A Fuzzy Set-based Methodology for Autonomous Navigation

Ehsan Adel Rastkhiz*, Howard Schwartz and Ioannis Lambadaris

Department of System and Computer Engineering, Carleton University, 1125 Colonel By Drive, Ottawa, K1S 5B6, Ontario, Canada

ARTICLE INFO

Keywords:
Autonomous navigation
Path-planning
Collision avoidance
Fuzzy obstacle

ABSTRACT

This paper presents a fuzzy set-based methodology to achieve simultaneous path adherence and local collision avoidance in autonomous navigation. It targets scenarios where a global path to the vehicle's destination is already planned but with imperfect knowledge of the obstacles in the environment and their dynamics. Assuming that the vehicle can localize itself and the obstacles around it, the proposed methodology incorporates the global path and the acquired real-time knowledge of the local obstacles by the vehicle to create a comprehensive fuzzy representation of the environment. This fuzzy representation is then utilized to assess the desirability of the vehicle states within an optimization framework that balances global path adherence and obstacle avoidance objectives. The proposed gradient-based solution to this optimization problem navigates the vehicle such that it maintains its global course toward the designated destination while avoiding collision with local obstacles. Extensive simulations on a mobile robot validate the method's efficacy in guiding vehicles along desired paths, maneuvering around obstacles with minimal deviation from the route, and negotiating local minima without oscillations. Additionally, simulation results highlight the proposed fuzzy set-based methodology's low computational complexity, real-time operation capability, and adaptability in handling complex geometries of paths and obstacles.

1. Introduction

Autonomous vehicles have always been in the research spotlight for their pivotal role in the future of transportation and, subsequently, all other dependent domains. These transportation-dependent applications span a broad spectrum, encompassing domains such as logistics and package delivery [1, 2], industrial automation [3, 4, 5, 6], as well as mobility assistance and medical transportation in healthcare [7, 8, 9]. Automating such intricate tasks through autonomous vehicles promises enhanced performance and cost reduction, thus substantiating the rationale for continued research endeavors in this realm. Despite the apparent diversity in autonomous vehicle applications owing to differences in environments, vehicle types, and operational tasks, they share fundamental requisites. Path planning, among others, is a pivotal requirement and a foundational element in achieving autonomy.

Path planning in autonomous navigation typically comprises two stages. The initial phase involves establishing a mathematical representation of the surrounding environment, and the second is utilizing this model to determine a feasible route to the intended destination. A prevalent method for environment representation involves leveraging graphs. These graphs can be derived from basic occupancy grids or through more sophisticated techniques such as cell decomposition [10] or probabilistic roadmaps [11]. More advanced dual graph representations further offer a means to integrate path quality metrics such as turn costs within the environment graph, enhancing the smoothness of resultant

*Corresponding author
ehsanadelrastkhiz@cmail.carleton.ca (E.A. Rastkhiz)
ORCID(s): 0000-0003-4279-3208 (E.A. Rastkhiz)

paths [6]. Employing these graph representations facilitates the subsequent stage—navigating to the target location. This navigation process can be performed using heuristic strategies such as the recursive rewarding strategy outlined in [12] or through dedicated graph search techniques such as Dijkstra's algorithm [13, 14, 6] or the A* algorithm [15].

The A* algorithm performs a heuristically guided search on a graph representation of the environment to find the minimum-cost path to the target [16]. This algorithm has gone through extensive adaptations since its emergence. For example, variants of the A* algorithm detailed in [17] and [18] [19] focus on enhancing the original version for smoother trajectories, real-time efficiency, shorter paths, and reduced turning maneuvers. Versions of the A* algorithm have also been proposed to tailor it for specific applications by integrating contextual requirements into the heuristic search process. For instance, in [20, 21], aquatic conditions such as water current and traffic separation have been integrated into the A* search algorithm for marine applications. The D* algorithm can also be regarded as an improved version of the A* to make it able to replan the path quickly in dynamic environments [22, 23, 24].

Another graph-based path-planning algorithm is the Rapidly Exploring Random Tree (RRT). This technique constructs and randomly expands a tree that explores the environment until the goal is reached [25]. RRT-connect is also an improved variant of RRT that grows two separate trees emerging from the start and goal nodes and tries to connect them to complete the path [26]. Despite its potential for finding shorter paths, RRT-connect introduces added computational complexity. The main drawbacks of RRT and RRT-connect in practice are their computational complexity, sub-optimality, and low path quality. The RRT* variant [27] and extended state space trees [28, 29] have been proposed in response to these disadvantages of RRT.

RRT* considers path quality costs and tries to minimize them while exploring the environment for a path to the target. Unlike RRT and RRT-connect, which generate sub-optimal routes, RRT* asymptotically converges to the optimal path. Quickly converging implementations of RRT* have already been proposed in the literature [30, 31] together with variants that can generate continuous paths [32]. Extended state space trees also aim to improve the feasibility and quality of the path simultaneously. They do so by considering the vehicle and environment dynamics when planning the route. Despite all these improvements, graph-based planning methods still confront challenges with computational load and inability to adapt to rapidly changing environments, precluding real-time implementation on resource-constrained autonomous vehicles. Consequently, graph-based algorithms are predominantly employed for global planning that is less time-sensitive.

Artificial Potential Fields (APFs) offer an alternate approach to environment representation, employing attractive potentials for desired target locations and repulsive ones to model areas to avoid. In APF-based techniques, the vehicle navigates by forming a force field from the potential gradient, minimizing repulsive effects while maximizing attractive ones [33]. Despite their low computational complexity and capacity to incorporate vehicle dynamics, APFs often get trapped in local minima, hindering progress toward the target. Various methods have been proposed to overcome

APF's local minima problem including leveraging harmonic potential functions [34], integrating virtual sub-target points [35, 36, 37], introducing tangential velocity factors [38], and employing alternative optimization algorithms to the greedy gradient descent. Examples of such algorithms are [39] that uses particle swarm optimization, [40] which proposes a hybrid ant colony-APF approach, [41] that introduces a deterministic annealing strategy, and [42] that applies harmony search with APFs. Another practical limitation of APFs is that they assume complete knowledge of the environment, which is not possible except in the local vicinity of the vehicle itself. As a result of this limitation and considering the low computational complexity of APFs, they are primarily used as local planners.

The main takeaway from the review of the existing path-planning algorithms (PPAs) so far is that they excel within their scopes but lack universality for practical deployment in autonomous vehicles. Graph-based methods are suitable for global planning in larger, stationary environments, while APFs perform better in local planning. A holistic solution, however, must address both local and global scopes, prompting research to merge these paradigms. Examples of the works in this thread include [43], which uses a global PPA to plan a path to the target while considering static obstacles only. It then uses the global path in the local APF-based planner as attractive sub-targets to guide the vehicle toward its destination while avoiding stochastically moving obstacles. Similarly, in [37] and [44], global path guidance is incorporated into the conventional APFs to address the target unreachability problem. APFs have also been fused with graph-based techniques as in [45]. The method therein combines a modified A* algorithm with APFs to reduce the computational burden of the A* search and achieve smoother paths. At the same time, this fusion can reduce the occurrence of local minimum configurations inherent in APFs. Such universal path-planning methods discussed so far still contend with limitations linked to environment shape (obstacles and the route), dependency on complete knowledge of the environment, and structural complexity.

This study presents a new methodology addressing the shortcomings of the available universal path planning techniques. It presupposes a global path and assumes that the vehicle can localize itself, detect nearby obstacles, and accurately estimate their locations and velocities. It then employs fuzzy sets to create a unified environment model encompassing the intended global route and local obstacles. The developed algorithm optimizes an objective function designed delicately to balance the path adherence and obstacle avoidance. The proposed gradient-based solution to this optimization problem guides the vehicle along the fuzzy environment's contour lines and as close as possible to the path core while deviating temporarily to avoid collisions. This approach offers advantages over existing methods by:

- Operating as a purely algebraic algorithm agnostic to obstacle and path shapes.
- Effectively managing local minima traps associated with concave-shaped objects.
- Reducing computational demands, as it relies on global path information for only a brief horizon.
- Extending versatility to various vehicles navigating different environments—be it airborne, terrestrial, or aquatic.

- Accounting for uncertainties and imprecision inherent in environment perception.

The paper's structure is as follows. Section 2 briefly reviews essential concepts from fuzzy set theory and details the construction of the fuzzy set representation of the environment. Section 3 formulates the simultaneous path adherence and collision avoidance problem using the fuzzy representation of the environment and describes its solution. Section 4 presents the simulation studies, followed by Section 5, which concludes the paper, summarizing the proposed approach and highlighting achieved performance enhancements.

2. Fuzzy Representation of the Environment

Prior to defining fuzzy obstacles and paths, the following definitions from fuzzy set theory are made. Definitions 1 and 2, taken from [46], describe the core of a fuzzy set and the requirements for its convexity, respectively. Definition 3 also presents Yager's class of t-conorms [47].

Definition 1. *A continuous normal fuzzy set's core (center) is the central point in its α-cut set at $\alpha = 1$.*

Definition 2. *A normal fuzzy set is convex if its α-cut sets are convex for all $\alpha \in [0 \;\; 1]$.*

Definition 3. *Yager's class of t-conorms for every $a, b \in [0 \;\; 1]$ is defined as (1) in which $p \in [1 \;\; +\infty)$ and sat(.) is the saturation function.*

$$S(a,b) = a \dot{+} b = \min\left(1, (a^p + b^p)^{1/p}\right) = sat\left((a^p + b^p)^{1/p}\right) \tag{1}$$

2.1. Fuzzy Objects and Obstacles

An object is a subset of the space, $\mathbb{R}^2$, that a single entity physically occupies. These subsets are usually detected by processing the data from a camera or Lidar. Imperfections such as sensor limitations, calibration issues, and motion blur can introduce uncertainty to object perception. Obstacles, in turn, denote areas within the vehicle's operational space where its presence is undesirable, either due to safety concerns or hardware limitations such as maintaining camera visibility. These regions encompass object-occupied spaces or denote areas where the vehicle's presence is undesired, regardless of object presence. Therefore, it can be inferred that an obstacle exists that corresponds to each object in a vehicle's workspace, but the inverse is not necessarily true. The spreads of such object-induced obstacles depend on their corresponding objects' velocities, geometrical shapes, and required safe distances.

The noteworthy point is that obstacles inherit the uncertainty from their parent objects, and this uncertainty must be factored into vehicle path planning. One approach to dealing with obstacle uncertainty in path planning is using fuzzy set theory. The data-driven fuzzy planner in [48], the fuzzy logic controller with Bandler-Kohout triangle subproduct inference in [49], the fuzzy behavior-based navigation schemes in [50, 51], and the fuzzy cognitive map approach in [52] can be named as examples of methods that try to address the uncertainty problem in autonomous navigation.

However, neither of these approaches uses a fuzzy representation to reflect the uncertainty in the perception of the environment directly.

