{des}) \\ P_{ori} * (\delta_{des} - \phi_r) \end{pmatrix} \quad (27)$$

Where P_{ori} is the proportional gain of the orientation controller. This is converted into reference angular velocity commands for each wheel by eq. (28).

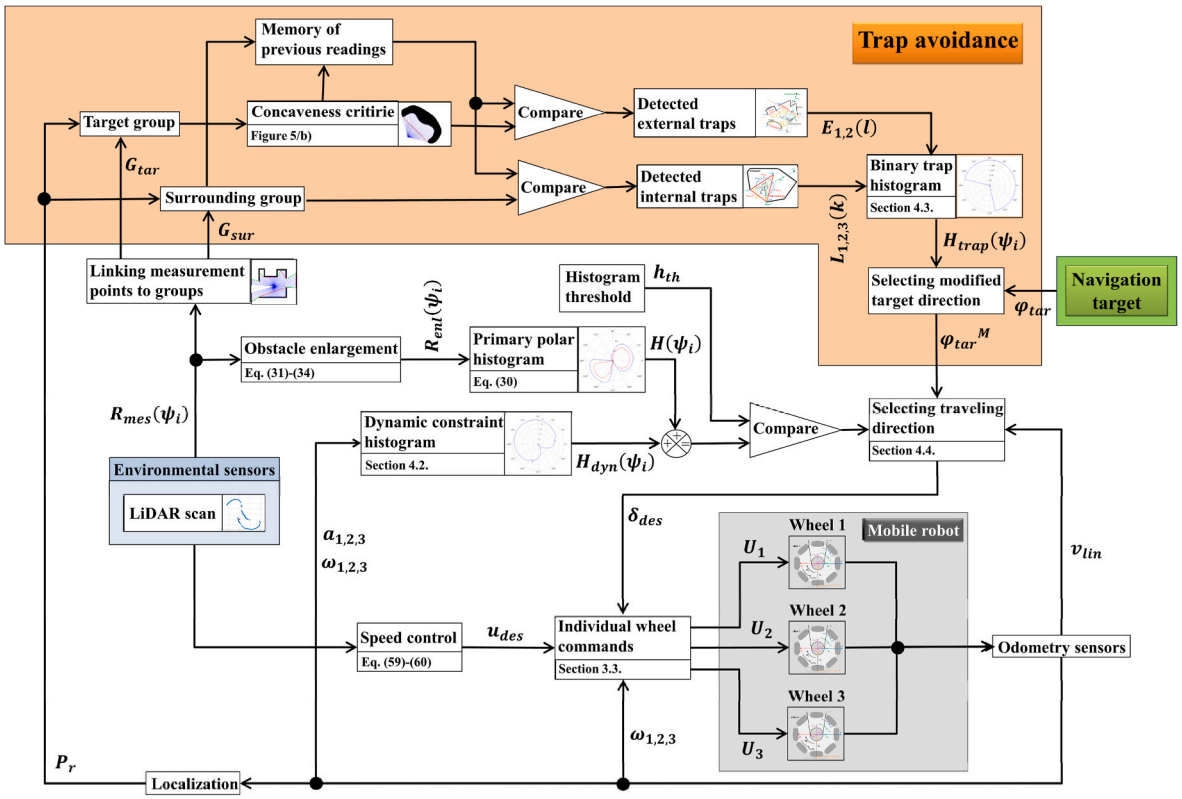


Fig. 3. Robotic navigation control block diagram of the proposed VFH + T algorithm using LiDAR-integrated polar and dynamic histograms for enhanced trap detection and path adjustment.

$$\vec{\omega}_{ref} = \frac{1}{r_w}\left(R_{v_w \to v_i} \times \vec{v}_{des}\right) \tag{28}$$

The individual motor input voltages are then given by a PI controller based on the angular velocities of the motors, as eq. (29) describes.

$$\begin{pmatrix} U_1 \\ U_2 \\ U_3 \end{pmatrix} = P_\omega * \left(\vec{\omega}_{ref} - \begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{pmatrix}\right) + I_\omega * \int \left(\vec{\omega}_{ref} - \begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{pmatrix}\right) dt \tag{29}$$

Where P_ω and I_ω are the proportional and integral gains of the motor controllers. For the robot model defined by the parameters in Table 1, the empirically tuned $P_{ori} = 0.2$; $P_\omega = 1.2$ and $P_\omega = 0.15$ values yield satisfactory results. The voltage commands are saturated to the range of $[-12; 12]$ [V].

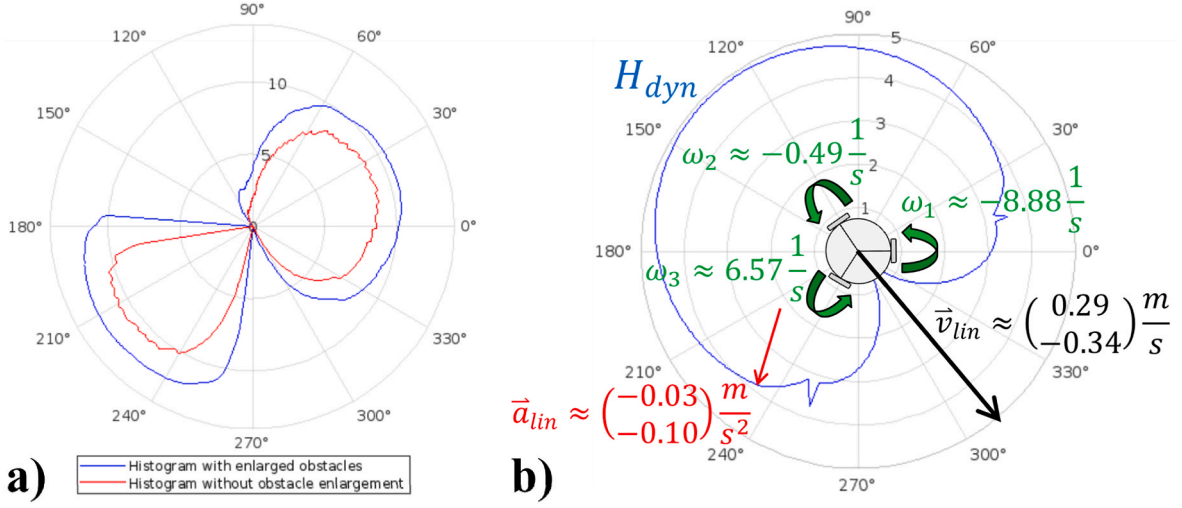


Fig. 4. a): Polar histogram with obstacle enlargement: the obstacle enlargement not only compensates the robot's size but also acts as a filter and smooths the primary polar histogram. b): Dynamic constraint histogram: values are proportional to the time required to reach every potential traveling direction based on the given motion state of the vehicle.

4. The navigation algorithm

4.1. The primary polar histogram

The methodology for determining the desired traveling direction in the proposed system utilizes three distinct histograms generated by the algorithm. A block diagram outlining this system is depicted in Fig. 3. The primary histogram used is a polar histogram, closely mirroring the approach of the original VFH + method. The difference in this version is the direct utilization of current distance measurements from laser scans instead of the traditional Cartesian histogram grid. This advancement leverages the higher accuracy of modern laser sensors as opposed to the ultrasonic sensors used in the initial VFH + design. Ultrasonic sensors, while functional, offer less precision, leading the Cartesian grid to function primarily as a filter to separate valid detections from erroneous ones. With the advent of precise laser-based distance measurement technologies, constructing histograms directly from laser scan data has become feasible, eliminating the need for a filtering grid and streamlining the process of histogram construction.

Based on the utilized LiDAR sensor and the measured depth points $R_{mes}(\psi_i)$ at angles ψ_i, the primary polar histogram H is constructed using (30).

$$H(\psi_i) = \begin{cases} a_{hist} - b_{hist} * R(\psi_i), if\ R(\psi_i) \le d_{max} \\ 0, otherwise \end{cases} \tag{30}$$

Where $a_{hist}; b_{hist}$ are coefficients of the histogram and d_{max} is the radius of the active window (the maximum distance, where measurement points effect the navigation decision). Another difference from the standard VFH+ is the use of a circular shaped active window instead of the original rectangle shape. The measured obstacle cells are enlarged by a radius of $r_r + r_{safe}$; where r_{safe} is the required safety distance from obstacles to account for the robot's size. The enlargement caused by the measured obstacle point at angle θ_j (where $j \in N \mid j \le n_{scan}$) is calculated by eqs. (31) and (32).

$$R(\psi_i) = \begin{cases} R_{enl}(\psi_i), if\ \psi_i \in \left[\psi_j - \gamma_{enl}(\psi_j); \psi_j + \gamma_{enl}(\psi_j)\right]\ and R_{mes}(\psi_i) > R_{enl}(\psi_i) \\ R_{mes}(\psi_i), otherwise \end{cases} \tag{31}$$

$$R_{enl}(\psi_i) = \begin{cases} r_r + r_{safe}, if\ R_{mes}(\psi_j) < (r_r + r_{safe}) \\ R_{mes}(\psi_j) cos\left(\Delta_A(\psi_j; \psi_i)\right) - \xi_{i,j}, otherwise \end{cases} \tag{32}$$

Where $\gamma_{enl}(\psi_j)$ is the enlargement angle of the measured point at angle ψ_j and $\xi_{i,j}$ is the distance correction, defined by eqs. (33) and (34) respectively.

$$\gamma_{enl}(\psi_j) = sin^{-1}\left(\frac{r_r + r_{safe}}{R_{mes}(\psi_j)}\right) \tag{33}$$

$$\xi_{i,j} = \sqrt{\left(cos^2\left(\Delta_A(\psi_j; \psi_i)\right) - 1\right) R_{mes}(\psi_j) + (r_r + r_{safe})^2} \tag{34}$$

The effect of the obstacle enlargement on the primary polar histogram can be seen on Fig. 4/a).

