

OkayPlan: Obstacle Kinematics Augmented Dynamic Real-time Path Planning via Particle Swarm Optimization

Jinghao Xin^a, Jinwoo Kim^b, Shengjia Chu^a and Ning Li^{a,*}

^aDepartment of Automation, Shanghai Jiao Tong University, Shanghai 200240, P.R. China

^bSchool of Civil and Environmental Engineering, Nanyang Technological University, S639798, Singapore

ARTICLE INFO

Keywords:

Path planning
Dynamic environment
Real-time planning
Unmanned surface vehicles
Particle swarm optimization

ABSTRACT

Existing Global Path Planning (GPP) algorithms predominantly presume planning in a static environment. This assumption immensely limits their applications to Unmanned Surface Vehicles (USVs) that typically navigate in dynamic environments. To address this limitation, we present OkayPlan, a GPP algorithm capable of generating safe and short paths in dynamic scenarios at a real-time executing speed (125 Hz on a desktop-class computer). Specifically, we approach the challenge of dynamic obstacle avoidance by formulating the path planning problem as an obstacle kinematics augmented optimization problem, which can be efficiently resolved through a PSO-based optimizer at a real-time speed. Meanwhile, a Dynamic Prioritized Initialization (DPI) mechanism that adaptively initializes potential solutions for the optimization problem is established to further ameliorate the solution quality. Additionally, a relaxation strategy that facilitates the autonomous tuning of OkayPlan's hyperparameters in dynamic environments is devised. Comparative experiments involving canonical and contemporary GPP algorithms, along with ablation studies, have been conducted to substantiate the efficacy of our approach. Results indicate that OkayPlan outstrips existing methods in terms of path safety, length optimality, and computational efficiency, establishing it as a potent GPP technique for dynamic environments. The video and code associated with this paper are accessible at <https://github.com/XinJingHao/OkayPlan>.

1. Introduction


Unmanned Surface Vehicles (USVs), a novel class of vessels endowed with advanced capabilities such as autonomous perception, navigation, and mission execution, have been widely employed in various domains, including ocean resource exploration (Jin et al., 2018), maritime traffic monitoring (Liu et al., 2016), environmental protection (Li et al., 2018), and water rescue (Schofield et al., 2018). As USVs are generally characterized by delayed maneuverability, the safety of the navigation path is crucial for the successful execution of their unmanned tasks, necessitating an efficient design of the path planning technique, one capable of adjusting to the dynamic ocean environment adaptively and promptly.


Path planning is typically bifurcated into two categories: Global Path Planning (GPP) and Local Path Planning (LPP). The GPP algorithm is tasked with generating a collision-free global path with minimal length given the environmental map, and the LPP algorithm governs the USVs in adhering to the global path while concurrently circumventing the newly emerged obstacles. The global path functions as a higher-level instruction for the LPP algorithm, underlying

the autonomous navigation of USVs. Consequently, the development of advanced GPP algorithms is of paramount importance and constitutes the main focus of this research.

Canonical GPP algorithms encompass three primary classes: search-based, sample-based, and metaheuristic-based. Search-based GPP algorithms, a type of approach that searches for viable paths by exploring nodes within graph-based or grid-based maps, can be traced back to the foundational Breadth-First Search (BFS; Moore, 1959). BFS explores nodes equidistant from the starting node stage-by-stage until a viable path is identified. This simple mechanism ensures the discovery of the shortest path in unweighted maps. Dijkstra's algorithm (Dijkstra, 1959) enhances BFS by incorporating edge weights into the exploration process, allowing for the identification of the optimal path within weighted maps. A* (Hart et al., 1968) further extends Dijkstra's algorithm by introducing a heuristic function to direct the exploration. The heuristic function estimates the cost to reach the target node from a given node, prioritizing the exploration of nodes with lower estimated costs, thereby assisting the algorithm in reducing the search space. Nonetheless, due to the fact that the construction of graph-based or grid-based maps can be challenging in intricate environments, the applications of search-based GPP algorithms are immensely impeded. In contrast, the sample-based GPP algorithms, such as Probabilistic Roadmaps (PRM; Kavraki et al., 1996) and Rapidly-exploring Random Trees (RRT; LaValle, 1998), construct feasible paths via sampling directly in the configured space, rendering them more straightforward and flexible. However, this advantage comes at the cost of compromising the path smoothness and length optimality. As for the metaheuristic-based GPP

*Corresponding author

 xjhzsj2019@sjtu.edu.cn (J. Xin); jinwoo.kim@ntu.edu.sg (J. Kim); chu-sjtu@sjtu.edu.cn (S. Chu); ning_li@sjtu.edu.cn (N. Li)

 <https://github.com/XinJingHao> (J. Xin);
<https://sites.google.com/view/jinwoo-kim> (J. Kim);
<https://academic532.github.io> (N. Li)

ORCID(s): 0000-0001-9875-9098 (J. Xin); 0000-0002-2237-4965 (J. Kim); 0000-0003-1390-8512 (S. Chu); 0000-0003-1025-9641 (N. Li)

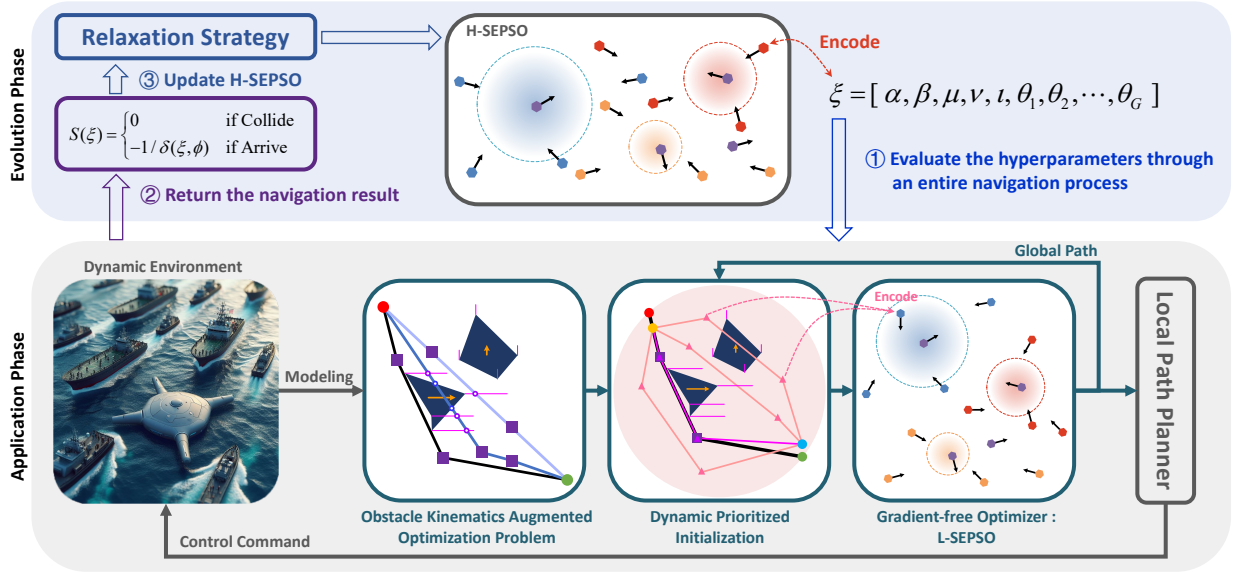


Fig. 1. An overview of OkayPlan. The algorithm encompasses two phases. The evolution phase is conducted beforehand to tune OkayPlan's hyperparameters autonomously. Having been equipped with the tuned hyperparameters, the algorithm is then employed to generate global paths for the USV and accomplish the navigation task in the application phase.

algorithms, the path planning problem is formulated as an optimization problem, with the goal of minimizing the path length, while collision constraints are incorporated as penalty terms. Conventionally, gradient-free metaheuristic optimizers, such as Genetic Algorithm (GA; [Holland, 1992](#)) and Particle Swarm Optimization (PSO; [Kennedy and Eberhart, 1995](#)), are resorted due to the non-differentiability of the optimization problem.