Given the description of objects and their corresponding obstacles in terms of sets, representing them with fuzzy sets is justified when the perception data is uncertain. Moreover, smooth fuzzy-set representations allow control over the graduality with which the vehicle perceives the obstacles. In essence, with a fuzzy set representing an obstacle, it does not abruptly appear to the vehicle. Instead, the degree of membership in the obstacle gradually increases as the vehicle approaches and decreases in the same manner upon leaving its vicinity. This gradual perception offers a significant advantage, enabling the planning system to react smoothly to obstacles. Hence, this paper adopts fuzzy sets to model an uncertain environment.

The challenge, however, is that objects and obstacles they create are geometrically complex in real-world scenarios. This complexity makes finding a membership function (MF) that can describe the obstacles difficult. A potential solution involves breaking down complex objects and obstacle shapes into several atomic ones characterized by fuzzy sets with simple membership functions. These atomic sets play an analogous role to pixels in a digital image, but they can have various sizes and are fuzzy rather than crisp. Since high fidelity is not of concern in path planning, all atomic sets of an obstacle can have uniform symmetric shapes. As a result, a single base set can represent all of the constituting atomic sets of an obstacle with appropriate spatial transformations applied to its MF. The membership function of this base set, called the base membership function (BMF) henceforth, is denoted by $\mu(\chi)$. The variable χ is a generic point in the frame of reference of the base set, which is co-aligned with the inertial frame $\{I\}$ of the vehicle's environment.

Let $\mathbf{X} \in \mathbb{R}^2$ denote the operational environment of an autonomous vehicle, and $\mathbf{x} = [x_1, \quad x_2]^T$ be a generic point in $\mathbf{X}$. Consider N atomic objects, $O_1, \dots, O_i, \dots, O_N$ present in the environment, each moving at a velocity of $v_i, \quad i = 1, \dots, N$. The fuzzy membership functions describing the corresponding atomic obstacles $\tilde{O}_i, \quad i = 1, \dots, N$ can then be derived from the BMF, $\mu(\chi)$, by applying the transformations $\Lambda_i^{-1}\left(\mathbf{x} - \mathbf{c}_i\right), \quad i = 1, \dots, N$ as described in (2).

$$\mu_{\tilde{O}_i}(\mathbf{x}) = \mu\left(\Lambda_i^{-1}\left(\mathbf{x} - \mathbf{c}_i\right)\right) \tag{2}$$

where $\mathbf{c}_i, \quad i = 1, \dots, N$ are the locations of the objects in the inertial frame and $\Lambda_i, \quad i = 1, \dots, N$ are scale matrices selected according to (3).

$$\Lambda_i = \left(\lambda_i + r_i\right) I_2 + \left[q_i, q_i^{\perp}\right] \begin{bmatrix} \delta||v_i||_2 & 0 \\ 0 & 0 \end{bmatrix} \left[q_i, q_i^{\perp}\right]^T \tag{3}$$

The parameters $r_i \geq 0, \quad i = 1, \dots, N$ in (3) are the mandated safe distances from each object O_i. The scaling parameters $\lambda_i > 0, \quad i = 1, \dots, N$ should be selected in such a way that the core of the obstacle set $\tilde{O}_i$ tightly contain the object O_i

when $r_i = 0$ and $||v_i||_2 = 0$. This ensures that the obstacle $\tilde{O}_i$ is identical to its parent object O_i when no safety distance is considered and the object is not moving. The second term on the right-hand side of (3) further inflates the obstacles $\tilde{O}_i$ in the direction of their velocities. The degree of inflation depends on the magnitude of the objects' velocities, with $\delta \geq 0$ being a constant of proportionality. Taking the normalized velocities $q_i = v_i/||v_i||_2$ and the unit vectors normal to them $q_i^{\perp} = R(\pi/2)q_i$ as the eigenvectors, the term responsible for inflation would have an eigendecomposition as the second term in the expression of Λ_i in (3). The matrix $R(\pi/2)$ also denotes the 2D rotation matrix by $\pi/2$ radians. The collective obstacle region within the vehicle's workspace, denoted as $\tilde{O}$, then can be expressed by combining all individual obstacles, $\tilde{O}_i$ through a fuzzy union as described in (4) and (5).

$$\tilde{O} = \bigcup_{i=1}^{N} \tilde{O}_i \tag{4}$$

$$\mu_{\tilde{O}}(\mathbf{x}) = \mu_{\tilde{O}_1}(\mathbf{x}) \dot{+} \ldots \dot{+} \mu_{\tilde{O}_i}(\mathbf{x}) \dot{+} \ldots \dot{+} \mu_{\tilde{O}_N}(\mathbf{x}) \tag{5}$$

For the planning technique proposed in this work, the BMFs and the applied t-conorms for implementing the fuzzy unions must fulfill some requirements. The BMFs, particularly, must be convex and piece-wise differentiable (at least once) in their universes of discourse. In this work, the bell-shaped membership function given in (6) is used as the BMF for the fuzzy representation of obstacles.

$$\mu(\chi) = \tanh\left((\chi^T \chi)^{-b}\right) \tag{6}$$

where the parameter $b \geq 1$ controls how smoothly the membership degree drops moving away from the object in an arbitrary direction. Figure 1 demonstrates the effect of the parameter b in (6) with color-coded membership degrees. Both subplots (a) and (b) show that the core of the fuzzy set described by (6) is a circle centered at the origin with a unit radius. Comparing the two subplots further shows that larger values of the parameter b result in a sharper reduction of the membership degree outside the core, making the fuzzy set closer to a crisp one and vice versa. Therefore, the BMF in (6) can represent both crisp and fuzzy sets with an appropriate selection of the parameter b.

The applied t-conorm in (5) should also preserve the piece-wise differentiability of the participating MFs; hence, it should be a smooth function and have low computational complexity. Yager's class of t-conorms, (1), is ideal for this purpose. The reason is that the t-conorms in this class are parametric, which allows their behavior to be adjusted anywhere between the drastic sum and maximum t-conorms. It is well-known in fuzzy logic literature that a drastic sum results in the greatest union set of two fuzzy subsets. In contrast, maximum generates the smallest union set of the same subsets [53]. In Yager's class of t-conorms given by (1), smaller values of the parameter p result in a behavior

closer to the drastic sum, whereas large values of p make it behave closer to the maximum. In the context of obstacle representation, using a Yager t-conorm with the smallest p corresponds to the highest level of conservativeness because it gives the largest possible obstacle area resulting from several objects in the environment.

Additionally, members of Yager's class of t-conorms are essentially saturated p-norms. Therefore, they can be expressed as differentiable functions of their operands using smooth implementations of the saturation function in (1). An example of a smooth saturation function that is used in this study is the hyperbolic tangent function $tanh(.)$. Finally, t-conorms in Yager's class are computationally efficient when closed-form MFs are required for the union of many fuzzy subsets. Remark 1, in the following, generalizes the binary definition of Yager's class of t-conorms for multiple operands and demonstrates its computational efficiency in symbolic calculations. Derivation of the multi-operand version of Yager's class of t-conorms is straightforward and can be found in the appendix.

Remark 1. *Let* $a_i, \quad i = 1, 2, \dots, m$ *be* $m \geq 2$ *operands such that* $a_i \in [0 \quad 1], \quad for \quad i = 1, .., m$. *The overall Yager's t-conorm of the m operands is given by* (7) *with* $\dot{+}$ *denoting the binary t-conorm in* (1).

$$a_1 \dot{+} \dots \dot{+} a_i \dot{+} \dots \dot{+} a_m = \min\left(1, \left(\sum_{i=1}^{m} a_i^p\right)^{1/p}\right) = sat\left(\left(\sum_{i=1}^{m} a_i^p\right)^{1/p}\right) \tag{7}$$

The following example demonstrates the performance of the fuzzy approach to representing objects and obstacles.

Example 1: Consider the object O depicted in Figure 2 (a) and its breakdown into $N = 9$ circular atomic sets with centers, $c_i, \quad i = 1, \dots, 9$, equal spreads of $\lambda_i = 0.75 \quad (m)$ for all i, and $b = 4$. The resulting fuzzy obstacle using (2)-(6) is then derived for two cases, one assuming that the obstacle is static, which is shown in Figure 2 (b), and the other with the assumption that the obstacle is moving at a velocity of $[2, 2]^T (m/s)$. The parameters for the dynamic obstacle representation in this example are $\delta = 0.25$ and $r = 0.1 \quad (m)$, which result in the obstacle region in Figure 2 (c).

A key question is how atomic obstacles of various objects are constructed. This topic typically falls within the domain of mapping research. While closely linked to planning, mapping remains a distinct field; hence, in this study, we

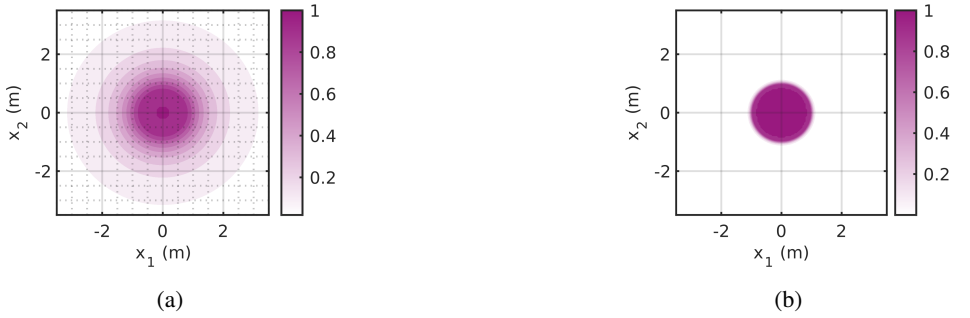

Figure 1: BMF for obstacle representation. (a) Contour plot with $b = 1$, (b) Contour plot with $b = 10$

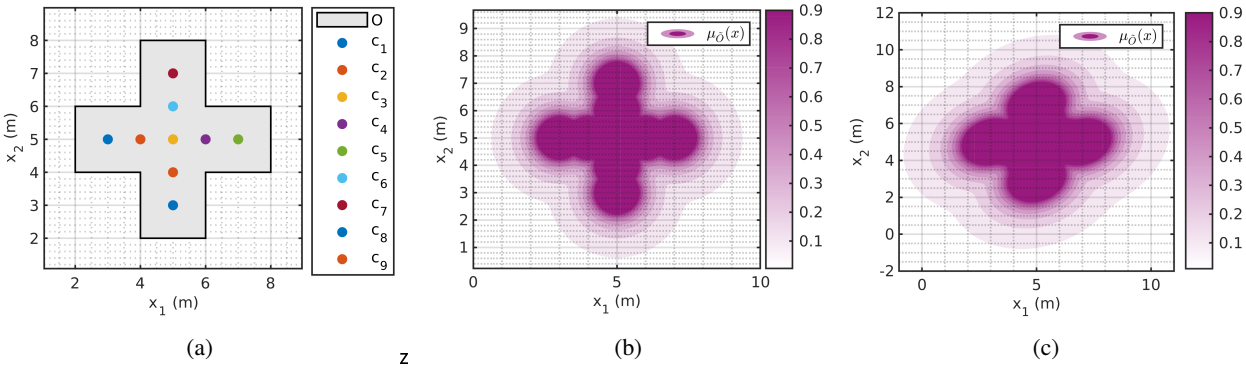

Figure 2: The Object O in Example 1 (a) The actual object and its atomic object centers, (b) The fuzzy representation of the obstacle assuming stationary condition. (c) The fuzzy representation of the obstacle in motion.

utilize an expert-provided representation of a simple environment to remain focused on our path-planning technique. However, a more practical and autonomous approach would involve applying fuzzy clustering techniques, such as [54, 55, 56], to the measurements gathered from the vehicle's range sensor (e.g., LiDAR). These clusters represent the atomic fuzzy obstacles corresponding to all objects in the scene. A subsequent segmentation system can then be developed and trained to associate these atomic obstacles with the specific objects they represent. Examples of applicable techniques for this task can be found in [57, 58, 59, 60].