4.2. The dynamic constraint histogram

The original VFH + algorithm introduced a method that simplified the integration of vehicle dynamics into navigation. It assumed that the robot's trajectory would follow curves defined by dynamic and kinematic constraints, which include a minimum turning radius. Directions obstructed by obstacles along these minimal radius curves were classified as unreachable. To advance this foundational work, a new method involving the construction of a secondary histogram is proposed. This histogram calculates the time required to reach each potential traveling direction from the robot's current state of motion. This method is based on the physical model outlined in Section 3.1. but is adaptable to various vehicle models. During the histogram's development, each angle is evaluated as a potential direction for travel. The feasibility of reaching each direction from the robot's current state is analysed, enhancing the practicality and effectiveness of the navigation tool. The candidate velocity vector $\vec{v}_{can}$ of angle ψ_j is calculated based on eqs. (35) and (36).

$$\vec{v}_{can}(\psi_j) = u * \begin{pmatrix} cos(\psi_j + \phi_r) \\ sin(\psi_j + \phi_r) \end{pmatrix} \tag{35}$$

$$u = \begin{cases} \left|\vec{v}_{lin}\right| * cos\left(\Delta_A\left(\delta_{vel}; \psi_j + \phi_r\right)\right), if\ \left|\vec{v}_{lin}\right| * cos\left(\Delta_A\left(\delta_{vel}; \psi_j + \phi_r\right)\right) > v_{min} \\ v_{min}, otherwise \end{cases} \tag{36}$$

Where $\left|\vec{v}_{lin}\right|$ is the magnitude and δ_{vel} is the direction of the current velocity. Consequently, the required velocity change vector and the unit vector pointing in the direction of the velocity change $(\vec{e})$ are calculated from eqs. (37) and (38).

$$\vec{\Delta v}\left(\psi_j\right) = u * \begin{pmatrix} cos(\psi_j + \phi_r) \\ sin(\psi_j + \phi_r) \end{pmatrix} - \begin{pmatrix} v_x \\ v_y \end{pmatrix} \tag{37}$$

$$\vec{e}(\psi_j) = \begin{pmatrix} e_x \\ e_y \end{pmatrix} = \frac{\vec{\Delta v}\left(\psi_j\right)}{\left|\vec{\Delta v}\left(\psi_j\right)\right|} \tag{38}$$

The robot's maximum acceleration is limited by the exerted joint forces. Using (14) and (22) The joint force can be expressed as a function of the voltage input in eq. (39).

$$F_i = k_g \frac{K_\tau}{r_w R_\Omega} * U_i - \left(\frac{J_w}{{r_w}^2} + m_w\right) * a_i - \left(\frac{\nu_w + {k_g}^2 \nu_m}{r_w} + \frac{{k_g}^2 {K_\tau}^2}{r_w R_\Omega}\right) * \omega_i \tag{39}$$

The terms containing a_i and ω_i describe the effect of the robot's current kinematic and dynamic state on the maximum reachable acceleration ($i = 1, 2, 3$ is the wheel index). By changing the voltage input U_i to the lower and upper limits −12V and 12V the lower and upper bounds of the possible joint forces can be calculated from eq. (40). The s_i variable dictates which limit needs to be determined (positive or negative), based on (46).

$$F_i^{lim} = k_g \frac{K_\tau}{r_w R_\Omega} * s_i * U_{max} - \left(\frac{J_w}{{r_w}^2} + m_w\right) * a_i - \left(\frac{\nu_w + {k_g}^2 \nu_m}{r_w} + \frac{{k_g}^2 {K_\tau}^2}{r_w R_\Omega}\right) * \omega_i \tag{40}$$

Expressing the joint forces as a function of the robot's acceleration from (23) yields eq. (41), which generalized for wheel number i results eq. (42).

$$\begin{bmatrix} F_1 \\ F_2 \\ F_3 \end{bmatrix} = \begin{bmatrix} -sin(\theta_1) & cos(\theta_1) & r_r \\ -sin(\theta_2) & cos(\theta_2) & r_r \\ -sin(\theta_3) & cos(\theta_3) & r_r \end{bmatrix} \times \begin{bmatrix} m_r a_x \\ m_r a_y \\ J_r \beta_r \end{bmatrix} \tag{41}$$

$$F_i = m_r\left(-sin(\theta_i) * a_x + cos(\theta_i) * a_y\right) + J_r r_r * \beta_r \tag{42}$$

Eq. (42) describes the general relation between the wheel joint forces and the robot's acceleration. To calculate the dynamic constraint histogram, we need to apply this equation to each candidate acceleration vector $\vec{a}_{lin}(\psi_j)$, calculated by eq. (43).

$$\vec{a}_{lin}(\psi_j) = c * \vec{e}(\psi_j) \tag{43}$$

Where c is the magnitude of the maximum allowed acceleration in the

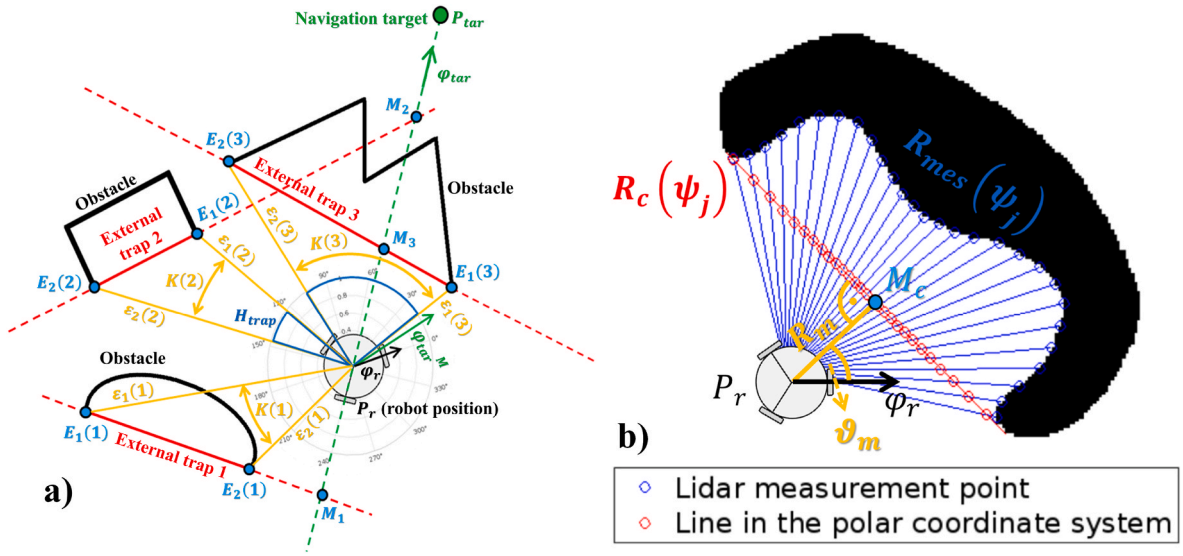


Fig. 5. Advanced geometric and polar coordination for trap detection in robotic navigation of the proposed VFH + T. a): Visualizes the external trap avoidance and trap detection process, highlighting how environmental data is linked with geometric analysis to enhance navigational accuracy. b): Demonstrates the concaveness criterion and integration of LiDAR measurements in the polar coordinate system, facilitating real-time obstacle mapping and contributing to the robot's adaptive navigation capabilities.

direction of $\vec{e}(\psi_j)$. Applying (43) to (42) gives eq. (44), where the values of $c_i(\psi_j) \mid i = 1,2,3$ represent the acceleration limit in the direction of ψ_j posed by wheel number i, calculated by eq. (45) if we assume no angular acceleration ($\beta_r = 0$).

$$F_i^{lim} = c_i(\psi_j)*m_r(-sin(\theta_i)*e_x + cos(\theta_i)*e_y) + J_r r_r*\beta_r \tag{44}$$

$$c_i(\psi_j) = \frac{F_i^{lim}}{m_r*(-sin(\theta_i)*e_x + cos(\theta_i)*e_y)} = \frac{F_i^{lim}}{\sigma_i(\psi_j)} \tag{45}$$

$$s_i = sign(\sigma_i(\psi_j)) \tag{46}$$

However, in certain cases the maximum linear acceleration of an omnidirectional robot can be increased by having a non-zero angular acceleration. The optimal angular acceleration based on the current motion state can be calculated as follows. First, we express $c_i(\psi_j)$ from (44) and (45) with non-zero β_r value, as eq. (47) shows.

$$c_i(\psi_j) = \frac{F_i^{lim}}{\sigma_i(\psi_j)} - \frac{J_r r_r}{\sigma_i(\psi_j)}*\beta_r \tag{47}$$

In the algorithm we calculate a $c_i(\psi_j)$ value for all 3 wheels at every ψ_j candidate angle, because every wheel will pose a limit to the maximum allowed acceleration in each direction. From these 3 limit values at a given angle $\psi_j = \psi_{fix}$ the lowest absolute value will determine the maximum achievable acceleration in that direction; therefore, we need to set β_r to a value which maximizes this. This can be achieved by taking the maximum and minimum values of $c_i(\psi_j = \psi_{fix})$: $c_{min}(\psi_j)$ and $c_{max}(\psi_j)$. If these 2 values have different sign increasing the angular acceleration will increase one and decrease the other value. Consequently, the maximum acceleration is achieved if $c_{min}(\psi_j) = c_{max}(\psi_j)$. If all $c_i(\psi_{fix})$ values have the same sign it is only necessary to consider $c_{min}(\psi_j)$, as described in eq. (48).

$$\beta_r = \begin{cases} \dfrac{F_{c_{min}}^{lim}\sigma_{c_{max}} - F_{c_{max}}^{lim}\sigma_{c_{min}}}{J_r r_r*(\sigma_{c_{max}} - \sigma_{c_{min}})}, if\ sign(c_{min}) \neq sign(c_{max}) \\ \dfrac{F_{c_{min}}^{lim}}{J_r r_r}, otherwise \end{cases} \tag{48}$$

Once the ideal angular acceleration is determined, the $c_i(\psi_j)$ values are updated based on (47). The maximum reachable acceleration from the current position in the direction of ψ_j is the minimum of the calculated $c_i(\psi_j) \mid i = 1,2,3$ values. In other words, each of the 3 wheels impose limitations on the maximum achievable acceleration and we use the most restrictive constraint for each candidate direction to build the dynamic constraint histogram H_{dyn} as eqs. (49) and (50) defines.

$$\Delta t(\psi_j) = \frac{|\vec{\Delta v}(\psi_j)|}{min(c_1(\psi_j), c_2(\psi_j), c_3(\psi_j))} \tag{49}$$

$$H_{dyn}(\psi_j) = d_{hist}*\Delta t(\psi_j) \tag{50}$$

Where $min(.)$ is the minimum function, d_{hist} is the dynamic histogram coefficient and $\Delta t(\psi_j)$ is the predicted time required to reach a velocity pointing to candidate direction ψ_j. It is important to note that the method is not limited to omnidirectional vehicles as the dynamic constraint histogram is built by (50), which can be applied to any arbitrary vehicle model (including models with different level of detail for other vehicle construction types, such as differential drive, Ackermann-steering, etc.). The dynamic constraint histogram for an example motion state of an omnidirectional mobile robot is illustrated in Fig. 4/b).