The canonical GPP algorithms discussed above typically exhibit a drawback of low computational efficiency, which hinders their application in situations that demand real-time planning. Specifically, the search-based and sample-based methods are significantly precluded by their sequential computation characteristics. That is, the exploration of the subsequent node can only be initiated upon the completion of the current node, making them temporally entangled and computationally inefficient. In the case of metaheuristic-based methods, a large population is crucial for obtaining a favorable solution. However, iterating over a large population can be time-consuming. To boost iteration speed, [Xin et al. \(2023a\)](#) has proposed the Self-Evolving Particle Swarm Optimization (SEPSO), where a Tensor Operation Form (TOF) is developed to facilitate parallel computing and thus expedites the iteration of a large population. It has been reported that the SEPSO-based GPP algorithm is capable of yielding collision-free paths at a real-time speed.

Although extensive GPP algorithms have been established to date, few of them have endeavored to accommodate the dynamics of the environments. The dynamics, as often introduced by the moving obstacles or other participating agents, can induce uncertainties in the environments, which potentially undermine the performance of existing GPP algorithms. Given that USVs generally operate in dynamic

environments, the ability to supplement pre-planned paths with dynamic adjustments becomes crucial. To this end, we propose the OkayPlan, an improved GPP algorithm based on SEPSO, capable of generating safe and near-optimal paths in dynamic environments with a real-time planning frequency of 125 Hz on a desktop-class computer. The key contributions that underprop the OkayPlan are summarized as follows:

- An obstacle kinematics augmented optimization problem has been formulated, allowing the motion of dynamic obstacles to be seamlessly incorporated into SEPSO's objective function, thereby facilitating the derivation of safer paths.
- A Dynamic Prioritized Initialization (DPI) mechanism has been introduced. The DPI adaptively initializes the population of SEPSO in accordance with different planning stages, which bolsters its optimization capability and contributes to the discovery of shorter paths in dynamic environments.
- A relaxation strategy has been established to subdue the stochasticity of the dynamic environments, enabling a better tuning of OkayPlan's hyperparameters.

The remainder of the paper is organized as follows. Section 2 provides a review of the fundamental concepts related to the SEPSO algorithm and the optimization problem formulation of GPP. Section 3 introduces three improvements of OkayPlan over SEPSO: the obstacle kinematics augmented optimization problem, the DPI mechanism, and the relaxation strategy. Section 4 presents the comparative experiments and the ablation studies. Finally, Section 5 draws the conclusions and outlines promising directions for future researches.

2. Preliminaries

2.1. Problem formulation of global path planning

As introduced by Xin et al. (2023a), the 2D GPP problem can be formulated as an optimization problem illustrated in Fig. 2, wherein the decision variables are the coordinates of the waypoints, as denoted below:

$$X = [x_1, x_2, \dots, x_{D/2}, y_1, y_2, \dots, y_{D/2}] \quad (1)$$

where $D/2$ corresponds to the number of the waypoints, with D denoting the dimension of the search space. The objective function of the optimization problem is given below:

$$F(X) = L(X) + \alpha \cdot Q(X)^\beta \quad (2)$$

$$L(X) = \sum_{d=2}^{D/2} \sqrt{(x_d - x_{d-1})^2 + (y_d - y_{d-1})^2} + \sqrt{(x_t - x_{D/2})^2 + (y_t - y_{D/2})^2} + \sqrt{(x_s - x_1)^2 + (y_s - y_1)^2} \quad (3)$$

where $L(X)$ denotes the path length; (x_t, y_t) and (x_s, y_s) are the coordinates of the target and start point; the parameters α and β govern the magnitude of the penalty incurred from collision, with $Q(X)$ denoting the count of intersections between the path and the bounding boxes of the obstacles. As shown in Fig. 2, $Q(\text{Path 1}) = 0$, and $Q(\text{Path 2}) = 2$.

Though minimizing Eq. (2), one is expecting to identify a collision-free path with minimal length.

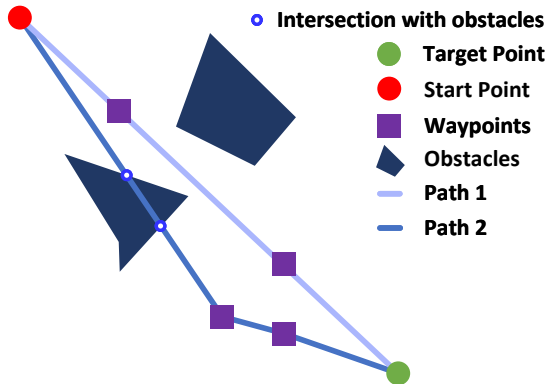


Fig. 2. Problem formulation of global path planning.

2.2. Self-evolving particle swarm optimization

Due to the non-differentiability of Eq. (2), gradient-based optimization methods are inapplicable. Although gradient-free optimization techniques such as GA and PSO are generally feasible, their unsatisfactory computational efficiency and susceptibility to local optima render them less than ideal for the GPP problem. Extensive attempts have

been undertaken to alleviate this limitation in recent decades (Guo et al., 2020; Li and Yu, 2023; Xin et al., 2023b). Among these, the Diversity-based Parallel Particle Swarm Optimization (DPPSO; Xin et al., 2023b) has drawn much attention due to its simple concept and robust performance. In DPPSO, the standard PSO is split into multiple groups, each of which performs diversified search parameters θ_g to preserve population diversity and thus avoid local optima. Meanwhile, the information of different groups is shared to ensure convergence. The pivotal update formulas of DPPSO are as follows:

$$\begin{aligned} V_{g,n}^{k+1} = & \omega_g^k V_{g,n}^k + C1_g \cdot R1_{g,n}^k \cdot (Pbest_{g,n}^k - X_{g,n}^k) \\ & + C2_g \cdot R2_{g,n}^k \cdot (Gbest_g^k - X_{g,n}^k) \\ & + C3_g \cdot R3_{g,n}^k \cdot (Tbest^k - X_{g,n}^k) \end{aligned} \quad (4)$$

$$X_{g,n}^{k+1} = X_{g,n}^k + V_{g,n}^{k+1} \quad (5)$$

$$\omega_g^k = \omega_g^{init} - \frac{\omega_g^{init} - \omega_g^{end}}{T} \cdot k \quad (6)$$

$$\theta_g = [C1_g, C2_g, C3_g, \omega_g^{init}, \omega_g^{end}, V_g^{limit}] \quad (7)$$

where k is the iteration counter; T is the maximum iteration times; g is the group's index within the total group number G ; n is the particle's index within its group, with a maximum value of N ; $V_{g,n}^k$ and $X_{g,n}^k$ are the velocity and position of the particle n in group g at iteration k , respectively; $R1_{g,n}^k, R2_{g,n}^k, R3_{g,n}^k$ are three uniform random variables within range $(0,1)$; $C1_g, C2_g, C3_g$ are three positive acceleration constants; ω_g^k is the inertia factor, with its initial and final value respectively given by ω_g^{init} and ω_g^{end} ; V_g^{limit} is the hyperparameter that control the range of the velocity of group g such that $V_g^{range} = V_g^{limit} \cdot X^{range}$, where X^{range} is the search space of the optimization problem being considered; $Pbest_{g,n}^k$ is the best solution identified by the particle n in group g within iteration k ; $Gbest_g^k$ is the best solution found by group g within iteration k ; $Tbest^k$ is the best solution discovered by the whole population within iteration k ; $Pbest_{g,n}^k, Gbest_g^k, Tbest^k$ are updated according to Algorithm 1 at each iteration.

Through the above iteration, the particles dynamically explore the search space and progressively converge towards potential optima. Furthermore, the diversified search parameters employed by DPPSO attain a more favorable balance between exploration and exploitation, consequently leading to improved optimization performance. Nonetheless, the DPPSO encounters two noteworthy unresolved challenges: 1) its calculation speed is prohibitively slow due to its particle-wise manipulation; 2) determining the diversified

Algorithm 1: Updating principle for the best values and positions

Input: particle $X_{g,n}^k$
Output: the best positions $Pbest_{g,n}^k, Gbest_g^k, Tbest^k$
 and the best values $U_{g,n}^{Pbest}, U_g^{Gbest}, U^{Tbest}$

// $f(\cdot)$ is the fitness function
 // For minimization problems:

```

1 if  $f(X_{g,n}^k) < U_{g,n}^{Pbest}$  then
2   |  $U_{g,n}^{Pbest} \leftarrow f(X_{g,n}^k)$  and  $Pbest_{g,n}^k \leftarrow X_{g,n}^k$ 
3 end
4 if  $f(X_{g,n}^k) < U_g^{Gbest}$  then
5   |  $U_g^{Gbest} \leftarrow f(X_{g,n}^k)$  and  $Gbest_g^k \leftarrow X_{g,n}^k$ 
6 end
7 if  $f(X_{g,n}^k) < U^{Tbest}$  then
8   |  $U^{Tbest} \leftarrow f(X_{g,n}^k)$  and  $Tbest^k \leftarrow X_{g,n}^k$ 
9 end

```

search parameter, denoted as θ (a collection of search parameters from all groups, such that $\theta = [\theta_1, \theta_2, \dots, \theta_g]$), poses a significant difficulty.