2.2. Representation of the Path

Fuzzy sets can characterize a desired route for a vehicle with the advantage of providing the planning system with more flexibility when performing collision avoidance maneuvers than crisp definitions. Fuzzy paths can be constructed by defining virtual obstacles. However, unlike actual ones, the vehicle's presence in these virtual obstacle regions is desirable. Similar to most global planning techniques that characterize a route to a desired target with a series of waypoints $w_1, \dots w_M$, a chain of fuzzy waypoints $W_1, \dots, W_M$ can represent the fuzzy path to a target. Gaussian membership functions are ideal candidates for fuzzy waypoint definition in this setup. The reason is that they are convex, smooth, and have singleton cores. A singleton core, in particular, makes fuzzy waypoints generalized versions of crisp ones considering uncertainty. The Gaussian membership function (8) introduces the BMF used in this work for path representation. The variable χ in (8) is a generic point in the frame of reference of the Gaussian base set, which is co-aligned with the inertial frame $\{I\}$ of the vehicle's operating environment.

$$\xi(\mathbf{x}) = \exp(-\frac{1}{2}\chi^T\chi) \tag{8}$$

The transformation $\chi = \Gamma_j^{-1}(\mathbf{x} - \mathbf{p}_j)$ applied with (8) results in the MFs of individual fuzzy waypoints given in (9). The variables $\mathbf{p}_j = 0.5(w_j + w_{j+1}),\ \ j = 1, \dots, M$ in (8) are the midpoints of the line segments connecting w_j to

w_{j+1}. The parameters $\Gamma_j, \ \ j = 1, \dots, M$ are also 2×2 matrices specifying the spread of the fuzzy waypoints.

$$\mu_{W_j}(\mathbf{x}) = \xi(\Gamma_j^{-1}(\mathbf{x} - \mathbf{p}_j)) \tag{9}$$

The spreads should be adjusted so that the neighboring fuzzy waypoints have enough overlap along the path and sufficient lateral distance is accommodated in their support to allow for deviations from the path near obstacles. A spread matrix as (10) can achieve this objective.

$$\Gamma_j = [h_j, h_j^{\perp}]^T \begin{bmatrix} \gamma ||w_{j+1} - w_j||_2 & 0 \\ 0 & \sigma_j \end{bmatrix} [h_j, h_j^{\perp}] \tag{10}$$

where $h_j = (w_{j+1} - w_j)/||w_{j+1} - w_j||_2$ is the normalized vector pointing from w_j to w_{j+1} and $h_j^{\perp} = R(\pi/2)h_j$ is its normal. The parameter $\gamma \in [0 \ \ 1]$ adjusts the overlap between adjacent fuzzy waypoints. The parameters $\sigma_j, \ \ j = 1, \dots, M$ also specify the allowed lateral deviation from the path.

The union of individual fuzzy waypoints generates the overall fuzzy path representation as expressed by (11) and (12) in which $\dot{+}$ denotes the Yager's t-conorm implemented using Theorem 1.

$$P = \bigcup_{j=1}^{M} W_j \tag{11}$$

$$\mu_P(\mathbf{x}) = \mu_{W_1} \ \dot{+} \ \dots \ \dot{+} \ \mu_{W_j}(\mathbf{x}) \ \dot{+} \dots \ \dot{+} \ \mu_{W_M}(\mathbf{x}) \tag{12}$$

Example 2 demonstrates the performance of the fuzzy path representation approach.

Example 2: Let the crisp waypoints of the path be generated by sampling a sinusoidal wave with unit frequency. The equivalent fuzzy waypoints and the resulting fuzzy path using (8) to (12) with $\sigma_j = 0.1 \ \ (m), \ \ \text{for all} \ \ j$ and $\gamma = 0.75$ are demonstrated in Figure 3.

2.3. Representation of the Environment

Considering the path as a virtual obstacle, a single fuzzy set E can represent the entire environment via combining the actual obstacles, $\tilde{O}$, and the virtual one, P, in a union as described in (13) and (14). The fuzzy union has the same requirements as in the case of fuzzy obstacles; hence, Yager's t-conorm is used for its implementation.

$$E = \tilde{O} \cup P \tag{13}$$

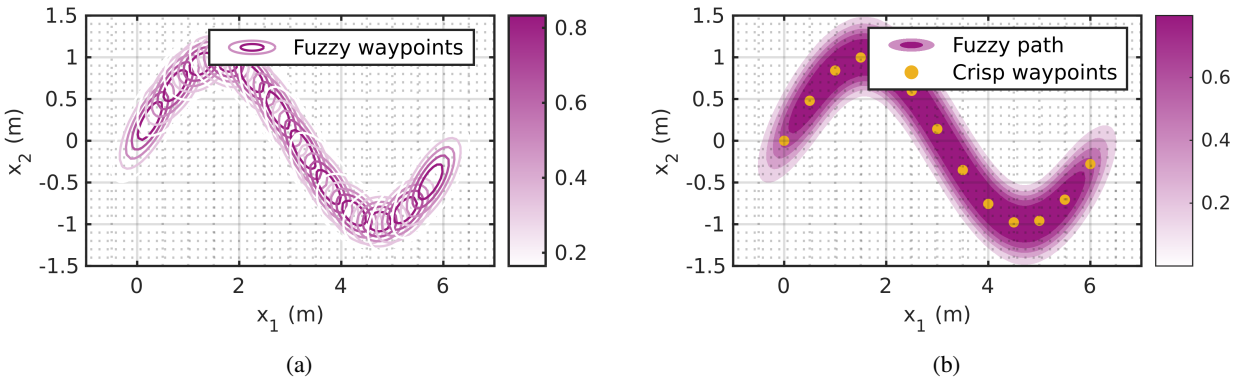

Figure 3: Fuzzy representation of the desired path. (a) Individual fuzzy waypoints, (b) Overall fuzzy path.

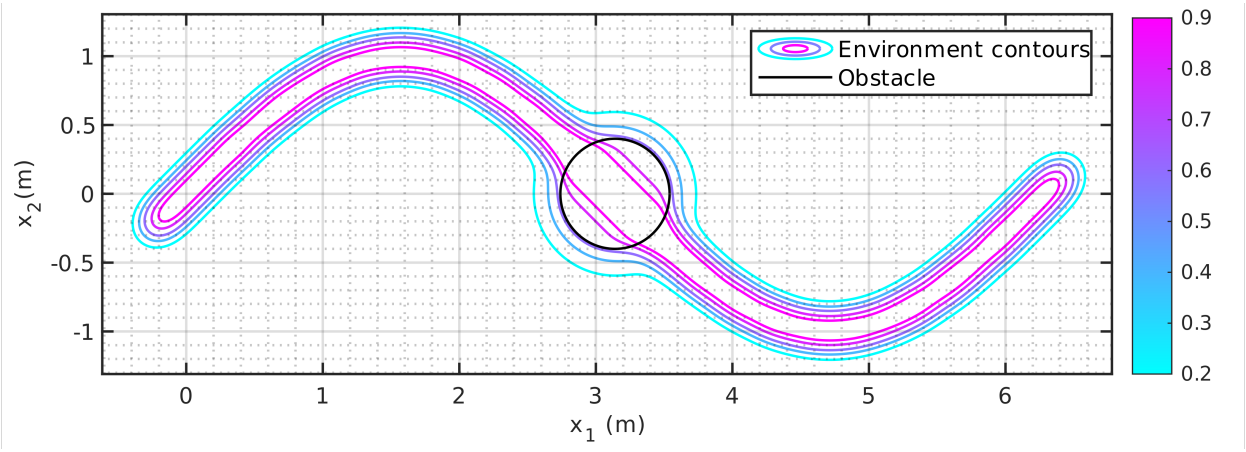

Figure 4: Example fuzzy representation of the environment.

$$\mu_E(\mathbf{x}) = \mu_{\tilde{O}}(\mathbf{x}) \dot{+} \mu_P(\mathbf{x}) \tag{14}$$

Example 3 demonstrates a sample fuzzy environment adopting the proposed representation approach.

Example 3: Consider the same sinusoidal path as in Example 2, together with a simple circular obstacle stationed at $[\pi, 0]^T$. Figure 4 depicts the resulting fuzzy environment using the developed representation scheme. It is evident from this figure that combining the actual obstacles with the path in a union results in an environment that is mostly the same as the fuzzy path. However, its contour lines are deformed in the vicinity of the obstacles.

The membership function in (14) represents a static environment. However, real-world environments are dynamic. This problem can be addressed by constructing dynamic fuzzy environment representations by sequences of static fuzzy models $\mu_{E^k}(\mathbf{x}),\ \ k = 1, 2, \ldots$, each representing the dynamic environment over a small sampling interval. We shall call $\mu_{E^k}(\mathbf{x})$ the k^{th} episode of the dynamic environment heareafter. Each episode is then formulated as (15) and (16).

$$E^k = \tilde{O}^k \cup P \tag{15}$$

$$\mu_{E^k}(\mathbf{x}) = \mu_{\tilde{O}^k}(\mathbf{x}) \dot{+} \mu_P(\mathbf{x}) \tag{16}$$

where $\tilde{O}^k$ are fuzzy sets describing the obstacles in the k^{th} sampling interval. In this study, we assume that the vehicle's desired global path is available in terms of a sequence of waypoints and that the vehicle can localize the obstacles and estimate their velocities to create the environment representations at each sample time.

3. Simultaneous Path Adherence and Obstacle Avoidance Using Fuzzy Environment Model

Let us assume that the example fuzzy environment in Figure 4 is the k^{th} episode in the sequence of static fuzzy models that describe a dynamic environment over each sampling interval t_s. It can be observed that the contour lines of this episode of the environment, $C = \{\mathbf{x} \in \mathbf{R}^2 | \mu_{E^k}(\mathbf{x}) = \mu_{th}\}, \ \ \mu_{th} \in (0 \ \ 1]$, can serve as a family of paths for the vehicle to its target. Although these contours guide the vehicle from the initial fuzzy waypoint to the target, they differ in their desirability. The path that is part of a contour with a higher μ_{th} is preferred as it is closer to the crisp path, whereas only the contours corresponding to low values of μ_{th} can generate obstacle-free paths.