4.3. The trap avoidance

The original VFH + algorithm utilized an additively updated Cartesian histogram grid as a form of environmental memory. This historic data could assist in navigating the robot out of local minima, typically when a global planner identified such a situation. In contrast, our modified approach abandons the histogram grid in favour of storing environmental memory as geometric objects. These objects encapsulate past detected traps that the robot must avoid, enhancing the robot's navigational efficiency. The process of trap detection relies on a specific analysis of the current distribution of sensor measurements. Points from laser scans are analysed and linked to form extended obstacles within the polar coordinate system through a counter-clockwise review of the data. If the Cartesian coordinates of two adjacent points (P_i and P_{i+1}) are within a predefined threshold distance ($2r_r$), they are grouped together.

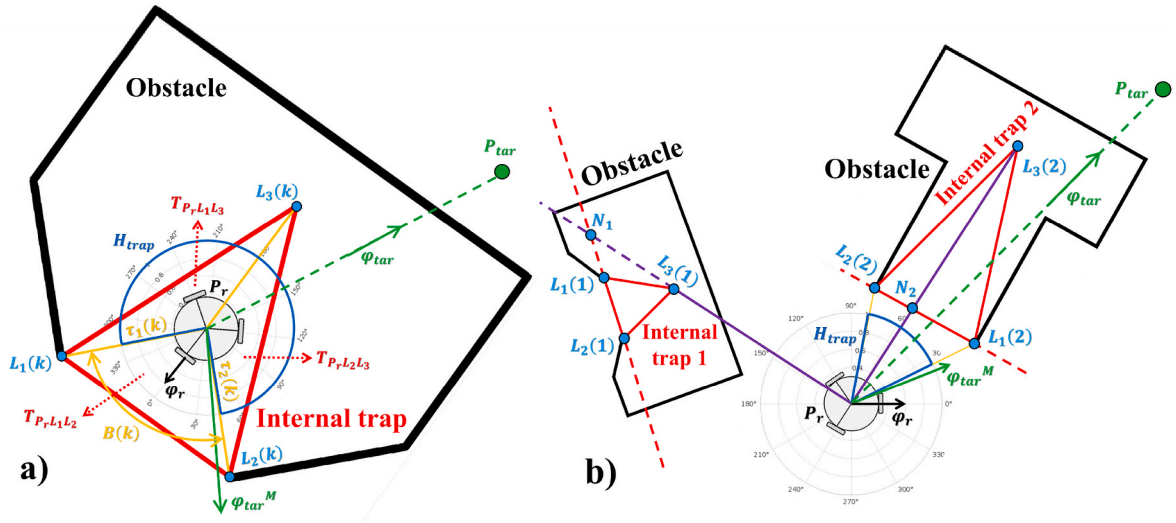


Fig. 6. Advanced computational geometry and polar coordinate analysis for ψ-trapezoidal trap detection in autonomous navigation algorithms. a): Algorithmic determination of boundary angles $\varepsilon_j(l)$ and $\tau_j(k)$ for internal trap avoidance using polar histograms H_{trap}. b): Geometric transformation of internal traps into external traps constraints, analysing endpoint vectors $L_1(k)$ and $L_2(k)$ for optimized pathfinding.

These groups form the basis for evaluating the current navigation scenario. Particularly, two significant groups are scrutinized: one that intersects a line from the robot's current position to the navigation target, designated as the target group G_{tar} and another that occupies more than half the azimuth, referred to as the surrounding group G_{sur} provided such a group exists in the scenario. These groups are crucial as they have the only realistic potential to lead to trap situations. Groups between the robot and the target (G_{tar}) are identified as potential external traps. These are confirmed as traps if their endpoints remain nearly constant over time and meet a concaveness criterion, suggesting they are sufficiently hollow to entrap the robot. The concaveness condition is the following: first a line in the polar coordinate system between the 2 endpoints of the obstacle group is constructed based on eq. (51). M_c denotes the closest point of this line to the robot position P_r and $(R_m; \vartheta_m)$ are the polar coordinates of M_c in the robot's body coordinate system $\mathcal{F}_b$ as illustrated on Fig. 5/b).

$$R_c(\psi_j) = R_m{}^*sec(\psi_j - \vartheta_m) \tag{51}$$

Subsequently, the calculated $R_c(\psi_j)$ distances are compared to the measured $R_{mes}(\psi_j)$ values inside the obstacle group G_{tar}. If the condition $R_c(\psi_j) + 0.1 < R_{mes}(\psi_j)$ is true for at least 80 % of the angles ψ_j inside G_{tar} the group is declared as concave and considered as a valid candidate for an external trap. The external trap are represented as lines between the two endpoints of the group $(E_1(l) = (x_1{}^{ext}(l); y_1{}^{ext}(l))$ and $E_2(l) = (x_2{}^{ext}(l); y_2{}^{ext}(l))$, where $l \in N \mid l \leq n_{ext}$ and n_{ext} is the total number of detected external traps).

The detection of internal traps is the following: if there is an obstacle group larger than half the azimuth (G_{sur}) for the observation time with near the same endpoint coordinates, the coordinates of those endpoints was saved, as well as the robots position at the time of the detection. These 3 points $(L_1(k) = (x_1{}^{int}(k); y_1{}^{int}(k)); L_2(k) = (x_2{}^{int}(k); y_2{}^{int}(k))$ and $L_3(k) = (x_3{}^{int}(k); y_3{}^{int}(k)) = (x_r; y_r)$, where $k \in N \mid k \leq n_{int}$ and n_{int} is the total number of detected internal traps) form a triangle which is the geometric representation of the internal trap. If an obstacle group occupies more than 70 % of the azimuth, basically surrounds the robot, it is immediately declared it as an internal trap.

Using the identified past trap situations of the memory a binary trap histogram is constructed based on the following. Let $A = [-\pi; \pi]$ be the set of angles in the polar coordinate system of the robot. For every detected external trap, the border angles $\varepsilon_j(l)$ is calculated as eq. (52) describes.

$$\varepsilon_j(l) = tan^{-1}\left(\frac{y_j{}^{ext}(l) - y_r}{x_j{}^{ext}(l) - x_r}\right) - \phi_r \;\middle|\; j = 1, 2 \tag{52}$$

The two angle values are switched if $\varepsilon_1(l) > \varepsilon_2(l)$ to ensure $\varepsilon_1(l)$ is the smaller value. The set of angles between $\varepsilon_1(l)$ and $\varepsilon_2(l)$ is denoted as $K(l)$ and it represents the traveling directions leading to the given trap situation from the current position, defined as eq. (53).

$$K(l) \in A \;\middle|\; K(l) = \begin{cases} [\varepsilon_1(l); \varepsilon_2(l)], \textit{if } |\varepsilon_2(l) - \varepsilon_1(l)| < \pi \\ [\varepsilon_1(l); \pi] \cup [-\pi; \varepsilon_2(l)], \textit{otherwise} \end{cases} \tag{53}$$

If the robots position is on the opposite half-plane of an external trap (defined by the line connecting $E_1(l)$ and $E_2(l)$) than the navigation target (P_{tar}), all sections $(\psi_i \mid i \leq n_{scan})$ in $K(l)$ are set to 1 as described in (54). M_l denotes the intersecting point of lines $\overline{E_1(l)E_2(l)}$ and $\overline{P_rP_{tar}}$. The process of avoiding external traps is illustrated on Fig. 5/a.

$$H_{trap}(\psi_i \in K(l)) = \begin{cases} 1, \textit{if } \left|\vec{P_rM_l}\right| + \left|\vec{M_lP_{tar}}\right| = \left|\vec{P_rP_{tar}}\right| \\ 0, \textit{otherwise} \end{cases} \tag{54}$$

In the case of internal traps, the boundary angles $\tau_1(k)$ and $\tau_2(k)$ are calculated similarly to (52) using the saved endpoints $L_1(k)$ and $L_2(k)$. The values are also switched if $\tau_1(k) > \tau_2(k)$ and the set of angles between $\tau_1(k)$ and $\tau_2(k)$ is denoted as $B(k)$, defined as eq. (55).

$$B(k) \in A \;\middle|\; B(k) = \begin{cases} [\tau_1(k); \tau_2(k)], \textit{if } |\tau_2(k) - \tau_1(k)| < \pi \\ [\tau_1(k); \pi] \cup [-\pi; \tau_2(k)], \textit{otherwise} \end{cases} \tag{55}$$

If the robot is inside the triangle defined by the 3 endpoints $\left(L_j(k) \mid j = 1, 2, 3\right)$, all sectors of the histogram outside $B(k)$ – in other words elements of the complement set $B^C(k) = A \backslash B(k)$ – are assigned with a value of 1, as eq. (56) indicates. To check whether or not point P_r is inside $L_1L_2L_3\Delta$, we compare the triangle areas $T_{P_rL_1L_2}; T_{P_rL_1L_3}$ and $T_{P_rL_2L_3}$ to $T_{L_1L_2L_3}$.

$$H_{trap}(\psi_i \in A \backslash B(k)) = \begin{cases} 1, \textit{if } T_{L_1L_2L_3} = T_{P_rL_1L_2} + T_{P_rL_1L_3} + T_{P_rL_2L_3} \\ 0, \textit{otherwise} \end{cases} \tag{56}$$

If the robot is outside of the triangle and in the opposite half plane (defined by the line connecting $L_1(k)$ and $L_2(k)$) than $L_3(k)$, the endpoints $L_1(k)$ and $L_2(k)$ are handled as an external trap, calculated as eq. (57). N_k denotes the intersecting point of lines $\overline{L_1(k)L_2(k)}$ and $\overline{P_rL_3(k)}$.