To address these two issues, [Xin et al. \(2023a\)](#) have introduced the SEPSO, which augmented the DPPSO with a TOF and a Hierarchical Self-Evolving Framework (HSEF). The TOF transforms the particle-wise manipulation into tensor operation, which endows the DPPSO with the feasibility to be deployed on the Graphics Processing Unit (GPU), thereby enabling parallel and rapid computation. Meanwhile, the HSEF facilitates autonomous tuning of the diversified search parameter θ . Specifically, the HSEF consists of a Higher-level Self-Evolving Particle Swarm Optimization (H-SEPSO) and a Lower-level Self-Evolving Particle Swarm Optimization (L-SEPSO). Both the H-SEPSO and the L-SEPSO are underpinned by the TOF-augmented DPPSO, each characterized by a distinct definition of particles and fitness functions. The L-SEPSO directly addresses the GPP problem, where each particle represents a potential path, and its fitness function corresponds to Eq. (2). Apparently, a lower fitness value indicates a more favorable path. Recall that the optimization capability of the L-SEPSO is notably conditioned by its search parameter θ . Then, the L-SEPSO can be conceptualized as a mapping from θ to the Lowest Fitness Value (LFV) it could optimize:

$$LFV = L\text{-SEPSO}(\theta) \quad (8)$$

Minimizing Eq. (8) holds the potential to identify a search parameter with better optimization capability. This optimization process is actually executed by the H-SEPSO, whose particle corresponds to the search parameter θ and fitness function aligns with Eq. (8). Though alternatively conducting the optimization of the L-SEPSO and H-SEPSO, θ autonomously evolves by itself and converges toward a more rational configuration ultimately. This entire process

is referred to as the evolution phase of SEPSO. Note that the evolution phase is computationally demanding, underscoring the imperative role of the efficient TOF.

2.3. Global path planning with SEPSO and prioritized initialization

Following the evolution phase outlined in the last section, the refined search parameter θ is then applied in conjunction with the TOF-augmented DPPSO to tackle the GPP problem formulated by Eq. (2), which is denoted as the application phase of the SEPSO. As widely recognized, the PSO-based optimization techniques are exceptionally suspicious to the initialization of the particles. A deliberate initialization serves not only to enhance solution quality but also to expedite convergence. In consideration of this, [Xin et al. \(2023a\)](#) have devised a Prioritized Initialization (PI) mechanism to assist particle initialization. As depicted by Fig. 3, at the commencement of each planning timestep, the PI mechanism inherits the planned path from the last timestep and randomly initializes a fraction of the population, denoted by γ , within the PI interval ρ . This strategy sets the preceding planned path as a reference for the current timestep, thereby averting the whole population from searching from scratch. The insights behind the PI mechanism pertain to the continuous change of the environment, implying that the optimal paths of two consecutive planning timesteps are likely to manifest gradual variations rather than abrupt change in most cases. In addition, to ensure a judicious level of exploration, the PI mechanism stochastically initializes the remaining population, represented by $1 - \gamma$, across the entire search space.

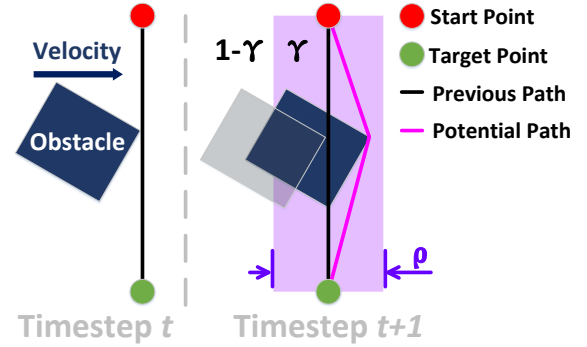


Fig. 3. Prioritized Initialization Mechanism.

3. Methodologies

Having comprehended the fundamental knowledge associated with the GPP and SEPSO, we proceed to introduce the OkayPlan. We commence by introducing the obstacle kinematics augmented optimization problem, wherein the motion of the obstacles is factored to derive safer paths. Following this, we propose a refinement of the PI, namely DPI, to initialize the population of SEPSO at different planning stages adaptively. Lastly, a relaxation strategy for the

evolution phase of the SEPSO has been developed, aiming to handle the stochasticity stemmed from the dynamic environments.

3.1. Obstacle kinematics augmented optimization problem

As illustrated by Fig. 4, the essential factor in guaranteeing safe navigation within dynamic environments is the prevention of "contention" originating from aggressive path planning. In particular, Fig. 4 (a) and (b) reveal that while the aggressive path is optimal in terms of path length, it may not be the safest choice and could lead to collision when navigating in dynamic environments. A more judicious way is to bypass the obstacles in a conservative fashion, as exemplified in Fig. 4 (c) and (d). This observation has uncovered a contradiction with the GPP's objective function delineated by Eq. (2), since we are seeking to ascertain a collision-free path of minimal length, but a newly emerged question pertains to how and to what extent should the length be shortened.

We endeavor to address this issue through incorporating the obstacle kinematics into the objective function. To be more specific, we measure the instantaneous velocities of the obstacles, represent them with segments, and integrate them into the configured space, as the purple segments shown in Fig. 5. The obstacle kinematics can be construed as a rough prediction of the obstacle's future trajectory and be harnessed to assist the path planning. In order to ensure the compatibility with the configured space, the length of the kinematic segments will be scaled by a hyperparameter, denoted as ι , in proportion to the instantaneous velocities. In this context, the obstacle kinematics augmented optimization problem for GPP in dynamic environments is given by:

$$F(X) = L(X) + \alpha \cdot Q(X)^\beta + \mu \cdot P(X)^\nu \quad (9)$$

where μ and ν are two hyperparameters that determine the magnitude of the penalty incurred from aggressive path planning, with $P(X)$ denoting the count of intersections between the path and kinematic segments. As shown in Fig. 5, $P(\text{Path 1}) = 1$, $P(\text{Path 2}) = 3$, and $P(\text{Path 3}) = 0$.

In terms of the motion prediction for the dynamic obstacles, various methodologies exist. A fancy way can be resorting to machine learning techniques, such as leveraging neural networks to forecast future trajectories (Hu et al., 2023; Kim et al., 2023). However, this approach may not be the most effective and practical due to the following reasons: 1) the computational demands of neural networks necessitate extra resources, posing a challenge for USVs with limited onboard computing capabilities; 2) the training of the prediction model requires additional efforts; 3) the generalization capability of the prediction model remains open to debate. In contrast, the advantages of our methods are evident: 1) the measurement of the obstacles' instantaneous velocities is tractable, generalizable, and computationally acceptable; 2) the kinematic segments can be seamlessly integrated into Eq. (2), with merely an extra term $\mu \cdot P(X)^\nu$.

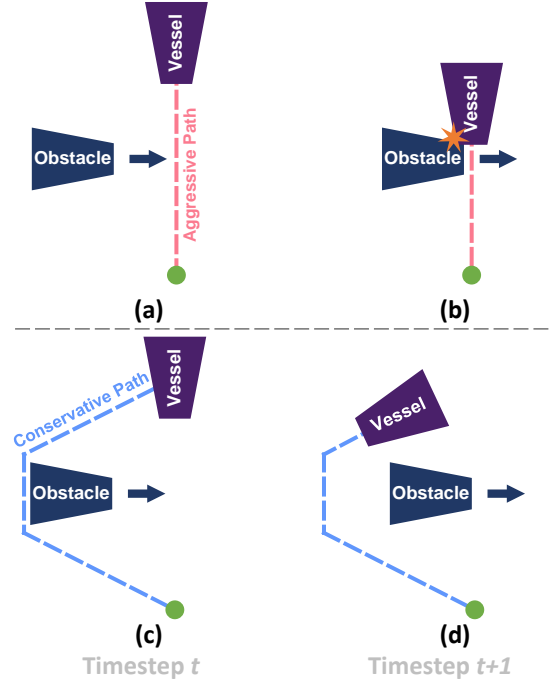


Fig. 4. A comparison of aggressive and conservative path planning.