In this conflicting situation, the planning algorithm must rapidly adjust the contour the vehicle is following to prevent collision with obstacles. Specifically, when no obstacles are detected, the algorithm should adhere to a contour with a high μ_{th} to remain close to the global path. However, in the presence of obstacles, it should temporarily transition to a contour with a lower μ_{th} and follow it until the collision risk is mitigated. Once the danger is eliminated, the algorithm can restore the vehicle to its original contour. This adaptive contour selection can be achieved using the mechanism described in (17).

$$\mu_{th} = \alpha(1 - \mu_{\tilde{O}^k}(\mathbf{x}_k))^{d_o} \tag{17}$$

where $\mathbf{x}_k$ is the vehicle's current position, $\alpha \in (0 \ \ 1]$ is the desired membership degree of the vehicle in the fuzzy path when no obstacles are present, and $d_o \geq 0$ is a control parameter. It is important to note that the value of the contour selection variable μ_{th} changes continuously based on the vehicle's membership degree in the obstacle region rather than its direct distance to the obstacle.

The parameters α and d_o can be tuned by considering two extreme scenarios. In the first scenario, when no obstacles are around, $\mu_{\tilde{O}^k}(\mathbf{x}_k) = 0$, the vehicle is expected to follow the global path closely. This means that the vehicle should follow a contour line $C = \{\mathbf{x} \in \mathbf{R}^2 | \mu_{E^k}(\mathbf{x}) = \mu_{th}\}, \ \ \mu_{th} \in (0 \ \ 1]$ with a $\mu_{th} = \alpha$ close to one. For example, a value of $\alpha = 0.99$ means that the vehicle path matches the global path by %99 when no obstacles are detected.

In the second scenario, an obstacle suddenly appears in close proximity to the vehicle, leading to a high $\mu_{\tilde{O}^k}(\mathbf{x}_k)$. For example, if $\mu_{\tilde{O}^k}(\mathbf{x}_k) = 0.99$, the vehicle position matches the obstacle area by %99 and collision is imminent. Under such conditions, the vehicle must transition to a contour with a significantly lower μ_{th}, for instance, $\mu_{th} = 0.01$, to induce a substantial deviation from the path and effectively avoid the imminent collision. By substituting the value of α from the first scenario and the required μ_{th} for the second scenario into equation (17), the minimum value of the parameter d_o needed to enforce this sharp reduction in μ_{th} near obstacles can be determined. For the example values of $\alpha = 0.99$, $\mu_{\tilde{O}^k}(\mathbf{x}_k) = 0.99$, and a target contour of $\mu_{th} = 0.01$ for imminent obstacle avoidanece, a minimum d_o value of 0.9978 is obtained.

The vehicle's velocity can also be dynamically upper bounded by v_{max} given in (18) for increased safety near obstacles. This adaptive upper bound is equal to the maximum allowed velocity on the path, v_m, when no obstacle is inbound but reduces near the obstacles according to (18) in which $d_v \geq 0$ controls the decay rate.

$$v_{max} = v_m(1 - \mu_{\tilde{O}^k}(\mathbf{x}_k))^{d_v} \tag{18}$$

The parameter d_v in (18) can be adjusted by considering the worst-case scenario when an obstacle is suddenly detected in close proximity to the vehicle resulting in a high $\mu_{\tilde{O}^k}(\mathbf{x}_k)$. In such situations, the vehicle must deviate significantly and rapidly from the global path to avoid a collision. However, the speed at which this maneuver can be executed, commonly referred to as the cornering velocity, depends on several factors, including the vehicle's height, path width, and turning radius. To ensure safe navigation, d_v should be adjusted so that v_{max} remains below the vehicle's cornering velocity, which can be analytically determined based on vehicle characteristics and varying road conditions. For instance, if $\mu_{\tilde{O}^k}(\mathbf{x}_k) = 0.99$, the maximum vehicle velocity is $v_m = 10\ \ m/s$, and the maximum cornering velocity is $1\ \ m/s$, then the minimum required value of d_v to reduce the vehicle's speed below the cornering threshold is found from (18) to be 0.5.

With the desired planning behavior specified, the path planning problem of concern in this study can now be formulated as minimizing the cost function (19) over each sampling interval subject to the velocity constraint (18) and the membership threshold μ_{th} governed by (17). It is important to note that the variables μ_{th} in (17) and V_{max} in (18) remain constant (zero-order hold condition) over each planning interval.

$$f^k(\mathbf{x}) = \frac{1}{2}(\mu_{E^k}(\mathbf{x}) - \mu_{th})^2 + \int_{\bar{C}} \mu_{E^k}(\mathbf{x}) ds \tag{19}$$

The integration path $\bar{C}$ is the arc of the contour $C = \{\mathbf{x} \in \mathbf{R}^2 | \mu_{E^k}(\mathbf{x}) = \mu_{E^k}(\mathbf{x}_k)\}$ starting from the current vehicle's position $\mathbf{x}_k$ and ending at the nearest point on the contour to the target. The element ds also denotes the differential arc length of C. The integral term in (19) represents the cross-sectional area that $\bar{C}$ cuts from the 3D surface of $\mu_{E^k}(\mathbf{x})$.

Since the integration path, i.e., $\bar{C}$, is part of the integrand's contour, the value of $\mu_{E^k}(\mathbf{x})$ remains constant at a value of $\mu_{E^k}(\mathbf{x}_k)$ all over the integration path. As a result, the cost function simplifies to (20).

$$f^k(\mathbf{x}) = \frac{1}{2}(\mu_{E^k}(\mathbf{x}) - \mu_{th})^2 + \mu_{E^k}(\mathbf{x}_k) \int_{\bar{C}} ds \tag{20}$$

The integral term in (20) is now the length of the arc $\bar{C}$. That is the length of the remaining path on the vehicle's current contour starting from the vehicle position and extending to the nearest point on the same contour to the destination. Therefore, minimizing (20) would maintain the vehicle on an obstacle-free contour of the environment as close as possible to the core of the fuzzy path. At the same time, it moves the vehicle in such a way that the remaining path length to the target is continuously reduced.

An important property of the objective function in equation (20) is that its first term on the right-hand side is minimized when the vehicle moves perpendicular to the contour lines of $\mu_{E^k}(\mathbf{x})$ towards the μ_{th}, while the second term is minimized when the vehicle moves tangentially along these contour lines towards the target destination. Therefore the directions at which the frist and second terms of the objective function (20) should move to achieve their minimums are perpendicular. This property, combined with the fact that both terms on the right-hand side of equation (20) are non-negative, allows us to minimize the objective function by independently minimizing each term.

Due to the differentiability of $\mu_{E^k}(\mathbf{x})$, the gradient descent algorithm is applicable to minimize the first term in (20). For the second term, however, deriving an explicit equation for the contour $\bar{C}$ is not possible. Therefore, the gradient descent algorithm cannot be applied to minimize the integral term in (20) directly. Nevertheless, the descent direction, i.e., the direction that points to the minimum of the integral term, is known to be tangent to $\bar{C}$ because moving in this direction continuously reduces the remaining length of $\bar{C}$. This tangent direction can be computed by finding the normal to the gradient of $\mu_{E^k}(\mathbf{x})$ at $\mathbf{x} = \mathbf{x}_k$ by (21).

$$t(\mathbf{x}_k) = \pm R(\pi/2) \frac{\nabla_{\mathbf{x}} \mu_{E^k}(\mathbf{x})}{||\nabla_{\mathbf{x}} \mu_{E^k}(\mathbf{x})||_2}\Big|_{\mathbf{x}=\mathbf{x}_k} \tag{21}$$

where $R(\pi/2)$ is the 2D rotation matrix by $\pi/2$ (rad). Since C is a closed contour, it can be followed to the target in either the clockwise or counterclockwise directions, leading to two choices for $\bar{C}$. The sign of $t(\mathbf{x}_k)$ in (21) determines which direction to follow. In the ideal case, the direction which leads to the minimum length for the remaining path should always be followed. Nevertheless, both the clockwise or counterclockwise directions lead to obstacle-free paths to the destination and can be selected by the planner. In this study, we always select the clockwise direction to enforce bypassing the obstacles from the left side (with respect to the vehicle's body frame) to be consistent with traffic regulations on actual roads.

The magnitude of the necessary tangential motion remains unknown, but it can be set to $v_{max}t_s$. This is, in fact, the amount of displacement over a planning interval t_s that results in the maximum safe velocity v_{max} determined earlier by (18).

The overall update rule for minimizing the cost in (20) is then found by the superposition of the gradient descent direction for the first term and the tangent element derived in (21) as follows,

$$\Delta\mathbf{x}_k = \left(-\eta_n(\mu_{E^k}(\mathbf{x}) - \mu_{th})\nabla_{\mathbf{x}}\mu_{E^k}(\mathbf{x}) \pm \eta_t R(\pi/2)\frac{\nabla_{\mathbf{x}}\mu_{E^k}(\mathbf{x})}{||\nabla_{\mathbf{x}}\mu_{E^k}(\mathbf{x})||_2}\right)\Big|_{\mathbf{x}=\mathbf{x}_k} \tag{22}$$

where $\eta_n > 0$ and $\eta_t > 0$ are step sizes in the normal (negative gradient) and tangent directions, respectively. The effect of the tangential and normal terms in (22) is demonstrated in Figure 5 considering the same environment as in Example 3. It can be observed in Figure 5 that the planned path is initially on the contour corresponding to $\mu_{th} = 0.92$. The normal component of the commanded path is almost zero when the vehicle is away from the obstacle. However, when the vehicle approaches the obstacle, the magnitude of the normal component in (22) increases, pushing the vehicle away. When the vehicle passes the obstacle, the normal component changes direction to bring the vehicle back to the initial $\mu_{th} = 0.92$ contour. Algorithm 1 summarizes the steps of the proposed fuzzy path-planning technique.

4. Simulations

The performance of the proposed fuzzy approach to path planning for autonomous vehicles is evaluated in simulations on a mobile robot. The robot is assumed to be governed by the kinematic model in (23) in which $\mathbf{x} = [x_1, \ x_2]^T$ denotes the vehicle's position, v is the linear velocity, and ω the angular velocity of the vehicle [37].