Internal trap avoidance is illustrated on Fig. 6.

$$H_{trap}(\psi_i \in B(k)) = \begin{cases} 1, if \left|\vec{P_r N_k}\right| + \left|\vec{N_k L_3}(k)\right| = \left|\vec{P_r L_3}(k)\right| \\ 0, otherwise \end{cases} \quad (57)$$

The evaluation of the binary trap histogram yields the momentary target direction (${\phi_{tar}}^M$). From the angles with a histogram value of 0 the one closest to the absolute navigation target direction (ϕ_{tar}) is chosen as the modified target direction. This means the target direction never points to the direction of any detected trap situation. This approach is more beneficial than blocking away directions which lead to trap situations as they are potentially useful temporary traveling direction. The logic behind the method is to not attract the agent towards traps, instead of forcing it away from them.

4.4. The desired traveling direction and traveling speed

The desired traveling direction of the robot is calculated from the resultant polar histogram $H_{res} = H + H_{dyn}$ similarly to the original VFH + algorithm. The angles where the histogram value is below a certain threshold h_{th} are considered as valleys. Each valley is denoted by its starting and finishing angles ${\vartheta_j}^{st}$ and ${\vartheta_j}^{en}$ (where $j \leq n_{val}$ and n_{val} is the number of detected valleys). A valley is defined as narrow if $\Delta_A\left({\vartheta_j}^{en}; {\vartheta_j}^{st}\right) < s_{lim}$ and wide otherwise. Wide valleys contain two potential traveling directions: $\delta_{j,1} = {\vartheta_j}^{st} + 0.5 s_{lim}$ and $\delta_{j,2} = {\vartheta_j}^{en} - 0.5 s_{lim}$, while narrow valleys contain one potential traveling direction $\delta_{j,1} = 0.5\left({\vartheta_j}^{en} + {\vartheta_j}^{st}\right)$. Additionally, if a valley contains the target angle it is considered as a third potential traveling direction $\delta_{j,3} = {\phi_{tar}}^M$. The desired traveling direction is chosen from these potential directions by an assigned cost function $G(\delta_{j,i}) \mid i = 1, 2, 3$; defined by eq. (58).

$$G(\delta_{j,i}) = \mu_1 {}^*\Delta_A\left({\phi_{tar}}^M; \delta_{j,i}\right) + \mu_2 {}^*\Delta_A\left(\delta_{vel}; \delta_{j,i}\right) + \mu_3 {}^*\Delta_A\left(\delta_{des}; \delta_{j,i}\right) + \mu_4 {}^*H_{trap}\left(\delta_{j,i}\right) \quad (58)$$

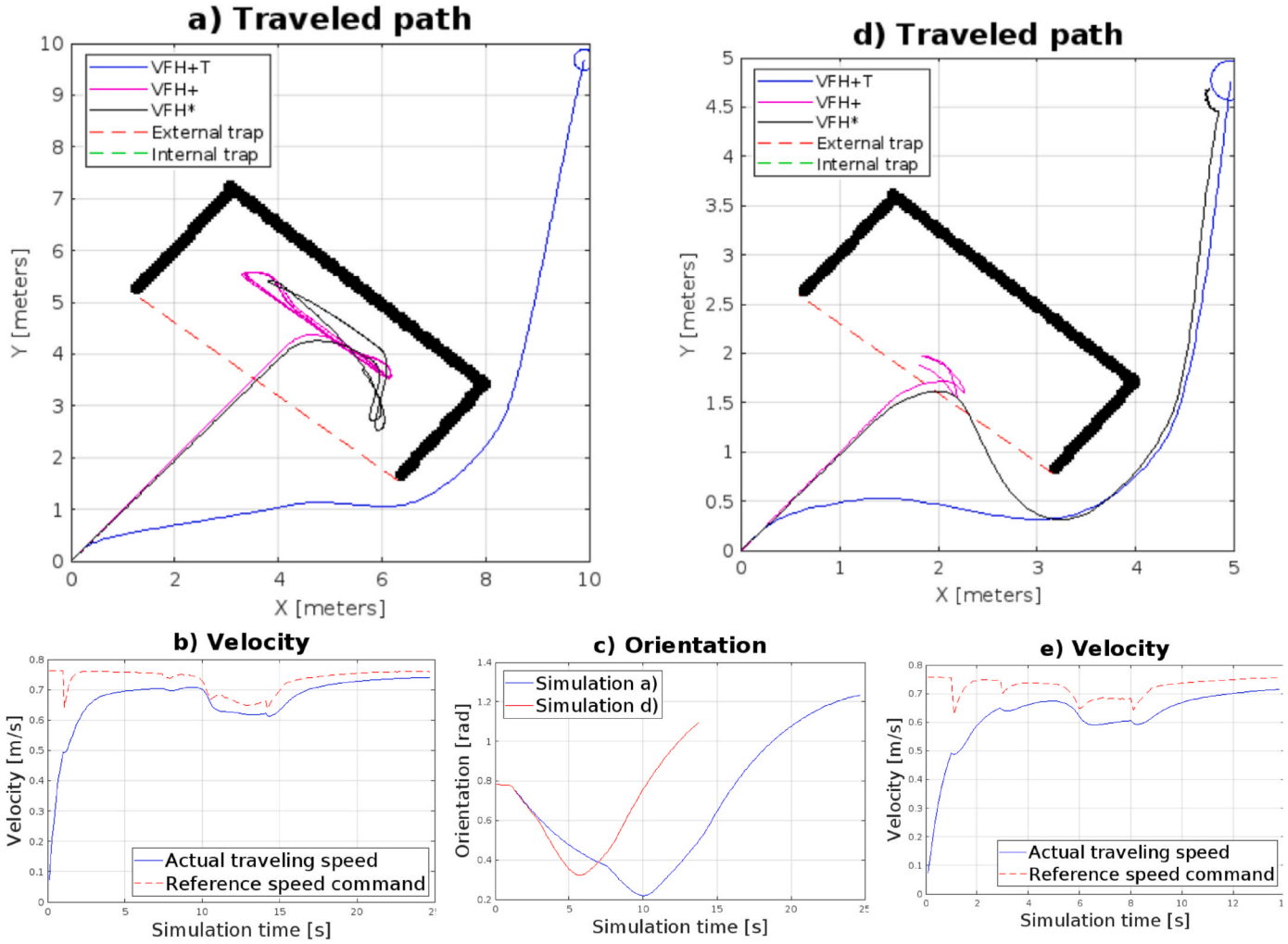


Fig. 7. Quantitative analysis of pathfinding algorithms in U-shaped obstacles of VFH + T, VFH+, and VFH* algorithms. a): Trajectory optimization in U-shaped obstacles: comparative path analysis of VFH + T, VFH+, and VFH* with emphasis on loop avoidance and path efficiency. b): Velocity profile metrics of the VFH + T trajectory depicted on Fig. 7/a. c): Orientation metrics across simulated trials, angular displacement analysis reflecting algorithmic response to navigational challenges. d): Enhanced obstacle proximity detection: VFH + T algorithm's early recognition and avoidance strategies reducing path length by 17 % relative to VFH*. e): Graphical depiction of actual vs. commanded speed during simulation of VFH + T trajectory illustrated on Fig. 7/d.

Where $\mu_1;\mu_2;\mu_3;\mu_4$ are the weighting coefficients of the cost function; ${\phi_{tar}}^M$ is the momentary target direction and δ_{des} is the previous selected traveling direction. The first 3 components of the cost function are identical to the standard VFH + algorithm and the 4. component is responsible for further discouraging the robot to choose a direction leading to a trap situation. The weighting coefficients can take any arbitrary value, however for a goal-oriented behaviour they must satisfy the condition $\mu_1 > \mu_2 + \mu_3 + \mu_4$. In our simulations the values $\mu_1 = 5$; $\mu_2 = \mu_3 = 2$ and $\mu_4 = 0.5$ were used, yielding satisfactory results.

The desired traveling speed (u_{des}) is calculated separately from the steering command based on the following. The obstacle density around the robot is determined by eq. (59).

$$\rho_{obs} = \sum_{i=1}^{n_{scan}} 0.2{*}e^{-0.4*R_{lid}(\psi_i)} \tag{59}$$

The traveling speed command is calculated from the obstacle density and the steering angle of the robot as eq. (60) describes.

$$u_{des} = cos(\Delta_A(\delta_{vel};\delta_{des}))*\left[\frac{v_{max}-v_{min}}{2} + \frac{v_{max}-v_{min}}{\pi}\tan^{-1}(0.06n_{scan}-\rho_{obs})\right] \tag{60}$$

Where $v_{min} = 0.1m/s$ and $v_{max} = 0.8m/s$ are the minimum and maximum allowed velocities of the robot. The calculated u_{des} values are saturated in the range of $[v_{min};v_{max}]$.

4.5. Algorithm structure

The pseudo-code of the proposed navigation algorithm – omitting the processing of sensor data and the low-level motor controller is shown in Algorithm 1.