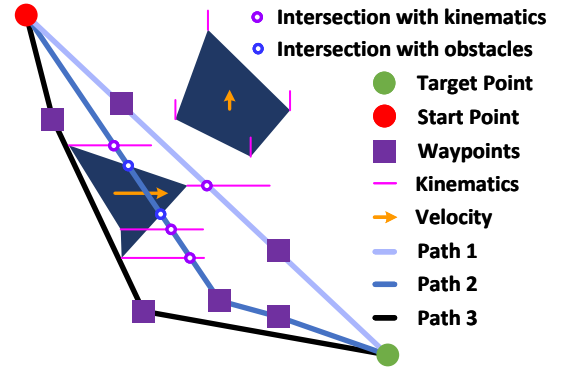


Fig. 5. Obstacle kinematics augmented GPP in dynamic scenarios. Here, Path 2 is deemed the least desirable among the three routes, primarily attributable to its intersections with both obstacles and kinematic segments. Despite being the shortest, Path 1 incurs a penalty owing to its interaction with kinematic segments, thereby diminishing its optimality. Contrastingly, Path 3 emerges as the most rational and secure option. Such configuration effectively obviates aggressive path planning, enhances navigation safety, and still prioritizes the pursuit of a short path. Note that the USV is abstracted as a point, whose volume can be incorporated by dilating the obstacles.

Another issue that warrants consideration is the magnitude alignment between the length term $L(X)$ and penalty terms. Neglecting such alignment can result in fluctuating performance at different navigation stages. An intuitive example is that the distance between the start and target point shrinks at the final navigation stage. This reduction

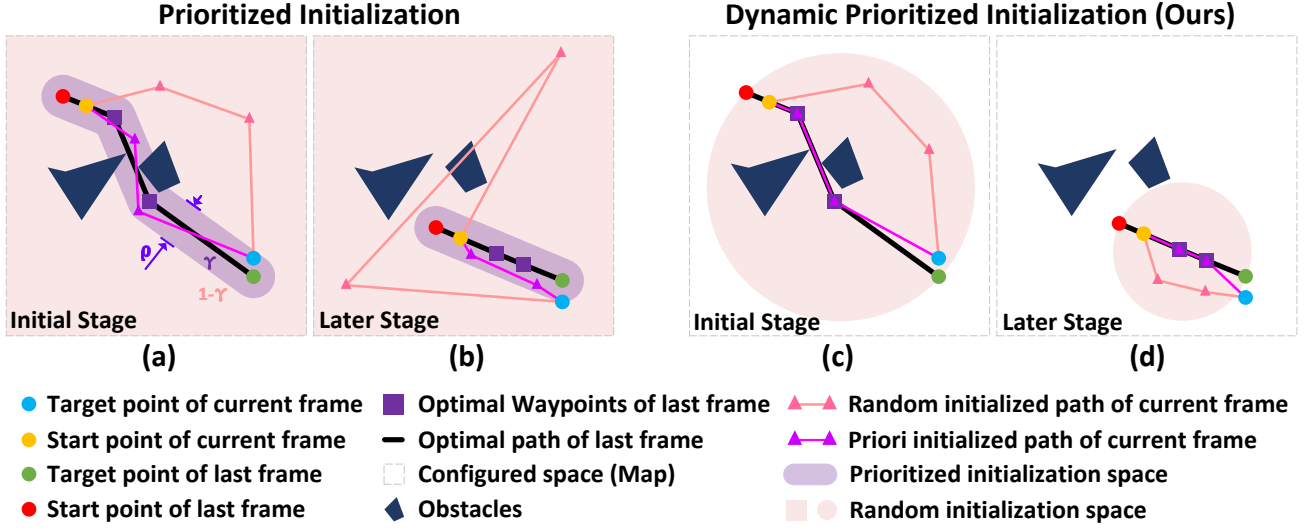


Fig. 6. Comparison between Prioritized Initialization and Dynamic Prioritized Initialization.

renders the length term negligible in comparison to the penalty terms, thereby impairing the planning performance. A solution can be supplementing the penalty terms with a dynamic normalization factor, such that the penalty terms exhibit analogous decreasing characteristics relative to the length term. To this end, the objective function is normalized as follows:

$$F(X) = L(X) + \eta(\alpha \cdot Q(X)^\beta + \mu \cdot P(X)^\nu) \quad (10)$$

where η corresponds to the Euclidean distance of the current start and target point.

3.2. Dynamic prioritized initialization

As introduced in Section 2.3, the PI proposed by Xin et al. (2023a) considerably ameliorates SEPSO's planning quality while expediting its convergence speed. Nevertheless, the PI is beset by two notable issues that detrimentally impact its performance in dynamic scenarios. Firstly, the PI fails to factor the shrinking nature of the navigation, leaving the random initialization space unmanaged throughout the entire navigation process. As revealed by Fig. 6 (b), such configuration is unbefitting at the later stage of navigation, where the unmanaged random initialization space may engender noisy or even irrational initial paths that are less meaningful for the SEPSO to start on. Secondly, the PI has introduced two hyperparameters, γ and ρ , whose determination relies on expert knowledge. Furthermore, these handcrafted hyperparameters also pose a potential threat to the robustness of the planning algorithm in dynamic environments.

To address these issues, the DPI mechanism is proposed, as illustrated by Fig. 6 (c) and (d). In particular, to accommodate the shrinking nature of the navigation, we narrow the random initialization space in accordance with the distance between the start and target points. More concretely, the unprioritized particles are randomly initialized within a

circular area, with its diameter being the distance from the start point to the target point, as illuminated by Fig. 6 (c) and (d). Concerning the fashion of prioritized initialization, our preliminary experiments indicate that inheriting the planned path from the last timestep for a portion of the population suffices to guide the whole swarm. In doing so, random initialization within the PI interval can be discarded, and the two hyperparameters γ and ρ can be omitted. In this context, at the onset of each planning timestep, each group of SEPSO employs one particle to inherit the previously planned path as the guidance for the current planning iteration. The remaining particles are then randomly initialized in the progressively narrowing circular area.

3.3. Autonomous hyperparameter tuning with relaxation strategy

Recalling Section 2.2, during the evolution phase, the H-SEPSO is updated according to Algorithm 1, with its fitness function given by Eq. (8). However, Eq. (8) is inapplicable for navigation tasks in dynamic environments due to its neglect of the collision cases and the environmental stochasticity. Since our pivotal objective is the determination of the search parameter that empowers safe and short navigations, a more comprehensive evaluation criterion for the planned path could be the total travel distance throughout the navigation. In this context, the determination of the search parameter can still be formulated as a minimization problem, with the fitness function of H-SEPSO defined as:

$$S(\theta) = \begin{cases} 0 & \text{if Collide} \\ -1/\delta(\theta, \phi) & \text{if Arrive} \end{cases} \quad (11)$$

where δ is the total travel distance; ϕ is a random variable denoting the stochasticity of the environment.

The presence of the random variable ϕ stems from the dependence of the total travel distance on both the capability of the optimization algorithm and the stochastic nature of

Algorithm 2: Relaxation-based updating principle
for the best values and positions

Input: particle $X_{g,n}^k$ **Output:** the best positions $Pbest_{g,n}^k, Gbest_g^k, Tbest^k$
and the best values $U_{g,n}^{Pbest}, U_g^{Gbest}, U^{Tbest}$ // $f(\cdot)$ is the fitness function; $R(a, b)$ denotes
a uniform random variable between (a, b)

// For minimization problems:

- 1 if $f(X_{g,n}^k) < R(\lambda, 1) \cdot U_{g,n}^{Pbest}$ then $Pbest_{g,n}^k \leftarrow X_{g,n}^k$;
 - 2 if $f(X_{g,n}^k) < U_{g,n}^{Pbest}$ then $U_{g,n}^{Pbest} \leftarrow f(X_{g,n}^k)$;
 - 3 if $f(X_{g,n}^k) < R(\lambda, 1) \cdot U_g^{Gbest}$ then $Gbest_g^k \leftarrow X_{g,n}^k$;
 - 4 if $f(X_{g,n}^k) < U_g^{Gbest}$ then $U_g^{Gbest} \leftarrow f(X_{g,n}^k)$;
 - 5 if $f(X_{g,n}^k) < R(\lambda, 1) \cdot U^{Tbest}$ then $Tbest^k \leftarrow X_{g,n}^k$;
 - 6 if $f(X_{g,n}^k) < U^{Tbest}$ then $U^{Tbest} \leftarrow f(X_{g,n}^k)$;
-

the dynamic environment. That is, even the optimal search parameter, in certain instances, is likely to yield an unfavorable navigation trajectory due to the disturbances induced by dynamic obstacles. Similarly, a suboptimal search parameter could be overestimated when a relatively easy navigation task emerges. Evolving the search parameter θ with Eq. (11) following Algorithm 1 can be problematic, as the overestimation could lead the whole population to exploit or even deadlock within a meaningless search space.