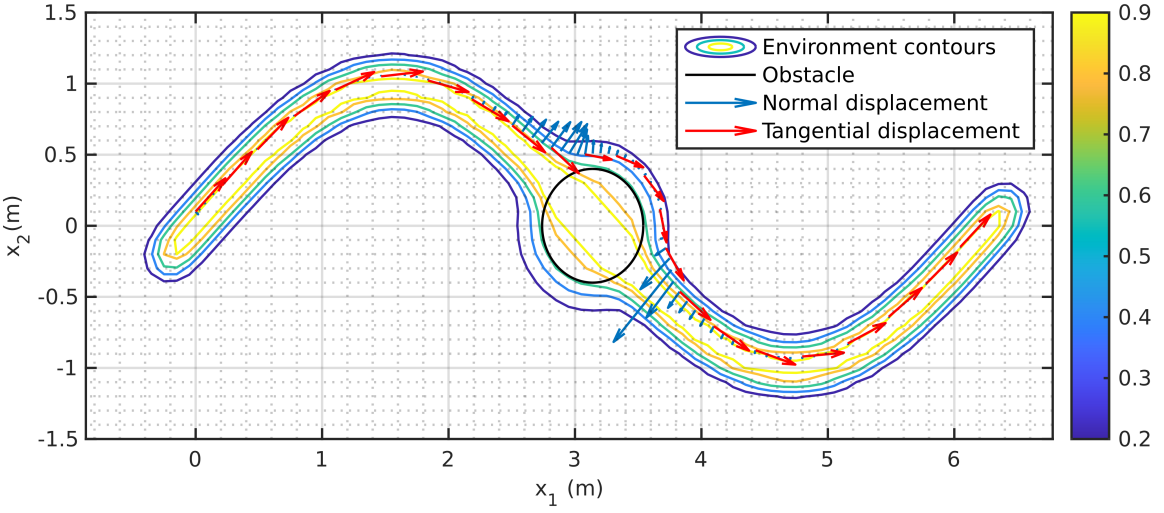

Figure 5: Effect of the tangential and normal components of the optimization rule.

Algorithm 1 The Fuzzy Path Planning

1: Detect the objects in the environment at the current planning interval.
2: Define the atomic sets for each object in terms of overlapping circles with appropriate centers and radii.
3: Derive the obstacle region for each atomic object using (2) and (3) and the BMF in (6).
4: Form the collective obstacle region using (4) and (5).
5: Find the fuzzy waypoints from the desired crisp ones using (8) to (10).
6: Generate the overall fuzzy path using (11) and (12).
7: Derive the overall fuzzy representation of the environment for the current planning interval by (15), and (16).
8: Apply (22) and (17) to move the vehicle while upper bounding the velocity by (18).

The variables $\dot{x}_1$ and $\dot{x}_2$ are rates of changes of the vehicle's position along each axis, and $\dot{\psi}$ is the rate of change of the vehicle's heading ψ.

$$\begin{aligned} \dot{x}_1 &= \upsilon \cos(\psi) \\ \dot{x}_2 &= \upsilon \sin(\psi) \\ \dot{\psi} &= \omega \end{aligned} \tag{23}$$

For simplicity, the vehicle is assumed to be a point object, which is justified as its dimensions can be absorbed in the obstacle definitions. The robot should maintain a safe distance of $r_i = 1.25 \ \ (m), \ \ \text{for all} \ \ i$, from each obstacle. This safe distance considers the vehicle's dimensions and the required clearance for bypassing the obstacles. Three simulation case studies are considered. The last two simulations compare the performance of the fuzzy path-planning approach (FPPA) with the improved artificial potential field (IAPF) technique presented in [37]. The IAPF method is implemented with the same parameters reported in [37] for a concise comparison. In all simulations, the sampling time is $t_s = 50 \ \ (ms)$, and Yager's t-conorm with $p = 1$ implements all fuzzy unions with tanh(.) approximation of the saturation function in (7). The rates of changes of position and heading of the vehicle, i.e., its linear and angular velocities, are obtained by direct differentiation of position and heading of the vehicle, respectively.

4.1. Case Study 1

This particular case study presents a seemingly uncomplicated yet intricate scenario wherein a vehicle navigates directly toward a symmetric concave-shaped object. Traditional APFs encounter a notable challenge in such instances, as they tend to converge towards a local minimum. This convergence occurs due to the counteraction between the repulsion force exerted by the obstacle and the attraction force along the intended path, resulting in a stall near the obstacle. Consequently, this hinders the mobile robot from traversing the remaining section of the intended path. The IAPF technique introduces an effective solution to address this issue by incorporating supplementary sub-target points. These points serve the purpose of guiding the robot to navigate around the obstacle, thereby ensuring the uninterrupted

continuation of its path. In contrast, the proposed FPPA method offers an alternative approach by tactfully tracing the contours within the fuzzy environment. This approach allows for circumventing the obstacles, enabling the vehicle's seamless progression along its intended path.

The simulated desired path, in this case, is a line commencing at coordinates $[0, \quad 20]^T$ and terminating at coordinates [200, 20]. For this specific case, the parameters employed for FPPA are selected as follows: $b = 1.25$, $\alpha = 0.99$, $\delta = 0.1$, $v_m = 5 \ \ (m/s)$, $d_v = 0.5$, $d_o = 1.25$, $\eta_t = v_{max} t_s$, and $\eta_n = 0.75$. A set of $M = 100$ waypoints are derived from the desired crisp path, ensuring a minimum separation distance of 2 (m) between each successive point. The corresponding fuzzy waypoints are then generated with the lateral deviation allowance specified by $\sigma_j = 1 \ \ (m), \quad j = 1, \dots, M$, and their overlap is set to $\gamma = 0.75$. Within the environment, a stationary obstacle is considered that assumes the form of an arc segment, spanning from $-3\pi/4$ to $3\pi/4$, with a radius of 30 (m), and the arc center at coordinates $\mathbf{x} = [70 \ \ 0]^T$. This obstacle is modeled by 97 atomic sets with $\lambda_i = 0.25, \quad i = 1, \dots, 97$, and their centers p_i selected uniformly on the arc.

Figure 6 shows the planned path by the FPPA. This illustration distinctly demonstrates the capability of FPPA to circumvent obstacles and seamlessly adhere to the desired path. Notably, the method showcases resilience against local minima, even when encountered with obstacles of concave nature. Figure 7 and Figure 8 depict the robot's linear velocity and heading angle, respectively, when it follows the path planned by the FPPA. Figure 9 and Figure 10 also demonstrate the resulting accelerations and yaw rates, respectively. The velocity and heading, as well as their rates of change shown in these figures, indicate that FPPA can generate smooth paths for the mobile robot to follow.

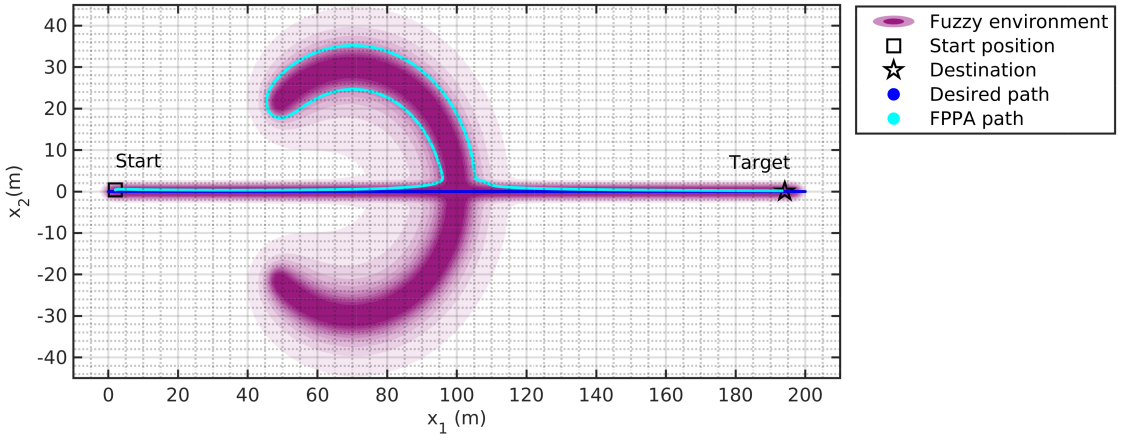

Figure 6: The paths generated by FPPA in Case Study 1.

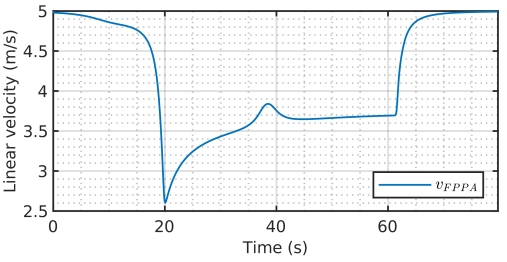

Figure 7: vehicle's velocity in Case Study 1.

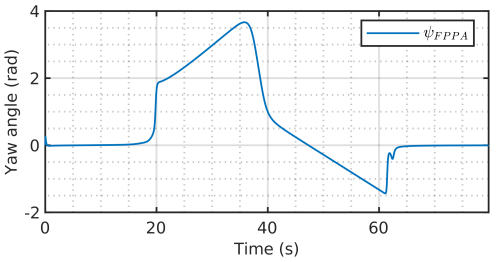

Figure 8: Vehicle's heading in Case Study 1.

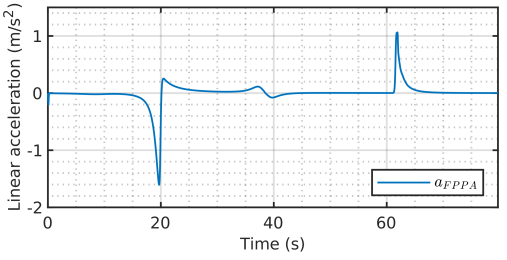

Figure 9: vehicle's acceleration in Case Study 1.

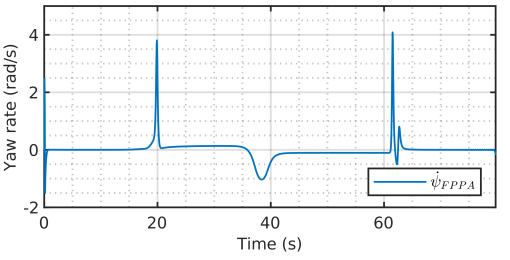

Figure 10: vehicle's yaw rate in Case Study 1.