Algorithm 1 VFH+T

Input: $R_{mes}(\psi_i)$; P_{tar}; P_r; ω_i; a_i

Output: δ_{des}; u_{des}

while $P_r \neq P_{tar}$ **do** /* while target is not reached
  for $j = 1$ **to** n_{scan} **do** /* obstacle enlargement
    calculate $\gamma_{enl}(\psi_j)$ /* eq. (33)
    for $i = j - \gamma_{enl}(\psi_j)$ **to** $j + \gamma_{enl}(\psi_j)$ **do**
      calculate $R_{enl}(\psi_i)$ /* eq. (32)
      if $R_{enl}(\psi_i) < R_{mes}(\psi_i)$ **then** $R(\psi_i) = R_{enl}(\psi_i)$
      else $R(\psi_i) = R_{mes}(\psi_i)$
  for $i = 1$ **to** n_{scan} **do** /* primary histogram and grouping measurement points in one loop
    if $R(\psi_i) \leq d_{max}$ **then** $H(\psi_i) = a_{hist} - b_{hist} * R(\psi_i)$ /* primary histogram
    else $H(\psi_i) = 0$
    if $|\overrightarrow{P_iP_{i+1}}| < 2r_r$ **then** $G_{gr}(n_{gr}) \leftarrow P_{i+1}$ /* join scanned point P_{i+1} to group number n_{gr}
    else $n_{gr}++$ /* create new group
  detect G_{tar}; G_{sur} /* trap detection starts here
  if *concaveness* **for** G_{tar} **is** TRUE **and** G_{tar} **is** *aligned with previous values* **then**
    $E_1(n_{ext})$; $E_2(n_{ext}) \leftarrow G_{tar}$ /* save G_{tar} as external trap
    $n_{ext}++$
  if G_{sur} **is** *aligned with previous values* **then**
    $L_1(n_{int})$; $L_2(n_{int})$; $L_3(n_{int}) \leftarrow G_{sur}$ /* save G_{sur} as internal trap
    $n_{int}++$
  for $l = 1$ **to** n_{ext} **do** /* calculate H_{trap} from external traps
    calculate $K(l)$; M_l
    if $|\overrightarrow{P_rM_l}| + |\overrightarrow{M_lP_{tar}}| = |\overrightarrow{P_rP_{tar}}|$ **then** /* the external trap is between robot and target
      for $i = 1$ **to** n_{scan} **do**
        if $\psi_i \in K(l)$ **then**
          $H_{trap}(\psi_i) = 1$
  for $k = 1$ **to** n_{int} **do** /* calculate H_{trap} from internal traps
    calculate $B(k)$; N_k
    if $T_{L_1L_2L_3} = T_{P_rL_1L_2} + T_{P_rL_1L_3} + T_{P_rL_2L_3}$ **then** /* robot is inside the internal trap
      for $i = 1$ **to** n_{scan} **do**
        if $\psi_i \notin B(k)$ **then**
          $H_{trap}(\psi_i) = 1$
    elseif $|\overrightarrow{P_rN_k}| + |\overrightarrow{N_kL_3(k)}| = |\overrightarrow{P_rL_3(k)}|$ **then** /* robot is outside the internal trap
      for $i = 1$ **to** n_{scan} **do**
        if $\psi_i \in B(k)$ **then**
          $H_{trap}(\psi_i) = 1$
  ${\varphi_{tar}}^M \leftarrow$ **select** $\min(\varphi_{tar} - \psi_i)$ **where** $H_{trap}(\psi_i) \neq 1$ /* calculate modified target direction
  for $j = 1$ **to** n_{scan} **do** /* dynamic constraint histogram and detecting valleys in one loop
    calculate $\overrightarrow{\Delta v}(\psi_j)$ /* candidate directions of the dynamic constraint histogram
    for $i = 1$ **to** 3 **do** /* calculating for all 3 wheels
      calculate ${F_i}^{lim}$; $c_i(\psi_j)$ /* eq. (40), (45), (47)
    calculate $\beta_r \rightarrow$ **update** $c_i(\psi_j)$; **calculate** $\Delta t(\psi_j)$ /* eq. (48), (49)
    $H_{dyn}(\psi_j) = d_{hist} * \Delta t(\psi_j)$ /* dynamic constraint histogram
    if $H(\psi_j) + H_{dyn}(\psi_j) < h_{th}$ **then** /* detecting valleys
      calculate $\delta_{j,i}$ /* potential traveling directions
      calculate $G(\delta_{j,i})$ /* cost function for each candidate direction – eq. (58)
  $\delta_{des} \leftarrow \min(G(\delta_{j,i}))$ /* desired traveling direction
  calculate u_{des} /* desired traveling speed – eq. (60)
end while

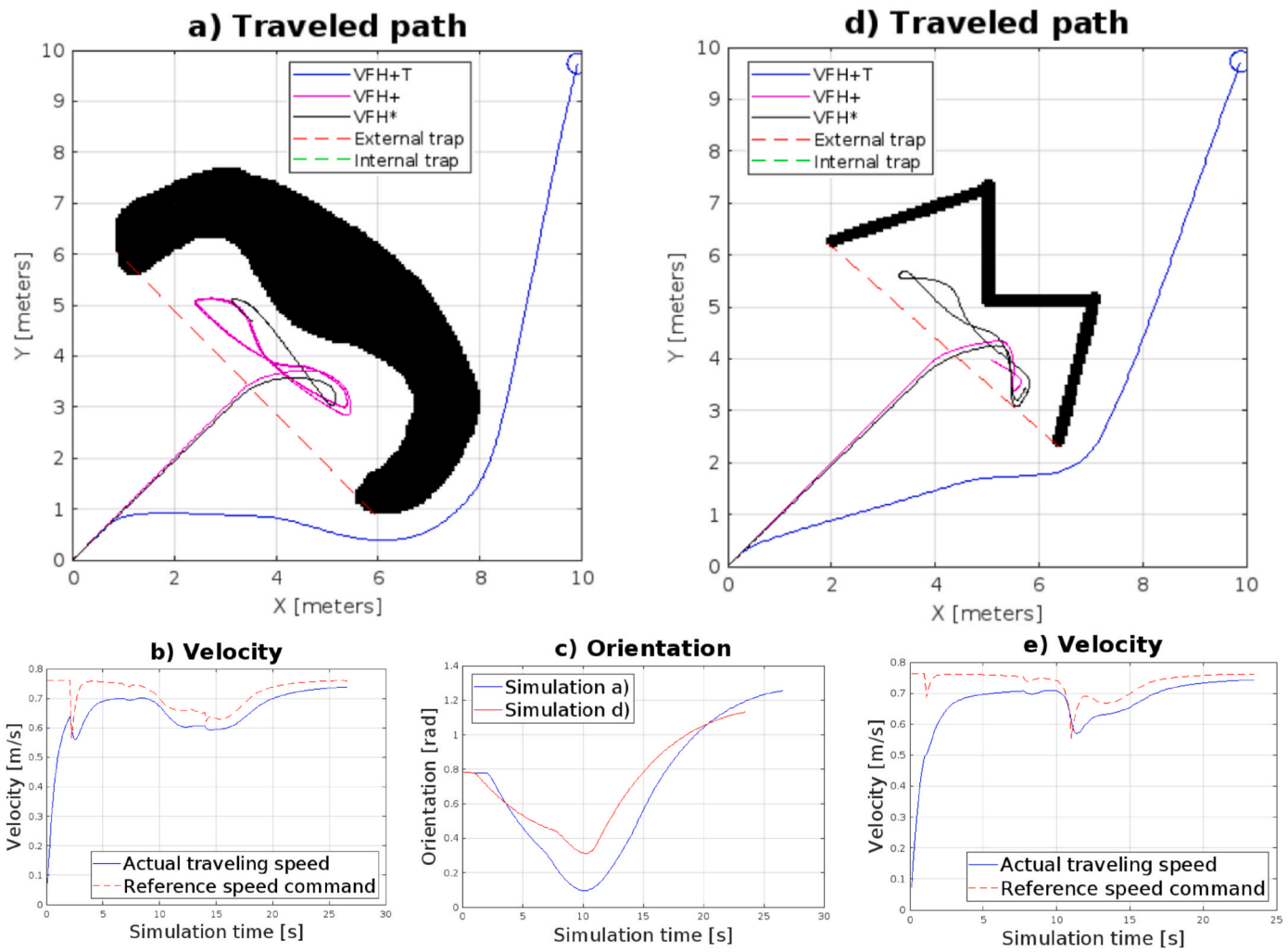


Fig. 8. Compares the navigation performance of VFH + T, VFH+, and VFH* algorithms in W-shaped obstacles. a): Travel path comparison in W-shaped obstacle scenario illustrates the navigational paths. b): Velocity profile for simulation a) shows actual versus commanded speeds. c): Details orientation adjustments. d): Travel path comparison in modified scenario depicts the navigational paths in a modified setup. e): Velocity profile for simulation d).

The blue notes in the pseudo-code are explaining comments to help link each section of the code to the corresponding parts of the methodology. During the navigation process the algorithm runs in high frequency loop to continuously adapt to the environmental changes. This allows reactive dynamic obstacle avoidance (similarly to most local navigation methods). However, since the focus of our method is trap avoidance, it lacks any kind of trajectory prediction feature (it is outside the scope of this paper), which would make the dynamic obstacle avoidance more reliable and robust. The pseudo-code doesn't contain the sensor data acquisition process, but the first step after that in every iteration is the augment of depth point array (obstacle enlargement) to compensate the dimensions of the mobile robot. After this, the primary polar histogram is constructed from the augmented distance measurements and the measurement points are linked into groups for the trap detection. For computational efficiency, these two steps are performed simultaneously, not in separate loops.

The trap detection process starts with identifying the target and surrounding group G_{tar} and G_{sur} (if they exist in the current iteration). We keep the positions of these two groups from the past couple iterations in the memory and compare them to the new values to filter out false detections (from measurement errors or moving obstacles). The target group also has to satisfy the concaveness criterion to be detected as an external trap. It should also be noted that traps with endpoints very close to previously detected ones are not saved due to unnecessary redundancy. Furthermore, the saved past trap situations should be dynamically deleted after certain amount of time has passed. Once the trap detection of the current iteration finished, we construct the binary trap histogram from the detected past traps. Once the binary trap histogram is calculated we determine the modified target direction $\phi_{tar}{}^{M}$.

Next the dynamic constraint histogram is built and simultaneously evaluated along with the primary polar histogram, yielding the potential traveling directions. After this, the desired traveling direction is selected with the help of the cost function and the desired traveling speed is calculated. These to values are passed down to the low-level motion controller of the vehicle, which is not part of the navigation algorithm.