To mitigate the issue discussed above, a relaxation strategy, as given by Algorithm 2, is proposed, wherein the update of the best positions $Pbest_{g,n}^k, Gbest_g^k, Tbest^k$ is subject to a relaxation strategy parameterized by λ . In the context of minimization, for problems with positive fitness value, $\lambda > 1$; whereas for problems with negative fitness value, $1 > \lambda > 0$. This approach allows the update of $Pbest_{g,n}^k, Gbest_g^k, Tbest^k$ without necessitating a stringent decrease in fitness value, enabling a potentially advantageous particle to assist the population in escaping from the deadlock. It is imperative to highlight that the relaxation is not extended to the update of the best values $U_{g,n}^{Pbest}, U_g^{Gbest}, U^{Tbest}$, as doing so would impair the monotonic optimization characteristics the PSO-based techniques.

Finally, to facilitate autonomous tuning of the GPP-associated hyperparameters, $[\alpha, \beta, \mu, \nu, \iota]$ are integrated into Eq. (11) as well. Correspondingly, the ultimate fitness function of H-SEPSO and the assembled hyperparameters are given as below:

$$S(\xi) = \begin{cases} 0 & \text{if Collide} \\ -1/\delta(\xi, \phi) & \text{if Arrive} \end{cases} \quad (12)$$

$$\xi = [\alpha, \beta, \mu, \nu, \iota, \theta_1, \theta_2, \dots, \theta_G] \quad (13)$$

4. Experimental results

In this section, we initially present two simulation environments that underprop our experiments. Subsequently,

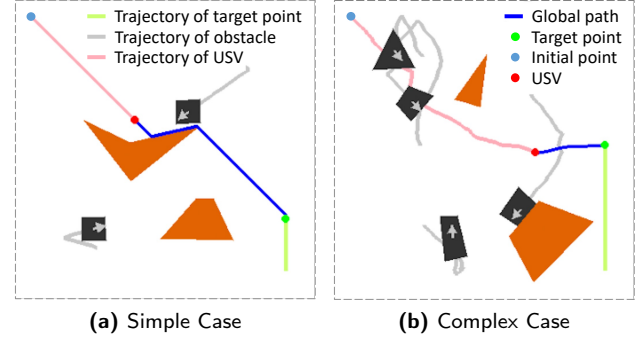


Fig. 7. Simulation environments for navigation. The static obstacles are marked in orange and the dynamic obstacles are marked in black.

Table 1
Hardware Platform.

Component	Description
CPU	Intel Core i9-13900KF
GPU	Nvidia RTX 4090
RAM	64GB 5600MHz
System	Ubuntu 20.04.1

we demonstrate the autonomous hyperparameter tuning with the relaxation strategy. Utilizing the tuned hyperparameters, the OkayPlan is then compared with existing GPP algorithms, where qualitative and quantitative analyses have been made. Finally, ablation studies on the key components of the OkayPlan are conducted to discern their respective contributions. The specifics of the hardware platform employed to conduct these experiments are reported in Table 1.

4.1. Simulation environments

To corroborate the effectiveness of OkayPlan, we have established two simulation environments of size $366m \times 366m$, as the Simple Case and the Complex Case shown in Fig. 7. In the Simple Case, the dynamic obstacles move in a uniform linear manner, which closely aligns with our obstacle kinematics augmented optimization problem explicated in Fig. 5. Meanwhile, to assess the robustness and scalability of our approach when confronted with more realistic circumstances, the Complex Case is instituted, wherein the dynamic obstacles move in a random fashion, rendering them more stochastic and unpredictable. For both cases, the horizontal and vertical velocities of the dynamic obstacles are restricted to $(0, 4) m/s$. To preclude them from exceeding the environments, the dynamic obstacles will be bounced back when reaching the boundaries.

The objective for both cases is to navigate the USV from the initial point to the target point. Generally, the navigation task requires collaboration between GPP and LPP algorithms. Given the paper's emphasis on the GPP algorithm, to eschew the potentially entangled repercussion from the LPP algorithm, we assume that the USV strictly

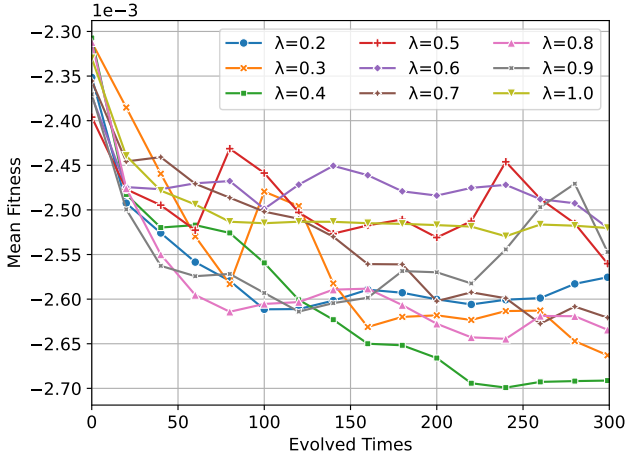


Fig. 8. Evolution curves of H-SEPSO with respect to different relaxation parameter. Here, every curve is averaged over 5 random seeds (0~4) to enhance credibility.

adheres to the global path with a velocity of 6 m/s so as to circumvent the LPP. Additionally, the USV is abstracted as a point for brevity. It is pertinent to note that this abstraction does not compromise the realism of our environments, as the volume of the USV could be factored through the dilation of the obstacles when necessary. To make the environment more general, the target point is dynamically moving as well, with a velocity of 3 m/s along the vertical axis. Considering the constantly changing environment, the GPP algorithm is consecutively executed for each frame, planning from the USV's current position to the target point. As a result, the dynamic environments have posed a significant challenge to the rationality of the GPP algorithm being evaluated, thereby forming a comprehensive testbed to ascertain their individual advantages and disadvantages.

4.2. Determination of hyperparameters

In this section, the relaxation strategy and HSEF is utilized to determine OkayPlan's hyperparameters ξ . Specifically, the decision variable of the L-SEPSO consists of the coordinates of the waypoints as defined in Eq. (1), and its fitness function is expressed by Eq. (10). The updating principle for the best values and positions of the L-SEPSO is delineated by Algorithm 1. On the other hand, the fitness function and the decision variable of H-SEPSO are given by Eq. (12) and Eq. (13), respectively. In order to subdue the overestimation arising from the dynamic environment, Algorithm 2 is applied as the updating principle for the best values and positions of the H-SEPSO.

We have adopted the experimental configurations from Xin et al. (2023a), wherein both L-SEPSO and H-SEPSO are partitioned into 8 groups to uphold population diversity. Each group of L-SEPSO is configured with 170 particles, with each particle represented as a 16-dimensional vector denoting the coordinates of 8 waypoints. In the case of H-SEPSO, each group is assigned with 10 particles. Each particle, namely the hyperparameter of L-SEPSO denoted

Table 2

SEPSO-associated hyperparameters.

Group	ω_{init}	ω_{end}	V_{limit}	C1	C2	C3
1	0.9000	0.9000	0.1000	1.0000	2.0000	1.0000
2	0.2000	0.1000	0.1000	1.4853	1.0000	1.0000
3	0.7434	0.9000	0.1389	1.0000	1.0000	2.0000
4	0.9000	0.9000	0.1000	1.0756	1.0000	1.2968
5	0.2000	0.9000	0.8000	2.0000	2.0000	2.0000
6	0.6094	0.1000	0.1000	1.0000	1.3316	2.0000
7	0.8271	0.1000	0.8000	2.0000	2.0000	1.0000
8	0.9000	0.7743	0.8000	1.9968	1.9253	1.0000

Table 3

GPP-associated hyperparameters.