4.2. Case Study 2

The second case study includes a more complex path with dynamic and static obstacles. This time, the waypoints are generated by sampling the desired path such that the resulting points are at least 5 (m) away from each other. Table 1 summarizes obstacles' location and motion properties. Parameters of the FPPA algorithm are also selected as $\delta = 0.1$, $\sigma_j = 1.5$ (m), $j = 1, \dots, M$, $v_m = 5$ (m/s), $d_v = 0.5$, $d_o = 1.25$, $\alpha = 0.99$, $\eta_t = v_{max} t_s$, and $\eta_n = 0.75$. The IAPF method uses the same set of parameters as in [37]. Figure 11 compares the resulting path by the FPPA with that of the IAPF in the second case study. Notably, the FPPA method has selected a different path from the IAPF for bypassing obstacles O_1, O_2, and O_7. The reason is that in this simulation, FPPA is set to go around all obstacles clockwise, assuming no prior knowledge of the path and obstacle shapes. On the contrary, the IAPF technique tries to derive the direction leading to the minimum length path but at the cost of oversimplifying assumptions regarding obstacle geometries. Regardless of differences in the generated paths, both methods successfully follow the desired route and avoid the obstacles. Figure 12 shows the robot's linear velocities while following the paths generated by the FPPA and IAPF methods. Comparing linear velocities in Figure 12 confirms that FPPA's velocity management is

Table 1: Obstacles in Case Study 2

Object	O_1	O_2	O_3	O_4	O_5	O_6	O_7
Position (m)	$\begin{bmatrix} 69.5 \\ 76.5 \end{bmatrix}$	$\begin{bmatrix} 69 \\ 82.5 \end{bmatrix}$	$\begin{bmatrix} 186 \\ 53 \end{bmatrix}$	$\begin{bmatrix} 192 \\ 56 \end{bmatrix}$	$\begin{bmatrix} 185 \\ 48 \end{bmatrix}$	$\begin{bmatrix} 45 \\ 112 \end{bmatrix}$	$\begin{bmatrix} 79.5 \\ 10.2 \end{bmatrix}$
Radius (m)	2	3	2	3	3	3	3
Velocity (m/s)	0	0	0	0	0	3	2

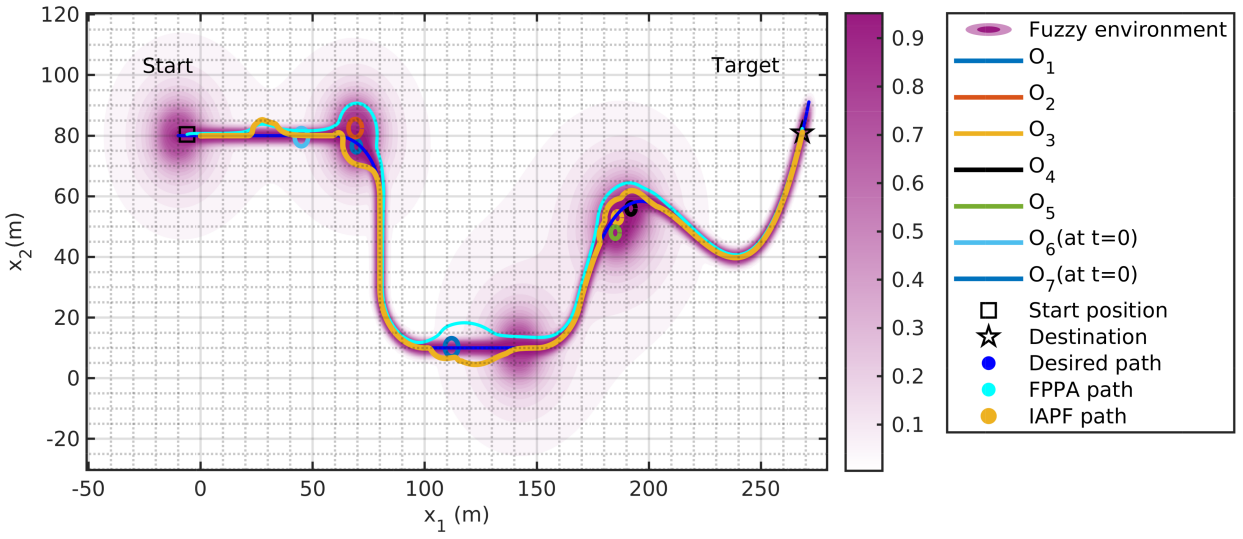

Figure 11: Comparison of the paths generated by FPPA and IAPF in Case Study 2.

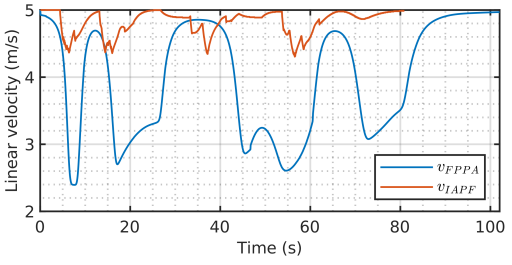

Figure 12: vehicle's velocity in Case Study 2.

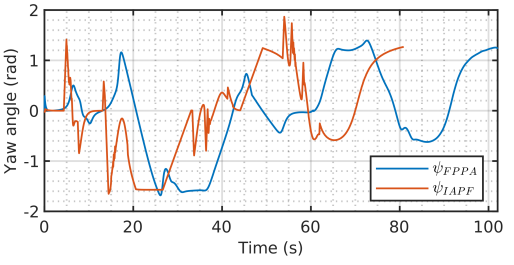

Figure 13: Vehicle's heading in Case Study 2.

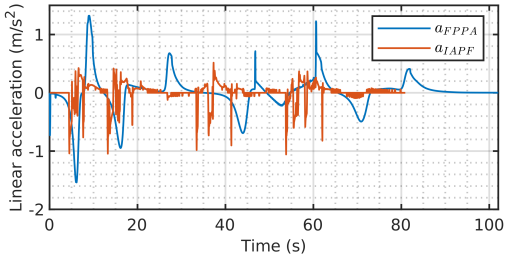

Figure 14: vehicle's acceleration in Case Study 2.

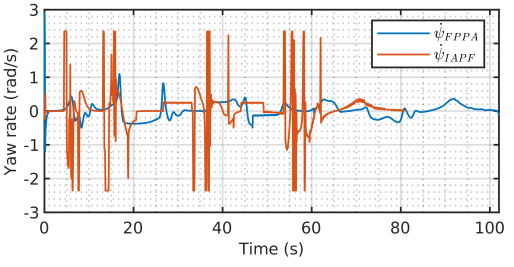

Figure 15: vehicle's yaw rate in Case Study 2.

far smoother than the IAPF technique. The same conclusion can be made by comparing the headings in Figure 13, accelerations in Figure14, and the yaw rates in Figure 15.

The acceleration and yaw rate plots further show that the IAPF method suffers from high-frequency oscillations and thus cannot be applied to an actual vehicle. The FPPA method, however, has generated reasonably smooth velocity and heading commands, rendering it a viable option for practical use.

Figure 16 also exhibits the significantly low computational burden of the proposed fuzzy path planning algorithm in this case study. This figure indicates that the execution time of each epoch of planning with FPPA takes less than almost

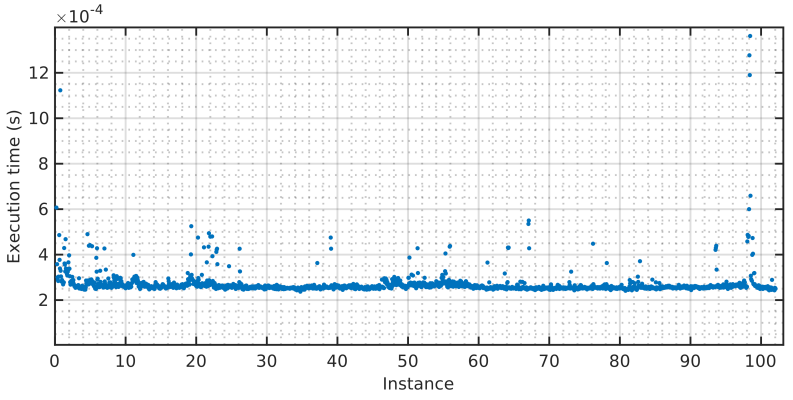

Figure 16: Computational complexity of the FPPA.

0.3 (ms) on a CoreI7 CPU clocked at 4.9 GHz. Comparing the planning time of the FPPA to the sampling time of 50 (ms) used in the simulation confirms the capability of this algorithm to operate in real-time.

4.3. Case Study 3

The final simulation is the same as Case Study 2. The only difference is that the maximum allowed velocity, v_m, is now raised to 12 (m/s). All other parameters of both the IAPF and FPPA algorithms are kept unchanged. Figure 17 compares the resulting path by FPPA and IAPF methods in this high-speed scenario. It shows that the IAPF planning procedure leads to collision with objects O_1, O_3, and O_4 when the same parameters as in the low-speed scenario in Case Study 2 are used. The FPPA, however, has successfully followed the path while maintaining the required safe distance from all obstacles. This observation indicates that the FPPA method can adapt to different velocities, whereas the IAPF method needs to be re-tuned. The robot's linear velocities using the FPPA and IAPF methods are also depicted in Figure 18, confirming the smooth velocity regulation capability of the FPPA method at higher speeds. Similarly, the comparison of the heading angles in Figure 19, linear accelerations in Figure 20, and the yaw rates in Figure 21

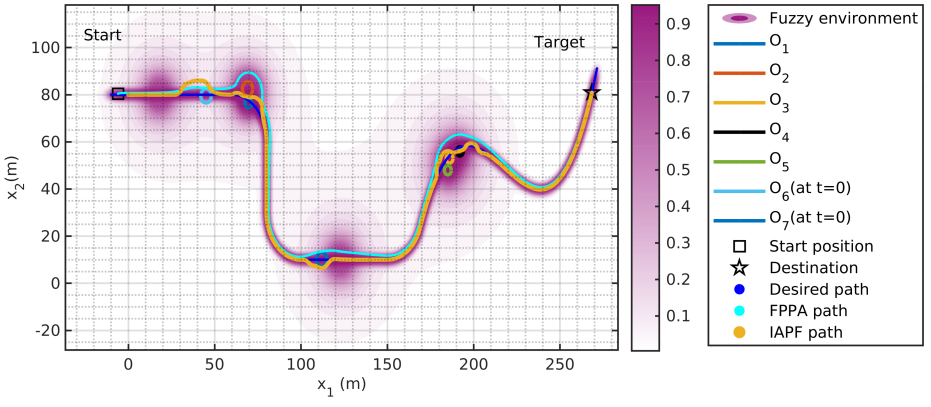

Figure 17: Comparison of the paths generated by FPPA and IAPF in Case Study 3.

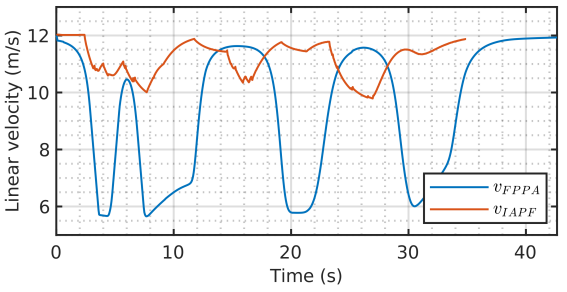

Figure 18: vehicle's velocity in Case Study 3.

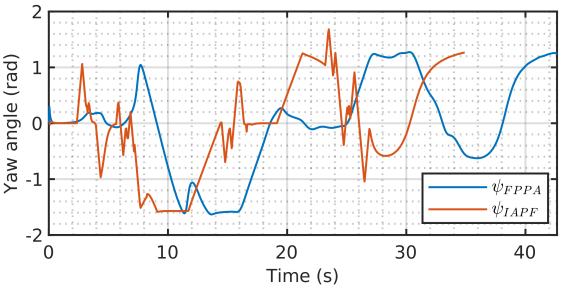

Figure 19: vehicle's heading in Case Study 3.