5. Simulation results and discussion

The performance of the proposed VFH + T algorithm was rigorously tested across a variety of environmental conditions, with outcomes depicted in Figs. 7–10 and numerically summarized in Table 2. This enhanced algorithm was benchmarked against the traditional Vector

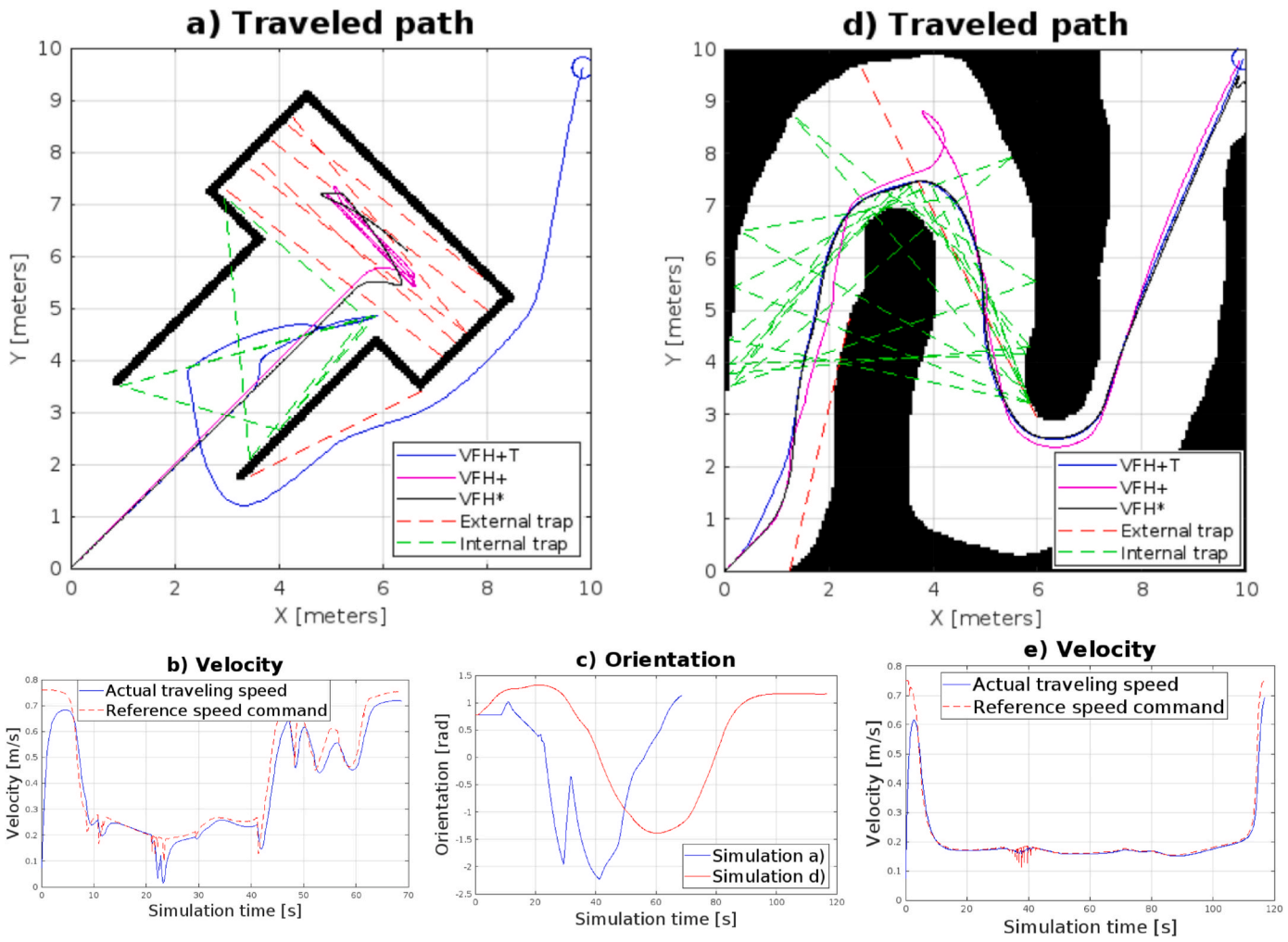


Fig. 9. Demonstrates algorithmic performance in navigating complex obstacles of VFH + T compared with VFH+, and VFH*. a): Depicts varied navigational paths through a t-shaped obstacle. b): Velocity profiles of VFH + T method in the T-shaped scenario. c): Charts orientation shifts changes. d): Highlights travel paths navigation efficiency in a curved corridor scenario. e): Velocity profile for curved corridor navigation with VFH + T.

Field Histogram Plus (VFH+) and VFH* algorithms to assess its navigational efficiency. Simulation results conducted in Ref. [41] highlight that while the standard VFH + algorithm struggles with local minima traps, as particularly noted in scenarios like that shown in Fig. 9(d), the VFH + T algorithm consistently navigates to its destination without encountering obstacles. This advanced performance is attributed to the novel integration of advanced sensor data processing and dynamic path adjustment capabilities within the VFH + T algorithm, which effectively overcomes the limitations previously observed in VFH + models. Each figure in the series includes an occupancy map that visually represents the agent's trajectory, marked distinctly with a blue line. These maps are complemented by detailed graphs plotting linear traveling speed and orientation over time, offering in-depth insights into the navigational dynamics of the agent. Orientations on these graphs are measured in the world-coordinate frame. In the simulation scenarios the VFH + T algorithm's performance in navigating complex environments is vividly demonstrated, particularly with the U-shaped obstacles depicted in Fig. 7(a). This algorithm efficiently identifies and manoeuvres around external traps, contrasting significantly with the VFH+ and VFH* algorithms. Where VFH + falls into repetitive looping behaviours, VFH* faces challenges due to its fixed look-ahead distance (3.6 m), which is inadequate for obstacles wider than this measurement, such as the 6.29-m wide entrance shown.

Further simulation results in Fig. 7(d), where obstacles closely match the look-ahead distance, demonstrate that VFH* can handle simpler traps, albeit recognizing them just before potential entrapment. This leads to inefficient navigation routes. In stark contrast, VFH + T proactively detects dead-ends 1.5 m before encountering the obstacle, enabling it to initiate early avoidance manoeuvres. This approach not only shortens the travel path by 17 % compared to VFH*, as shown in Table 2, but also significantly reduces rotational movements and steering intensity by 80 % and 72 %, respectively, ensuring smoother navigation. Increasing the look-ahead distance for VFH* could potentially replicate some benefits seen with VFH + T; however, it would also increase the computational demand, potentially limiting its use in scenarios requiring swift responses. On the other hand, VFH + T utilizes the maximum range of onboard sensors effectively without increasing computational overhead, provided the laser scans are of sufficient resolution. These findings underscore the VFH + T algorithm's superior capability to anticipate and adapt to dynamic environmental challenges, showcasing its enhanced operational efficiency and robustness. This predictive capability highlights its suitability for improving autonomous

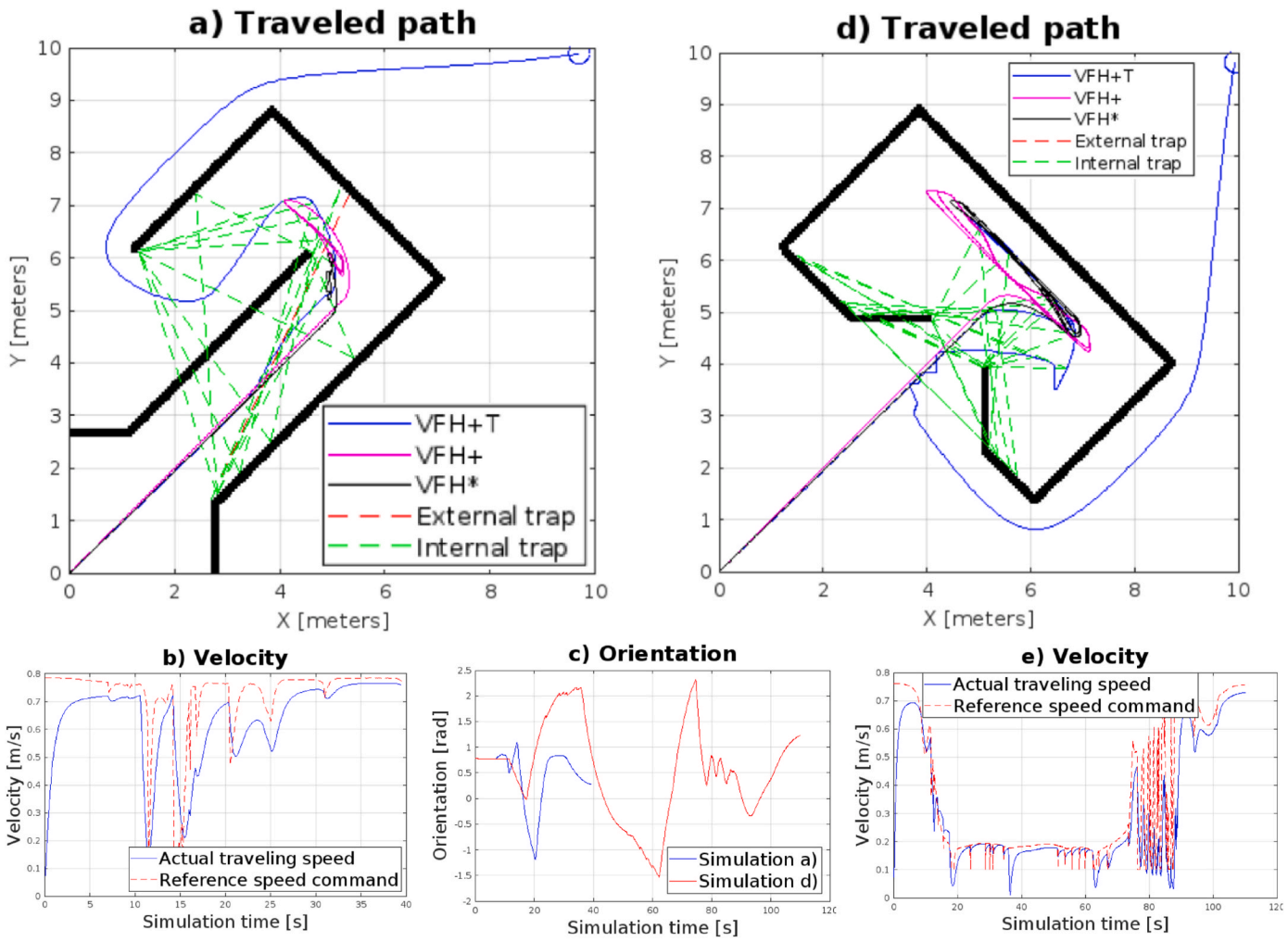


Fig. 10. Quantitative performance assessment of VFH + T compared with VFH+, and VFH* algorithms in handling complex geometric obstacles. a): Detailed trajectory analysis for L-shaped obstacle: internal trap detection and navigation strategy comparison. b): Velocity analysis of VFH + T in the L-shaped obstacle. c): Orientation dynamics across navigation algorithms: angular responses to complex obstacle encounters. d): In-depth simulation of navigation through pocket-shaped obstacles: challenges and adaptive responses by VFH + T. e): Velocity during VFH + T navigation in pocket-shaped obstacle.

navigation systems, making it a valuable tool for navigating through diverse and unpredictable terrains.