α	β	μ	ν	ι
4.0000	1.0000	3.9827	6.0000	5.2032

by ξ , is a vector of length 53 (5 GPP parameters + 8 groups \times 6 SEPSO parameters). Note that the search parameter of H-SEPSO employs the recommended configuration from the original paper (Xin et al., 2023a).

Regarding the determination of the relaxation parameter λ , a coarse grid search from 0.2 to 1.0 with a resolution of 0.1 was conducted on the Complex Case. Within each search, the H-SEPSO was evolved for 300 times, and the mean fitness value over the entire population of H-SEPSO was recorded and drawn in Fig. 8. The mean fitness value serves as a metric for the collective cognition of the particles in H-SEPSO. The premature convergence observed in the curve ($\lambda = 1.0$) suggests that, in the absence of the relaxation strategy, the evolution of H-SEPSO is susceptible to entrapment in local optima. In contrast, the green curve ($\lambda = 0.4$) distinctly illustrates the efficacy of the proposed relaxation strategy, indicating that a suitable relaxation parameter can substantially enhance the collective cognition of the population and facilitate the attainment of a superior solution.

The best hyperparameters identified by $\lambda = 0.4$ are reported in Table 2 and 3, which will be employed to conduct the subsequent comparison and ablation studies. Here, the hyperparameters tuning for the Simple Case has been omitted, as our forthcoming experiments will demonstrate that the hyperparameters evolved from the Complex Case can effectively generalize to the Simple Case.

4.3. Comparative experiments

In this section, we seek to ascertain the advantages of the proposed OkayPlan in terms of path safety, length optimality, and computational efficiency versus existing GPP techniques. To this end, we have devised four statistical evaluation criteria to assess the navigation:

- **Fitness:** Defined by Eq.(12), where a lower fitness value signifies the GPP algorithm's capability to generate paths that are both short and safe.

Table 4

Details of global path planning algorithms.

Algorithm	Abbreviation	Type	Reference
OkayPlan	OkayPlan	Metaheuristic-based	This paper
SEPSO-based Global Path Planning	SGPP	Metaheuristic-based	Xin et al. (2023a)
RRT*	RRT*	Sample-based	Karaman and Frazzoli (2011)
Rapidly-exploring random trees	RRT	Sample-based	LaValle (1998)
A*	A*	Search-based	Hart et al. (1968)
Best-first Search	BF	Search-based	Pearl (1984)
Bidirectional A*	BA*	Search-based	Kaindl and Kainz (1997)
Breadth-first Search	BFS	Search-based	Moore (1959)
Dijkstra	Dijkstra	Search-based	Dijkstra (1959)
Minimum Spanning Tree	MST	Search-based	Kruskal (1956) ; Prim (1957)

Table 5Results comparison on **Simple Case**.

Algorithm	Metric	Fitness ($\times 10^{-3}$)	Arrived Travel Distance (<i>m</i>)	Time Per Planning (Sec.)	Arrival Rate
OkayPlan	mean	-2.79	366.18	0.008	1.00
	std.	0.36	62.21	0.001	
SGPP	mean	-2.24	351.08	0.009	0.78
	std.	1.20	33.95	0.001	
RRT*	mean	-2.08	365.37	0.023	0.76
	std.	1.17	5.47	0.012	
RRT	mean	-1.69	495.04	0.010	0.83
	std.	0.78	39.51	0.002	
A*	mean	-2.04	381.54	1.174	0.78
	std.	1.09	4.20	0.262	
BF	mean	-1.90	394.74	0.011	0.75
	std.	1.10	4.01	0.002	
BA*	mean	-1.96	428.41	0.643	0.84
	std.	0.86	5.55	0.171	
BFS	mean	-1.91	391.76	0.248	0.75
	std.	1.11	4.46	0.038	
Dijkstra	mean	-1.95	375.13	3.511	0.73
	std.	1.18	3.59	0.488	
MST	mean	-1.95	375.13	3.532	0.73
	std.	1.18	3.59	0.491	

Table 6Results comparison on **Complex Case**.

Algorithm	Metric	Fitness ($\times 10^{-3}$)	Arrived Travel Distance (<i>m</i>)	Time Per Planning (Sec.)	Arrival Rate
OkayPlan	mean	-2.75	365.10	0.009	1.00
	std.	0.14	21.56	0.001	
SGPP	mean	-1.35	391.73	0.008	0.52
	std.	1.31	60.11	0.002	
RRT*	mean	-1.19	369.00	0.051	0.44
	std.	1.35	9.09	0.014	
RRT	mean	-0.93	518.38	0.018	0.48
	std.	0.97	41.00	0.005	
A*	mean	-1.35	379.12	1.609	0.51
	std.	1.32	3.28	0.696	
BF	mean	-1.22	393.30	0.012	0.48
	std.	1.27	5.24	0.004	
BA*	mean	-1.37	380.75	0.417	0.52
	std.	1.31	6.26	0.132	
BFS	mean	-1.30	384.02	0.274	0.50
	std.	1.30	6.89	0.042	
Dijkstra	mean	-1.20	375.82	4.111	0.45
	std.	1.32	4.08	0.567	
MST	mean	-1.20	375.82	4.037	0.45
	std.	1.32	4.08	0.548	

- **Arrived Travel Distance:** The distance traveled when the USV successfully reaches the target. Note that the travel distance of collision case is discarded. This metric serves as an indicator of the length optimality of the planned paths.
- **Arrival Rate:** The rate at which the USV successfully navigates from the initial point to the target point, offering insights into the safety of the planned paths.
- **Time Per Planning:** The execution time of each planning averaged over the entire navigation, providing an assessment of the computational efficiency of the GPP algorithms.

To facilitate a comprehensive comparison, we have prepared 9 contemporary and canonical GPP algorithms, spanning metaheuristic-based, sample-based, and search-based classifications. The particulars of these algorithms are exhibited in Table 4. Harnessing the hyperparameters specified in Tables 2 and 3, the OkayPlan is compared against these 9 algorithms within both Simple Case and Complex Case. For a rigorous analysis, these experiments are repeated 100 times, ranging from random seed 0 to 99. The averaged results are then presented in Tables 5 and 6.

As listed in Table 5, the results of the Simple Case indicate that the proposed OkayPlan outstrips alternative approaches in terms of Fitness, Time Per Planning, and Arrival Rate, demonstrating the superiority of our method. However, a marginal performance decline in Arrived Travel Distance can also be noticed. We attribute this decline to the conservative planning traits steamed from the kinematic penalty $P(X)$. As exemplified by Paths 1 and 3 in Fig. 5, a safe path may compromise the length optimality. We argue that this trade-off is acceptable and necessary, as it effectively prevents the USV from colliding with dynamic obstacles and prominently elevates the success rate of navigation. Regarding the Complex Case, Table 6 reveals a more distinct performance gap between OkayPlan and the remaining 9 approaches, where the performance of other GPP algorithms deteriorates dramatically due to the intricacies of the environment. In contrast, the OkayPlan consistently yields commendable results, showcasing its robustness when facing complex environments with dynamic or even unpredictable obstacles.

In both Simple and Complex Cases, OkayPlan exhibits competitive computational efficiency when compared with its predecessor, SGPP. This observation suggests that the proposed obstacle kinematics augmented optimization problem and DPI have been seamlessly integrated without imposing additional computational burdens. Impressively, OkayPlan attains a planning frequency of 125 Hz (0.008 s) on a desktop-class computer, validating its capability to efficiently address real-time planning tasks in dynamic environments.

To enhance readers' comprehension of the behaviors exhibited by different GPP algorithms, we have supplemented the snapshots of their respective navigation processes, as

the Fig. 9 and Fig. 10 presented in Appendix. OkayPlan's safer planning strategy can be clearly observed at timestep 10 and 15 in Fig. 9. Leveraging obstacle kinematics, OkayPlan can circumvent path contention with dynamic obstacles by opting for a more conservative path. This contrasts with other GPP algorithms, which yield more aggressive paths, thereby increasing the potential for collisions and compromising navigation safety.