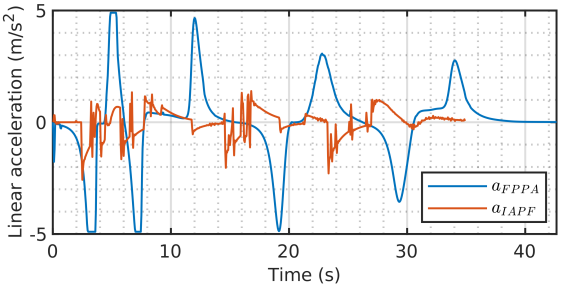

Figure 20: vehicle's acceleration in Case Study 3.

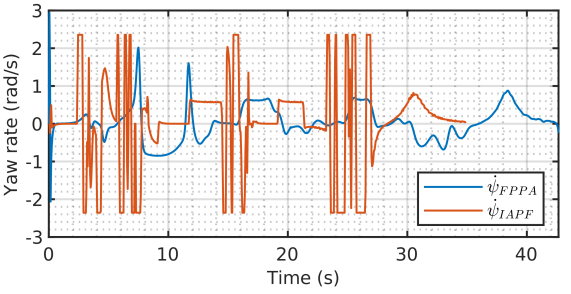

Figure 21: vehicle's yaw rate in Case Study 3.

further validate the smooth and oscillation-free performance of the FPPA at higher velocities whereas the IAPF fails to provide the minimum requirement of collision avoidance.

Table 2 summarizes the performance metrics achieved by the FPPA compared to IAPF and validated through simulation studies presented earlier in this section.

Table 2: Comparison of FPPA and IAPF in Path Planning for Autonomous Vehicles.

Feature	**FPPA**	**IAPF**
Support for Dynamic Environments	Yes	Yes
Smooth Path Generation	Yes	No
Path Comfort Optimization	Minimal	Yes
Real-Time Planning Capability	Yes	Yes
Complex Environment Handling	Yes	No
Adaptability to Uncertain Environments	Yes	No
Path Length	Longer	Shorter
Adherence to Global Path	Approximate	Exact
Planning Approach	Optimization-Based	Geometry-Based

5. Conclusions

This paper makes significant contributions to the field of autonomous vehicle navigation on two fronts. First, it introduces a new fuzzy set-based methodology for representing the dynamic work environment of autonomous vehicles. This representation is distinguished by its computational efficiency, compactness, and capacity to address the uncertainty inherent in environment perception. Second, it proposes a solution to simultaneous path adherence and collision avoidance problems in autonomous navigation utilizing the introduced fuzzy model. The presented planning approach formulates this problem as a sequence of optimization problems over regular planning intervals. The gradient-based solutions to these problems create an obstacle-free path for the vehicle with minimal deviation from the global route. The effectiveness of the proposed fuzzy path-planning approach (FPPA) is validated through extensive simulation studies using a mobile robot. Comparative analysis against a state-of-the-art Artificial Potential Field (APF)-based technique highlights the superior performance of the FPPA, particularly in achieving smoother paths with significantly lower computational overhead. Additionally, the FPPA demonstrates the ability to avoid unexpected obstacles, provided they are detected with sufficient time for evasive maneuvers. These findings underscore the practical reliability and effectiveness of the proposed methodology for autonomous vehicle navigation. However, the FPPA does face practical limitations. Particularly, maintaining FPPA's real-time performance becomes increasingly challenging as the number of obstacles grows, given the algorithm's reliance on a dedicated mapping process to construct the fuzzy environment model from sensor data. This mapping step imposes substantial computational demands, particularly in large, obstacle-dense environments, which can adversely impact the real-time operation of the navigation system. Addressing these challenges requires further research into the integration of efficient mapping techniques with fuzzy-based models. Despite these limitations, the proposed approach represents a significant advancement in autonomous vehicle path planning. It lays a solid foundation for future research and innovation in developing more robust, scalable, and efficient navigation systems for complex and dynamic environments.

Appendix A

The following is proof of Theorem 1.

Proof. Setting $m = 2$ in (7) reduces it to the binary version of the Yager's t-conorm in (1); thus, (7) holds for two operands. Now assuming that (7) also holds for $m - 1$ operands as expressed in (A.1), the overall t-conorm of the m operands would be (A.2).

$$s = a_1 \dot{+} \ldots, \dot{+} a_{m-1} = \min\left(1, \left(\sum_{i=1}^{m-1} a_i^p\right)^{1/p}\right) \tag{A.1}$$

$$a_1 \dot{+} \ldots \dot{+} a_m = s \dot{+} a_m = \min\left(1, \left(s^p + a_m^p\right)^{1/p}\right) \tag{A.2}$$

Two possibilities can be considered regarding the t-conorm in (8). The first possibility is that $s < 1$, which reduces (A.1) to (A.3).

$$s = a_1 \dot{+} \ldots, \dot{+} a_{m-1} = \left(\sum_{i=1}^{m-1} a_i^p\right)^{1/p} \tag{A.3}$$

Substituting (A.3) back into (A.2) results in (7) indicating that the theorem holds when $s < 1$. The second possibility is that $s = 1$, which according to (A.1) indicates that the inequality (A.4) must hold.

$$\left(\sum_{i=1}^{m-1} a_i^p\right)^{1/p} \geq 1 \tag{A.4}$$

Using (A.4) in the original theorem, (7), results in $a_1 \dot{+} \ldots, \dot{+} a_m = 1$. Similarly, the binary Yager's t-conorm in (A.3) with $s = 1$ results in $a_1 \dot{+} \ldots \dot{+} a_m = s \dot{+} a_m = 1 \dot{+} a_m = 1$. Therefore, the theorem also holds in the second scenario, which completes the proof by induction. □

CRediT authorship contribution statement

Ehsan Adel Rastkhiz: Conceptualization, Methodology, Software, Writing - original draft, Visualization. **Howard Schwartz:** Supervision, Conceptualization, Writing - review & editing, Funding acquisition. **Ioannis Lambadaris:** Supervision, Conceptualization, Writing - review & editing, Funding acquisition.

Acknowledgement

We acknowledge financial support from Ericsson and NSERC (Natural Sciences and Engineering Research Council) of Canada.

References

[1] D. Chen, X. Zhang, D. Gao, K. Gao, M. Wen, Z. Huang, Logistics Distribution Path Planning Based on Fireworks Differential Algorithm, in: 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2020, pp. 2797–2802. doi:10.1109/SMC42975.2020.9283001.

[2] P. Du, X. He, H. Cao, S. Garg, G. Kaddoum, M. M. Hassan, AI-based energy-efficient path planning of multiple logistics UAVs in intelligent transportation systems, Computer Communications 207 (2023) 46–55. doi:10.1016/j.comcom.2023.04.032.

[3] J. Jiang, Y. Ma, Path Planning Strategies to Optimize Accuracy, Quality, Build Time and Material Use in Additive Manufacturing, Micromachines 11 (7) (2020) 633. doi:10.3390/mi11070633.

[4] K. S. Suresh, R. Venkatesan, S. Venugopal, Mobile robot path planning using multi-objective genetic algorithm in industrial automation, Soft Computing 26 (15) (2022) 7387–7400. doi:10.1007/s00500-022-07300-8.

[5] Z. Jiang, Research on Multi-robot Material Picking and Autonomous Path Planning System in Industrial Environment, in: 2023 IEEE 2nd International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA), IEEE, Changchun, China, 2023, pp. 1948–1952. doi:10.1109/EEBDA56825.2023.10090823.

[6] M. Goutham, S. Boyle, M. Menon, S. Mohan, S. Garrow, S. S. Stockar, Optimal Path Planning Through a Sequence of Waypoints, IEEE Robotics and Automation Letters 8 (3) (2023) 1509–1514. doi:10.1109/LRA.2023.3240662.

[7] Z. Jiao, K. Ma, Y. Rong, P. Wang, H. Zhang, S. Wang, A path planning method using adaptive polymorphic ant colony algorithm for smart wheelchairs, Journal of Computational Science 25 (2018) 50–57. doi:10.1016/j.jocs.2018.02.004.

[8] Y. Jung, Y. Kim, W. H. Lee, M. S. Bang, Y. Kim, S. Kim, Path Planning Algorithm for an Autonomous Electric Wheelchair in Hospitals, IEEE Access 8 (2020) 208199–208213. doi:10.1109/ACCESS.2020.3038452.

[9] A. Merei, H. Mcheick, A. Ghaddar, Survey on Path Planning for UAVs in Healthcare Missions, Journal of Medical Systems 47 (1) (2023) 79. doi:10.1007/s10916-023-01972-x.

[10] Chenghui Cai, S. Ferrari, Information-Driven Sensor Path Planning by Approximate Cell Decomposition, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 39 (3) (2009) 672–689. doi:10.1109/TSMCB.2008.2008561.

[11] L. Kavraki, P. Svestka, J.-C. Latombe, M. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces, IEEE Transactions on Robotics and Automation 12 (4) (1996) 566–580. doi:10.1109/70.508439.

[12] F. Samaniego, J. Sanchis, S. García-Nieto, R. Simarro, Recursive Rewarding Modified Adaptive Cell Decomposition (RR-MACD): A Dynamic Path Planning Algorithm for UAVs, Electronics 8 (3) (2019) 306. doi:10.3390/electronics8030306.

[13] E. W. Dijkstra, A note on two problems in connexion with graphs, Numerische Mathematik 1 (1) (1959) 269–271. doi:10.1007/BF01386390.

[14] J. Mohanta, A. Keshari, A knowledge based fuzzy-probabilistic roadmap method for mobile robot navigation, Applied Soft Computing 79 (2019) 391–409. doi:10.1016/j.asoc.2019.03.055.

[15] O. A. A. Salama, M. E. H. Eltaib, H. A. Mohamed, O. Salah, RCD: Radial Cell Decomposition Algorithm for Mobile Robot Path Planning, IEEE Access 9 (2021) 149982–149992. doi:10.1109/ACCESS.2021.3125105.

[16] P. Hart, N. Nilsson, B. Raphael, A Formal Basis for the Heuristic Determination of Minimum Cost Paths, IEEE Transactions on Systems Science and Cybernetics 4 (2) (1968) 100–107. doi:10.1109/TSSC.1968.300136.

[17] F. Duchoň, A. Babinec, M. Kajan, P. Beňo, M. Florek, T. Fico, L. Jurišica, Path Planning with Modified a Star Algorithm for a Mobile Robot, Procedia Engineering 96 (2014) 59–69. doi:10.1016/j.proeng.2014.12.098.

[18] G. Tang, C. Tang, C. Claramunt, X. Hu, P. Zhou, Geometric A-Star Algorithm: An Improved A-Star Algorithm for AGV Path Planning in a Port Environment, IEEE Access 9 (2021) 59196–59210. doi:10.1109/ACCESS.2021.3070054.