Building on previous simulation results, Fig. 8(a) and (d) highlight the performance of the VFH + T algorithm in environments with W-shaped obstacles. These obstacles are basic yet challenging due to line-of-sight limitations that can mislead navigation systems into perceiving a clear path through the centre. In these cases, the standard VFH + algorithm frequently falls into looping behaviours, where it repeatedly attempts to traverse the seemingly open path, only to be rerouted by the actual layout of the obstacle. This repetitive error indicates a limitation in the algorithm's ability to accurately interpret and adapt to the immediate environment.

In contrast, the VFH + T algorithm identifies these W-shaped configurations as external traps and successfully navigates around them. This capability to correctly interpret environmental cues allows the robot to avoid the inefficient looping behavior typical of VFH+. The VFH* algorithm, with its restricted look-ahead distance, does not recognize these traps in time, leading to navigation errors like those observed with VFH+. These findings demonstrate that the VFH + T algorithm can more effectively navigate complex environments than VFH+ and VFH*, enhancing the robot's operational efficiency without increasing computational demands. This efficiency boost underlines VFH + T's utility in effectively handling scenarios that challenge traditional navigation algorithms, highlighting its practical importance in enhancing robotic navigation systems. Further expanding on the VFH + T algorithm's robust performance in complex environments, Fig. 9/a introduces a "T"-shaped obstacle. In this scenario, what appears as a feasible path from the obstacle's front is a dead-end, only recognizable after the standard VFH + algorithm has entered and become trapped.

Unlike VFH+, the VFH + T algorithm proactively identifies this as an internal trap at coordinates (5.88, 4.85)m, enabling the robot to effectively navigate around it. Thanks to the robot's low speed of 0.15 m/s, it can pivot nearly in place, aided by the modified target direction detailed in Section 4.3, to successfully exit the trap. In contrast, Fig. 9/d features a curved corridor which poses limited navigational challenges, thus representing the only scenario where the standard VFH + algorithm performs without fault. However, should the corridor walls' curvature increase, it would likely introduce trapping scenarios. Both the VFH + T and VFH* algorithms demonstrate their adaptability by completing this obstacle course using shorter and more efficient paths, by 12 % and 9 %

Table 2
Numerical results of the simulations.

	Travel time [s]			Path length [m]			Total rotation [rad]			Average steering angle [rad]		
	VFH+T	VFH+	VFH*	VFH+T	VFH+	VFH*	VFH+T	VFH+	VFH*	VFH+T	VFH+	VFH*
U-shape	-	-	-	-	-	-	-	-	-	-	-	-
***a* version Fig. 7/a)**	24.8	107.2	52.8	16.49	27.12	18.13	1.58	17.82	2.28	0.079	0.077	0.216
***b* version Fig. 7/d)**	13.9	20.2	46.5	8.55	4.69	10.27	1.24	3.27	6.20	0.122	0.300	0.441
W-shape	-	-	-	-	-	-	-	-	-	-	-	-
***a* version Fig. 8/a)**	26.7	104.9	28.1	17.32	26.59	10.44	1.85	16.93	1.86	0.087	0.041	0.253
***b* version Fig. 8/d)**	23.6	27.6	42.6	15.68	9.06	16.22	1.29	3.21	3.20	0.068	0.161	0.380
T-shape Fig. 9/a)	68.9	82.9	52.4	26.58	21.36	13.12	10.17	12.25	2.20	0.142	0.120	0.124
Curved corridor Fig. 9/d)	117	93.3	122.6	23.38	26.48	24.12	5.83	7.43	2.55	0.064	0.097	0.118
L-shape Fig. 10/a)	39.6	63.5	54.1	25.1	15.65	9.93	5.94	8.73	2.03	0.235	0.093	0.182
Pocket-shape Fig. 10/d)	110.2	111.1	111.9	34.40	28.34	24.74	18.07	15.74	3.67	0.274	0.115	0.159

respectively. Importantly, the VFH + T algorithm requires significantly less aggressive steering adjustments, 34 % less than VFH+, showcasing its efficiency and reduced operational strain. These results highlight the VFH + T algorithm's capability to interpret and adapt to complex environmental cues more effectively than traditional algorithms, thus improving navigation precision in challenging scenarios previously problematic for standard navigation systems. This performance underlines the practical utility of VFH + T in enhancing robotic navigation where traditional algorithms struggle, as demonstrated across different test environments.

Similarly, Fig. 10/a illustrates the challenges of navigating an "L"-shaped obstacle, which demands the recognition of an internal trap to successfully navigate past the critical turning point. This task is notably more complex than navigating U-shaped obstacles due to a separating wall that limits line-of-sight, preventing a clear view of the trap from a single perspective.

In these scenarios, both the VFH+ and VFH* algorithms struggle to escape the turning point. Their goal-oriented behaviour inadvertently pulls them back towards the global target, which coincidentally aligns with the direction of the dead-end. In contrast, the proposed VFH + T method successfully exits the trap by employing a modified target direction that points directly opposite to the global target within the turning area. Additionally, Fig. 10/d presents a pocket-shaped obstacle, which remains undetectable until the robot has already entered the trap. The narrow entry points further complicate navigation, presenting a significant challenge for local navigation methods. Despite these difficulties, the VFH + T algorithm detects the internal trap after approximately 50 s of manoeuvring within the obstacle. Table 2 provides key numerical results from the simulation scenarios involving various obstacle types. It includes data on the average steering angle, which measures the deviation between the current traveling direction and the desired traveling angle calculated by the navigation algorithm at each timestep. This metric effectively illustrates the aggressiveness of the steering required by each navigation algorithm, highlighting differences in their adaptability and efficiency. In the "Travel time" column the background colour green indicates successful while orange suggests failed navigation for each method in the actual obstacle course.

As the results suggest the VFH + T successfully navigated in each testing environment, while the VFH+ and VFH* had difficulties with trap situations most of the time and failed to reach the target. This analysis underscores the advanced capability of the VFH + T algorithm to interpret complex environments and adjust navigational strategies accordingly, enhancing operational efficiency and accuracy in challenging scenarios where traditional algorithms falter.

6. Conclusion

This paper introduced substantial enhancements to the Vector Field Histogram Plus (VFH+) path planning algorithm by integrating advanced trap-escaping functionalities, significantly advancing its capability to navigate complex environments. These improvements stemmed from incorporating a historical data-driven approach, where previously encountered traps were recorded and utilized to dynamically adjust the robot's navigation strategy. This method leveraged a binary trap histogram to alter the robot's target direction, effectively circumventing potential dead-ends and optimizing route selection, even in temporarily risky paths. A key innovation of this study was the dual-layered navigational framework. The first layer adjusted the target direction based on historical trap data, while the second layer used a polar histogram enhanced with a refined cost function to select the optimal steering direction, integrating real-time kinematic feasibility. This comprehensive approach not only improved the robot's decision-making capabilities but also enabled it to adeptly handle diverse obstacle-rich environments. The algorithm's efficacy was demonstrated through rigorous simulation results, which illustrated its ability to manage typical navigation challenges like dead-ends and dynamically adapt to changes in the environment. These simulations underscored the VFH + T algorithm's superior performance in avoiding navigational traps compared to traditional methods, thereby enhancing operational safety and efficiency. Traveling distance is reduced by 9–17 % compared to VFH+ and VFH* algorithms in the obstacle courses used for validation, along with a significant reduction in oscillating movement.

The future work of the research aims to conduct real-world testing and further refine the algorithm to accommodate the precise geometric dimensions of different vehicles. This includes moving beyond the simplistic circular safety region model to a more accurate representation of the vehicle's shape, which is expected to significantly improve the precision and applicability of the algorithm in highly cluttered

environments. Overall, this research marked a pivotal step forward in autonomous navigation technologies, showcasing the potential of memory-based path planning enhancements to significantly boost the capabilities of robotic navigation systems. The advancements detailed in this paper were poised to set new standards for reliability and versatility in robotic mobility, paving the way for more sophisticated autonomous navigation applications in various sectors, from automotive to industrial automation.

Funding

Not applicable.

CRediT authorship contribution statement

Husam A. Neamah: Writing – review & editing, Writing – original draft, Supervision, Resources, Methodology, Investigation, Formal analysis, Conceptualization. **Elek Donát:** Writing – review & editing, Writing – original draft, Visualization, Methodology, Data curation, Conceptualization. **Péter Korondi:** Supervision, Investigation, Funding acquisition.

Declaration of competing interest

The authors have no relevant financial or non-financial interests to disclose.

We confirm that this work is original and has not been published elsewhere, nor is it currently under consideration for publication elsewhere.

Data availability

No data was used for the research described in the article.

Acknowledgements

The authors wish to thank the support of the Hungarian Research Fund (OTKA K143595).

References

[1] S. Vanneste, B. Bellekens, and M. Weyn, "3DVFH+: Real-Time Three-Dimensional Obstacle Avoidance Using an Octomap".

[2] C. Ma, H. Zou, X. An, A complete coverage path planning approach for an autonomous underwater helicopter in unknown environment based on VFH+ algorithm, J. Mar. Sci. Eng. 12 (3) (Mar. 2024) 3, https://doi.org/10.3390/jmse12030412.

[3] H. Wang, L. Wang, J. Li, L. Pan, A vector polar histogram method based obstacle avoidance planning for AUV, in: 2013 MTS/IEEE OCEANS, Bergen, Jun. 2013, pp. 1–5, https://doi.org/10.1109/OCEANS-Bergen.2013.6608088.