The above findings corroborate the preeminence of OkayPlan in generating short and safe paths with prompt planning speed in dynamic environments. These attributes are particularly beneficial for USVs. Firstly, the superiority in path length is advantageous, given USVs' limited on-board energy resources. Meanwhile, considering the delayed maneuverability of USVs and the dynamic nature of their operating environments, the capacity to consider the motion of other obstacles and make timely adjustments to the planned path can significantly bolster the navigation safety.

4.4. Ablation studies

In this section, we conduct ablation studies on OkayPlan by removing each of its key components to explicate their individual contributions. The experimental setups for these ablation studies are as follows:

- **No Obstacle Kinematics** (Section 3.1): μ in Eq. (10) is set to be 0.
- **No Dynamic Normalization** (Section 3.1): η in Eq. (10) is set to be the Euclidean distance between the initial point and target point at the beginning of the navigation and remains constant though out the navigation.
- **No DPI** (Section 3.2): the DPI mechanism is degenerated into the PI mechanism (Xin et al., 2023a) as illustrated in Fig. 6 (a) and (b).
- **No Relaxation Strategy** (Section 3.3): OkayPlan employs the hyperparameter ξ evolved with relaxation parameter $\lambda = 1$.

The ablation studies are performed on both the Simple Case and Complex Case. Each of them is repeated for 100 times with random seeds ranging from 1 to 99, and the averaged results are reported in Tables 7 and 8. The results suggest that the four components under investigation exhibit no explicit impact on computational efficiency, mirroring the compact and integral design of OkayPlan. Omitting the Obstacle Kinematics results in a significant decrease in Arrival Rate, revealing its remarkable contribution to the safety of the planned paths. Meanwhile, as discussed in the preceding section, a trade-off between path safety and length optimality is observed when ablating the Obstacle Kinematics, which validates our previous conjecture. While the Dynamic Normalization contributes marginally to OkayPlan, its incorporation is still considered beneficial due to its negligible effect on the computational overhead. The DPI and the Relaxation Strategy manifest similar repercussions,

Table 7
Ablation Study on **Simple Case**.

Algorithm	Metric	Fitness ($\times 10^{-3}$)	Arrived Travel Distance (m)	Time Per Planning (Sec.)	Arrival Rate
OkayPath	mean	-2.79	366.18	0.008	1.00
	std.	0.36	62.21	0.001	
No Obstacle Kinematics	mean	-2.35	358.55	0.009	0.83
	std.	1.10	51.42	0.002	
No Dynamic Normalization	mean	-2.78	369.54	0.009	1.00
	std.	0.37	70.31	0.001	
No DPI	mean	-2.72	371.58	0.010	1.00
	std.	0.24	40.92	0.001	
No Relaxation Strategy	mean	-2.77	370.68	0.009	1.00
	std.	0.37	74.11	0.001	

Table 8
Ablation Study on **Complex Case**.

Algorithm	Metric	Fitness ($\times 10^{-3}$)	Arrived Travel Distance (m)	Time Per Planning (Sec.)	Arrival Rate
OkayPath	mean	-2.75	365.10	0.009	1.00
	std.	0.14	21.56	0.001	
No Obstacle Kinematics	mean	-1.58	349.09	0.010	0.55
	std.	1.43	6.78	0.002	
No Dynamic Normalization	mean	-2.71	367.70	0.010	0.99
	std.	0.32	26.66	0.001	
No DPI	mean	-2.18	464.72	0.010	0.97
	std.	0.57	105.53	0.001	
No Relaxation Strategy	mean	-2.59	387.82	0.009	0.99
	std.	0.39	56.15	0.001	

where their exclusion leads to a loss of length optimality, as indicated by the Fitness and Arrived Travel Distance.

5. Conclusions

This paper introduced the OkayPlan, aiming to address the global path planning problems in dynamic environments for USVs. Three techniques were proposed to tackle the dynamics of the environments from different aspects. Extensive experiments, encompassing comparative experiments and ablation studies, were conducted in two simulation environments with escalating difficulties. The results show that OkayPlan generates safer, shorter paths with a faster planning speed (125 Hz). Based on these impressive results, we contend our works successfully extend GPP algorithms to dynamic scenarios and could serve as a benchmark for future research. In addition, we have open-sourced our code and simulation environments to facilitate future development and comparisons. Promising future research directions include extending OkayPlan to multi-agent planning scenarios and exploring its applications in real-world settings.

Acknowledgement

We acknowledge the support from the National Natural Science Foundation of China under Grant No. 62273230.

CRedit authorship contribution statement

Jinghao Xin: Conceptualization, Methodology, Programming, Investigation, Writing: original draft. **Jinwoo Kim:** Resources, Supervision, Writing: review and editing. **Shengjia Chu:** Language and Data Checking, Writing: review and editing. **Ning Li:** Supervision, Funding acquisition, Writing: review and editing.

Appendix

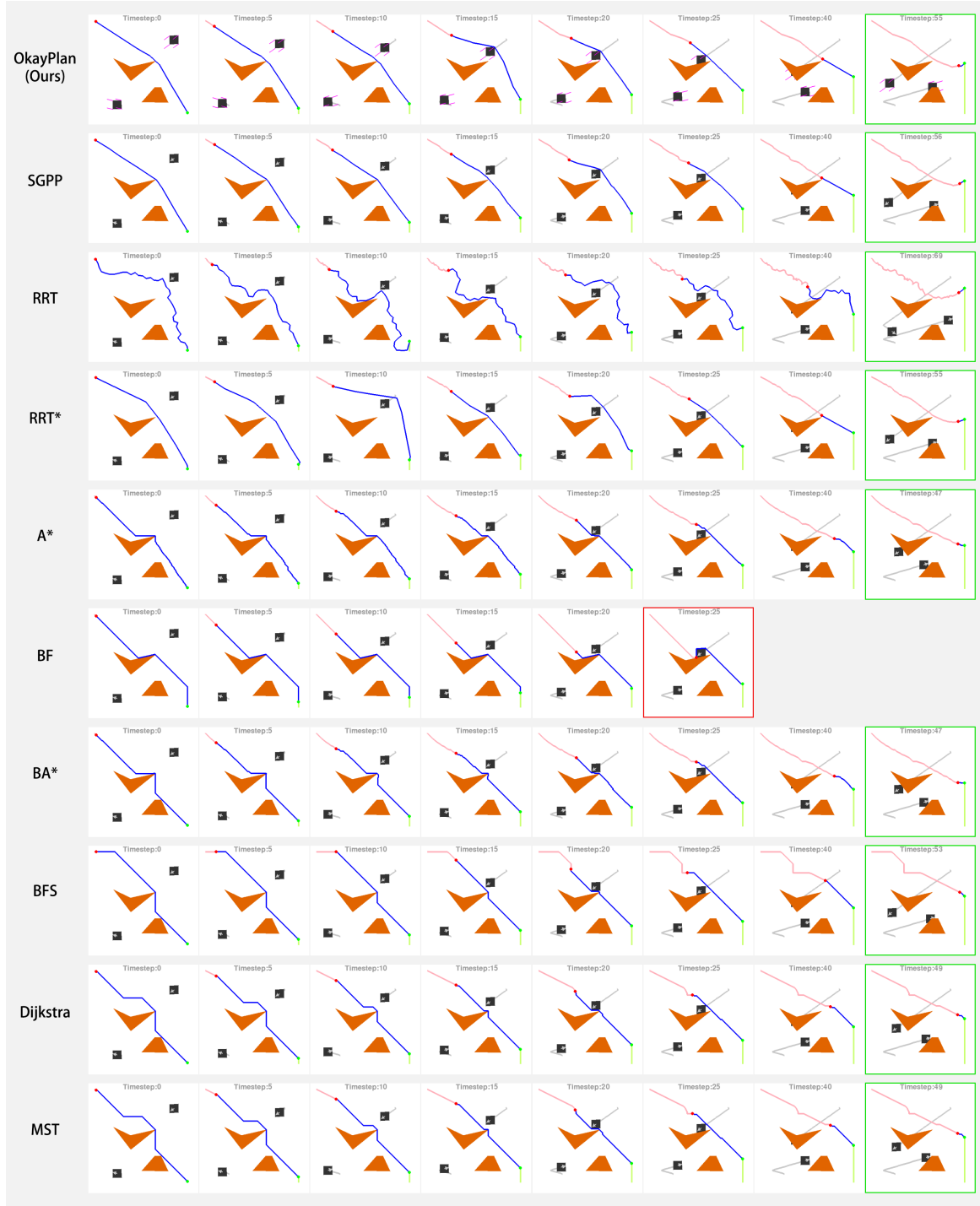


Fig. 9. Behaviors comparison between different GPP algorithms on **Simple Case**. Each row represents the planning results of a specific GPP algorithm at different timesteps. The temporal sequence progresses from left to right. On the rightmost side, the algorithm that successfully reaches the target point is marked with a green box, while algorithms that experience collisions are marked with a red box. OkayPlan's conservative planning strategy to avoid contention with the obstacles can be observed at timestep 15, where other algorithms choose aggressive planning, risking their path safety.