[19] M. M. Zafar, M. L. Anjum, W. Hussain, LTA*: Local tangent based A* for optimal path planning, Autonomous Robots 45 (2) (2021) 209–227. doi:10.1007/s10514-020-09956-3.

[20] C. Liu, Q. Mao, X. Chu, S. Xie, An Improved A-Star Algorithm Considering Water Current, Traffic Separation and Berthing for Vessel Path Planning, Applied Sciences 9 (6) (2019) 1057. doi:10.3390/app9061057.

[21] R. Zhen, Q. Gu, Z. Shi, Y. Suo, An Improved A-Star Ship Path-Planning Algorithm Considering Current, Water Depth, and Traffic Separation Rules, Journal of Marine Science and Engineering 11 (7) (2023) 1439. doi:10.3390/jmse11071439.

[22] A. Stentz, Optimal and efficient path planning for partially-known environments, in: Proceedings of the 1994 IEEE International Conference on Robotics and Automation, 1994, pp. 3310–3317 vol.4. doi:10.1109/ROBOT.1994.351061.

[23] A. Stentz, The Focussed D* Algorithm for Real-Time Replanning, in: International Joint Conference on Artificial Intelligence, 1995, p. 1652–1659.

[24] S. Koenig, M. Likhachev, Fast replanning for navigation in unknown terrain, IEEE Transactions on Robotics 21 (3) (2005) 354–363. doi:10.1109/TRO.2004.838026.

[25] S. M. LaValle, Rapidly-exploring random trees : a new tool for path planning, The annual research report (1998).

[26] J. Kuffner, S. LaValle, RRT-connect: An efficient approach to single-query path planning, in: Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), Vol. 2, IEEE, San Francisco, CA, USA, 2000, pp. 995–1001. doi:10.1109/ROBOT.2000.844730.

[27] S. Karaman, E. Frazzoli, Sampling-based algorithms for optimal motion planning, The International Journal of Robotics Research 30 (7) (2011) 846–894. doi:10.1177/0278364911406761.

[28] S. M. LaValle, J. James J. Kuffner, Randomized Kinodynamic Planning, The International Journal of Robotics Research 20 (5) (2001) 378–400. doi:10.1177/02783640122067453.

[29] D. Hsu, R. Kindel, J.-C. Latombe, S. Rock, Randomized Kinodynamic Motion Planning with Moving Obstacles, The International Journal of Robotics Research 21 (3) (2002) 233–255. doi:10.1177/027836402320556421.

[30] I.-B. Jeong, S.-J. Lee, J.-H. Kim, Quick-RRT*: Triangular inequality-based implementation of RRT* with improved initial solution and convergence rate, Expert Systems with Applications 123 (2019) 82–90. doi:10.1016/j.eswa.2019.01.032.

[31] Y. Li, W. Wei, Y. Gao, D. Wang, Z. Fan, PQ-RRT*: An improved path planning algorithm for mobile robots, Expert Systems with Applications 152 (2020) 113425. doi:10.1016/j.eswa.2020.113425.

[32] S. A. Eshtehardian, S. Khodaygan, A continuous RRT*-based path planning method for non-holonomic mobile robots using B-spline curves, Journal of Ambient Intelligence and Humanized Computing 14 (7) (2023) 8693–8702. doi:10.1007/s12652-021-03625-8.

[33] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, in: Proceedings. 1985 IEEE International Conference on Robotics and Automation, Vol. 2, 1985, pp. 500–505. doi:10.1109/ROBOT.1985.1087247.

[34] J.-O. Kim, P. Khosla, Real-time obstacle avoidance using harmonic potential functions, IEEE Transactions on Robotics and Automation 8 (3) (1992) 338–349. doi:10.1109/70.143352.

[35] T. Luan, Z. Tan, B. You, M. Sun, H. Yao, Path planning of unmanned surface vehicle based on artificial potential field approach considering virtual target points, Transactions of the Institute of Measurement and Control (2023) 01423312231190208doi:10.1177/01423312231190208.

[36] G. Hao, Q. Lv, Z. Huang, H. Zhao, W. Chen, UAV Path Planning Based on Improved Artificial Potential Field Method, Aerospace 10 (6) (2023) 562. doi:10.3390/aerospace10060562.

[37] W. Li, Y. Wang, S. Zhu, J. Xiao, S. Chen, J. Guo, D. Ren, J. Wang, Path Tracking and Local Obstacle Avoidance for Automated Vehicle Based on Improved Artificial Potential Field, International Journal of Control, Automation and Systems 21 (5) (2023) 1644–1658. doi:10.1007/s12555-022-0183-8.

[38] X. Xia, T. Li, S. Sang, Y. Cheng, H. Ma, Q. Zhang, K. Yang, Path Planning for Obstacle Avoidance of Robot Arm Based on Improved Potential Field Method, Sensors 23 (7) (2023) 3754. doi:10.3390/s23073754.

[39] L. Zheng, W. Yu, G. Li, G. Qin, Y. Luo, Particle Swarm Algorithm Path-Planning Method for Mobile Robots Based on Artificial Potential Fields, Sensors 23 (13) (2023) 6082. doi:10.3390/s23136082.

[40] Y. Chen, G. Bai, Y. Zhan, X. Hu, J. Liu, Path Planning and Obstacle Avoiding of the USV Based on Improved ACO-APF Hybrid Algorithm With Adaptive Early-Warning, IEEE Access 9 (2021) 40728–40742. doi:10.1109/ACCESS.2021.3062375.

[41] Z. Wu, J. Dai, B. Jiang, H. R. Karimi, Robot path planning based on artificial potential field with deterministic annealing, ISA Transactions 138 (2023) 74–87. doi:10.1016/j.isatra.2023.02.018.

[42] L. Xiangde, Z. Xiang, Z. Yi, L. Hua, H. Wentao, Global Dynamic Path Planning Algorithm Based on Harmony Search Algorithm and Artificial Potential Field Method, in: 2022 IEEE 6th Information Technology and Mechatronics Engineering Conference (ITOEC), IEEE, Chongqing, China, 2022, pp. 346–351. doi:10.1109/ITOEC53115.2022.9734602.

[43] H.-T. Chiang, N. Malone, K. Lesser, M. Oishi, L. Tapia, Path-guided artificial potential fields with stochastic reachable sets for motion planning in highly dynamic environments, in: 2015 IEEE International Conference on Robotics and Automation (ICRA), IEEE, Seattle, WA, USA, 2015, pp. 2347–2354. doi:10.1109/ICRA.2015.7139511.

[44] Y. Chen, P. Wang, Z. Lin, C. Sun, Global path guided vehicle obstacle avoidance path planning with artificial potential field method, IET Cyber-Systems and Robotics 5 (1) (2023) e12082. doi:10.1049/csy2.12082.

[45] L. Liu, B. Wang, H. Xu, Research on Path-Planning Algorithm Integrating Optimization A-Star Algorithm and Artificial Potential Field Method, Electronics 11 (22) (2022) 3660. doi:10.3390/electronics11223660.

[46] L. Zadeh, Fuzzy sets, Information and Control 8 (3) (1965) 338–353. doi:10.1016/S0019-9958(65)90241-X.

[47] R. R. Yager, On a general class of fuzzy connectives, Fuzzy Sets and Systems 4 (3) (1980) 235–242. doi:10.1016/0165-0114(80)90013-5.

[48] M. Al-Khatib, J. J. Saade, An efficient data-driven fuzzy approach to the motion planning problem of a mobile robot, Fuzzy Sets and Systems 134 (1) (2003) 65–82. doi:https://doi.org/10.1016/S0165-0114(02)00230-0.

[49] L.-D. Bui, Y.-G. Kim, An obstacle-avoidance technique for autonomous underwater vehicles based on BK-products of fuzzy relation, Fuzzy Sets and Systems 157 (4) (2006) 560–577. doi:10.1016/j.fss.2005.05.042.

[50] M. Wang, J. N. Liu, Fuzzy logic-based real-time robot navigation in unknown environment with dead ends, Robotics and Autonomous Systems 56 (7) (2008) 625–643. doi:10.1016/j.robot.2007.10.002.

[51] O. R. E. Motlagh, T. S. Hong, N. Ismail, Development of a new minimum avoidance system for a behavior-based mobile robot, Fuzzy Sets and Systems 160 (13) (2009) 1929–1946. doi:10.1016/j.fss.2008.09.015.

[52] O. Motlagh, S. Tang, N. Ismail, A. Ramli, An expert fuzzy cognitive map for reactive navigation of mobile robots, Fuzzy Sets and Systems 201 (2012) 105–121. doi:10.1016/j.fss.2011.12.013.

[53] L.-X. Wang, Further Operations on Fuzzy Sets, in: A Course in Fuzzy Systems and Control, Prentice-Hall, Inc., 1996, pp. 34–47.

[54] M. Javadian, A. Malekzadeh, G. Heydari, S. Bagheri Shouraki, A clustering fuzzification algorithm based on ALM, Fuzzy Sets and Systems 389 (2020) 93–113. doi:10.1016/j.fss.2019.10.013.

[55] A. Farooq, K. H. Memon, Kernel possibilistic fuzzy c-means clustering algorithm based on morphological reconstruction and membership filtering, Fuzzy Sets and Systems 477 (2024) 108792. doi:10.1016/j.fss.2023.108792.

[56] M. Palangetić, C. Cornelis, S. Greco, R. Słowiński, Multi-class granular approximation by means of disjoint and adjacent fuzzy granules, Fuzzy Sets and Systems 478 (2024) 108765. doi:10.1016/j.fss.2023.108765.

[57] P. D'Urso, J. M. Leski, Fuzzy clustering of fuzzy data based on robust loss functions and ordered weighted averaging, Fuzzy Sets and Systems 389 (2020) 1–28. doi:10.1016/j.fss.2019.03.017.

[58] Y. Jin, M. Hoffmann, A. Deligiannis, J.-C. Fuentes-Michel, M. Vossiek, Semantic Segmentation-Based Occupancy Grid Map Learning With Automotive Radar Raw Data, IEEE Transactions on Intelligent Vehicles 9 (1) (2024) 216–230. doi:10.1109/TIV.2023.3322353.

[59] C. Lu, M. J. G. Van De Molengraft, G. Dubbelman, Monocular Semantic Occupancy Grid Mapping With Convolutional Variational Encoder–Decoder Networks, IEEE Robotics and Automation Letters 4 (2) (2019) 445–452. doi:10.1109/LRA.2019.2891028.

[60] M. Schreiber, V. Belagiannis, C. Glaser, K. Dietmayer, A Multi-Task Recurrent Neural Network for End-to-End Dynamic Occupancy Grid Mapping, in: 2022 IEEE Intelligent Vehicles Symposium (IV), IEEE, 2022, pp. 315–322. doi:10.1109/IV51971.2022.9827360.