[4] X. Wang, M. Cheng, S. Zhang, H. Gong, Multi-UAV cooperative obstacle avoidance of 3D vector field histogram Plus and dynamic window approach, Drones 7 (8) (Aug. 2023) 8, https://doi.org/10.3390/drones7080504.

[5] H.A. Neamah, D. Al Ardi, P. Korondi, Enhancing autonomous robot perception for precision positioning and localization, in: 2024 IEEE/SICE International Symposium on System Integration (SII), IEEE, 2024, pp. 1058–1063 [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10417351/. (Accessed 19 February 2024).

[6] R. Rey, M. Corzetto, J. Cobano, L. Merino, F. Caballero, Human-robot Co-working System for Warehouse Automation, Sep. 2019, pp. 578–585, https://doi.org/10.1109/ETFA.2019.8869178.

[7] P. Pappas, M. Chiou, G.-T. Epsimos, G. Nikolaou, R. Stolkin, VFH+ based shared control for remotely operated mobile robots, in: 2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), 2020, pp. 366–373, https://doi.org/10.1109/SSRR50563.2020.9292585.

[8] K. Katona, H.A. Neamah, P. Korondi, Obstacle avoidance and path planning methods for autonomous navigation of mobile robot, Sensors 24 (11) (Jan. 2024) 11, https://doi.org/10.3390/s24113573.

[9] D. Foead, A. Ghifari, M.B. Kusuma, N. Hanafiah, E. Gunawan, A systematic literature review of A* pathfinding, Procedia Comput. Sci. 179 (Jan. 2021) 507–514, https://doi.org/10.1016/j.procs.2021.01.034.

[10] S.M. LaValle, Rapidly-exploring random trees: A new tool for path planning. Report No. TR 98-11, Computer Science Department, Iowa State University, 1998 vailable online: http://lavalle.pl/papers/Lav98c.pdf. (Accessed 1 March 2024).

[11] L.E. Kavraki, P. Svestka, J.-C. Latombe, M.H. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces, IEEE Trans. Robot. Autom. 12 (4) (1996) 566–580, https://doi.org/10.1109/70.508439.

[12] S. Jameel Al-Kamil, R. Szabolcsi, Optimizing path planning in mobile robot systems using motion capture technology, Results Eng 22 (Jun. 2024) 102043, https://doi.org/10.1016/j.rineng.2024.102043.

[13] N. Promkaew, S. Thammawiset, P. Srisan, P. Sanitchon, T. Tummawai, S. Sukpancharoen, Development of metaheuristic algorithms for efficient path planning of autonomous mobile robots in indoor environments, Results Eng 22 (Jun. 2024) 102280, https://doi.org/10.1016/j.rineng.2024.102280.

[14] D. Fox, W. Burgard, S. Thrun, The dynamic window approach to collision avoidance, IEEE Robot. Autom. Mag. 4 (1) (Mar. 1997) 23–33, https://doi.org/10.1109/100.580977.

[15] J. Minguez, L. Montano, Nearness diagram (ND) navigation: collision avoidance in troublesome scenarios, IEEE Trans. Robot. Autom. 20 (1) (2004) 45–59, https://doi.org/10.1109/TRA.2003.820849.

[16] J. Wu, X. Ma, T. Peng, H. Wang, An improved timed elastic Band (TEB) algorithm of autonomous ground vehicle (AGV) in complex environment, Sensors 21 (24) (Jan. 2021) 24, https://doi.org/10.3390/s21248312.

[17] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, in: 1985 IEEE International Conference on Robotics and Automation Proceedings, Mar. 1985, pp. 500–505, https://doi.org/10.1109/ROBOT.1985.1087247.

[18] Y. Koren, J. Borenstein, Potential field methods and their inherent limitations for mobile robot navigation, 1991 IEEE International Conference on Robotics and Automation Proceedings 2 (Apr. 1991) 1398–1404, https://doi.org/10.1109/ROBOT.1991.131810.

[19] J. Borenstein, Y. Koren, The vector field histogram-fast obstacle avoidance for mobile robots, IEEE Trans. Robot. Autom. 7 (3) (Jun. 1991) 278–288, https://doi.org/10.1109/70.88137.

[20] I. Ulrich, J. Borenstein, VFH+: reliable obstacle avoidance for fast mobile robots, in: Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146), vol. 2, 1998, pp. 1572–1577, https://doi.org/10.1109/ROBOT.1998.677362.

[21] I. Ulrich, J. Borenstein, VFH/sup */: local obstacle avoidance with look-ahead verification, in: Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), vol. 3, Apr. 2000, pp. 2505–2511, https://doi.org/10.1109/ROBOT.2000.846405.

[22] A. Babinec, F. Duchoň, M. Dekan, P. Pásztó, M. Kelemen, VFH*TDT (VFH* with Time Dependent Tree): a new laser rangefinder based obstacle avoidance method designed for environment with non-static obstacles, Robot. Autonom. Syst. 62 (8) (Aug. 2014) 1098–1115, https://doi.org/10.1016/j.robot.2014.05.003.

[23] A. Suty, F. Duchon, Investigation of the influence of the parameters of the VFH + method on the navigation efficiency of the mobile robot, WSEAS Trans. Syst. Control 16 (2021) 328–334, https://doi.org/10.37394/23203.2021.16.28.

[24] D. Díaz, L. Marín, VFH+D: an improvement on the VFH+ algorithm for dynamic obstacle avoidance and local planning, IFAC-PapersOnLine 53 (2) (Jan. 2020) 9590–9595, https://doi.org/10.1016/j.ifacol.2020.12.2450.

[25] H.A. Neamah, R. Butdee, Optimization modeling parameters for industrial AMR slippage using ANFIS system in dynamic environment, in: P. Janmanee, S. Chujuarjeen, S. Butdee, P. Srikhumsuk, A.D.L. Batako, A. Burduk, M.A. Xavior (Eds.), *Advanced In Creative Technology- Added Value Innovations In Engineering, Materials And Manufacturing*, Vol. 979, Lecture Notes in Networks and Systems, vol. 979, Springer Nature Switzerland, Cham, 2024, pp. 214–223, https://doi.org/10.1007/978-3-031-59164-8_18.

[26] F. Ye, J. Yang, C. Ma, H.-J. Rong, Combing VFH with bezier for motion planning of an autonomous vehicle, J. Phys. Conf. Ser. 887 (Aug. 2017) 012008, https://doi.org/10.1088/1742-6596/887/1/012008.

[27] B. Kazem, A.H. Hamad, M. Mozael, Modified vector field histogram with a neural network learning model for mobile robot path planning and obstacle avoidance, Int. J. Adv. Comput. Technol. 2 (5) (Dec. 2010) 166–173, https://doi.org/10.4156/ijact.vol2.issue5.18.

[28] K. Mohamed, A. Aliedani, A. Al-Ibadi, Adaptive vector field histogram Plus (VFH+) algorithm using fuzzy logic in motion planning for quadcopter, J. Robot. Control JRC 5 (Mar. 2024) 582–596, https://doi.org/10.18196/jrc.v5i2.21540.

[29] A.H. Hamad, F.B. Ibrahim, Path planning of mobile robot based on modification of vector field histogram using neuro-fuzzy algorithm, Int. J. Adv. Comput. Technol. 2 (3) (Aug. 2010) 129–138, https://doi.org/10.4156/ijact.vol2.issue3.14.

[30] A. Babinec, M. Dekan, F. Duchoň, A. Vitko, Modifications of VFH navigation methods for mobile robots, Procedia Eng. 48 (Jan. 2012) 10–14, https://doi.org/10.1016/j.proeng.2012.09.478.

[31] Y. Zhang, G. Wang, An improved RGB-D VFH+ obstacle avoidance algorithm with sensor blindness assumptions, in: 2017 2nd International Conference on Robotics and Automation Engineering (ICRAE), 2017, pp. 408–414, https://doi.org/10.1109/ICRAE.2017.8291420.

[32] "VPH: a new laser radar based obstacle avoidance method for intelligent mobile robots | IEEE Conference Publication | IEEE Xplore." Accessed: March. 26, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/1342407.

[33] J. Gong, Y. Duan, Y. Man, G. Xiong, VPH+: an enhanced vector polar histogram method for mobile robot obstacle avoidance, in: 2007 International Conference on Mechatronics and Automation, 2007, pp. 2784–2788, https://doi.org/10.1109/ICMA.2007.4304000.

[34] K. Liu, J. Gong, H. Chen, VPH+ and MPC Combined Collision Avoidance for Unmanned Ground Vehicle in Unknown Environment, 2018.
[35] S.-H. Lee, G. Eoh, B.H. Lee, Robust robot navigation using polar coordinates in dynamic environments, J. Ind. Intell. Inf. 2 (1) (2014) 6–10, https://doi.org/10.12720/jiii.2.1.6-10.
[36] G. Jianming, Z. Shouping, X. Jia, Z. Shenghui, Kalman Prediction Based VFH of Dynamic Obstacle Avoidance for Intelligent Vehicles, Oct. 2010, https://doi.org/10.1109/ICCASM.2010.5620252.
[37] X. Teng, Z. Shen, L. Huang, H. Li, W. Li, Multi-sensor fusion based wheeled robot research on indoor positioning method, Results Eng 22 (Jun. 2024) 102268, https://doi.org/10.1016/j.rineng.2024.102268.
[38] D.U. Rijalusalam, I. Iswanto, Implementation kinematics modeling and odometry of four omni wheel mobile robot on the trajectory planning and motion control based microcontroller, J. Robot. Control JRC 2 (5) (2021), https://doi.org/10.18196/jrc.25121.
[39] "Control Principles of Autonomous Mobile Robots Used in Cyber-Physical Factories | IEEE Conference Publication | IEEE Xplore." Accessed: March. 26, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/9447468.
[40] L. Raj, A. Czmerk, Modelling and simulation of the drivetrain of an omnidirectional mobile robot, Automatika 58 (2) (Apr. 2017) 2, https://doi.org/10.1080/00051144.2017.1391612.
[41] MATLAB." Accessed: April. 28, 2024. [Online]. Available: https://matlab.mathworks.com/.