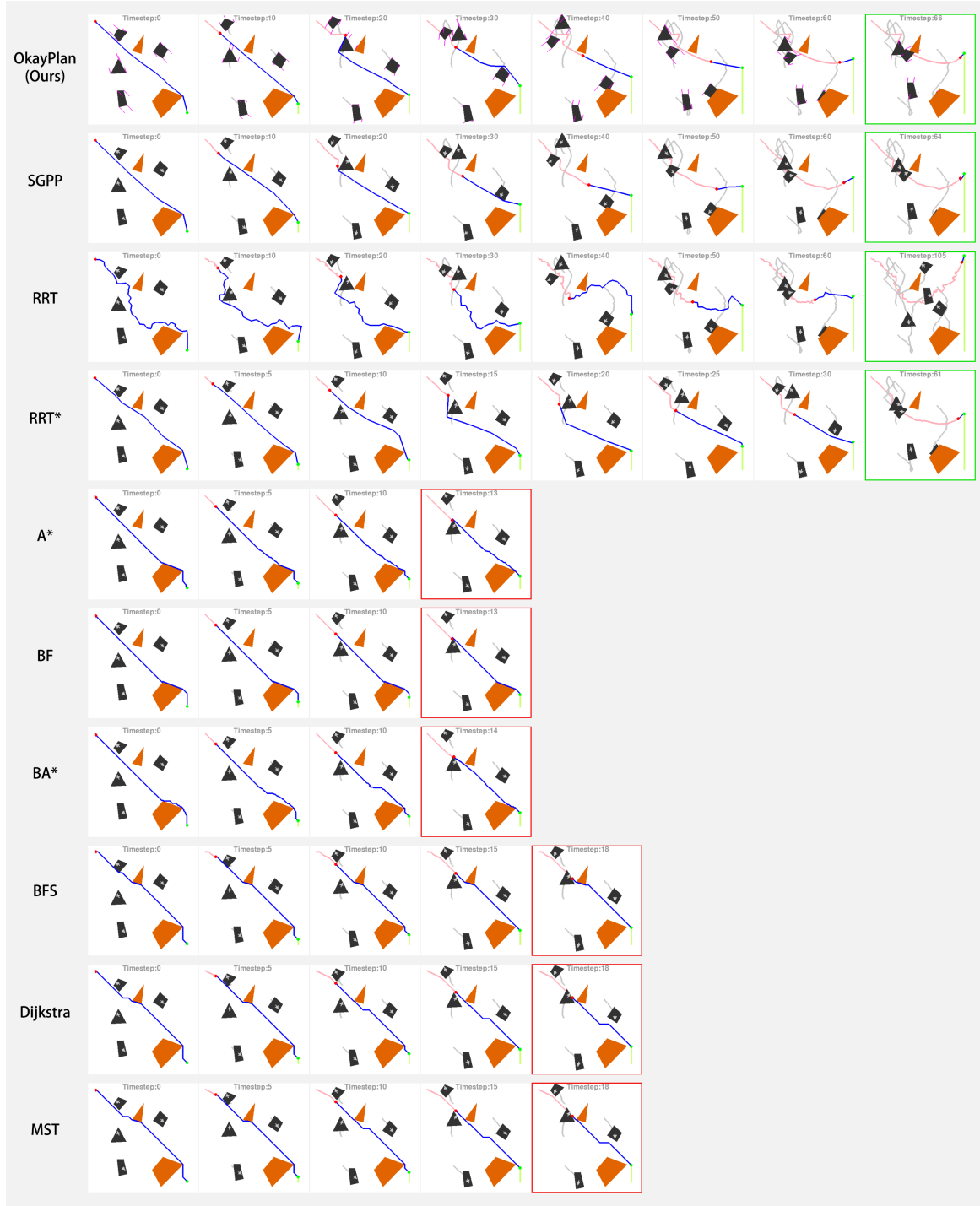


Fig. 10. Behaviors comparison between different GPP algorithms on **Complex Case**. Each row of snapshots represents the planning results of a specific GPP algorithm at different timesteps. The temporal sequence progresses from left to right. On the rightmost side, the algorithm that successfully reaches the target point is marked with a green box, while algorithms that experience collisions are marked with a red box.

References

- Dijkstra, E., 1959. A note on two problems in connexion with graphs. *Numerische Mathematik* 1, 269–271.
- Guo, X., Ji, M., Zhao, Z., Wen, D., Zhang, W., 2020. Global path planning and multi-objective path control for unmanned surface vehicle based on modified particle swarm optimization (psa) algorithm. *Ocean Engineering* 216, 107693.
- Hart, P.E., Nilsson, N.J., Raphael, B., 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics* 4, 100–107. doi:[10.1109/TSSC.1968.300136](https://doi.org/10.1109/TSSC.1968.300136).
- Holland, J.H., 1992. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press.
- Hu, H., Wang, Q., Cheng, M., Gao, Z., 2023. Trajectory prediction neural network and model interpretation based on temporal pattern attention. *IEEE Transactions on Intelligent Transportation Systems* 24, 2746–2759. doi:[10.1109/TITS.2022.3219874](https://doi.org/10.1109/TITS.2022.3219874).
- Jin, J., Zhang, J., Shao, F., Lyu, Z., Wang, D., 2018. A novel ocean bathymetry technology based on an unmanned surface vehicle. *Acta Oceanologica Sinica* 37, 99–106.
- Kaindl, H., Kainz, G., 1997. Bidirectional heuristic search reconsidered. *Journal of Artificial Intelligence Research* 7, 283–317.
- Karaman, S., Frazzoli, E., 2011. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research* 30, 846–894.
- Kavraki, L.E., Svestka, P., Latombe, J.C., Overmars, M.H., 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation* 12, 566–580.
- Kennedy, J., Eberhart, R., 1995. Particle swarm optimization, in: *Proceedings of ICNN'95-international conference on neural networks*, IEEE. pp. 1942–1948.
- Kim, S., Jeon, H., Choi, J.W., Kum, D., 2023. Diverse multiple trajectory prediction using a two-stage prediction network trained with lane loss. *IEEE Robotics and Automation Letters* 8, 2038–2045. doi:[10.1109/LRA.2022.3231525](https://doi.org/10.1109/LRA.2022.3231525).
- Kruskal, J.B., 1956. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society* 7, 48–50.
- LaValle, S., 1998. *Rapidly-exploring random trees: A new tool for path planning*. Research Report 9811.
- Li, D., Wang, P., Du, L., 2018. Path planning technologies for autonomous underwater vehicles-a review. *Ieee Access* 7, 9745–9768.
- Li, X., Yu, S., 2023. Three-dimensional path planning for auvs in ocean currents environment based on an improved compression factor particle swarm optimization algorithm. *Ocean Engineering* 280, 114610.
- Liu, Z., Zhang, Y., Yu, X., Yuan, C., 2016. Unmanned surface vehicles: An overview of developments and challenges. *Annual Reviews in Control* 41, 71–93.
- Moore, E.F., 1959. The shortest path through a maze, in: *Proc. of the International Symposium on the Theory of Switching*, Harvard University Press. pp. 285–292.
- Pearl, J., 1984. *Heuristics: intelligent search strategies for computer problem solving*. Addison-Wesley Longman Publishing Co., Inc.
- Prim, R.C., 1957. Shortest connection networks and some generalizations. *The Bell System Technical Journal* 36, 1389–1401.
- Schofield, R.T., Wilde, G.A., Murphy, R.R., 2018. Potential field implementation for move-to-victim behavior for a lifeguard assistant unmanned surface vehicle, in: *2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pp. 1–2. doi:[10.1109/SSRR.2018.8468602](https://doi.org/10.1109/SSRR.2018.8468602).
- Xin, J., Li, Z., Zhang, Y., Li, N., 2023a. Efficient real-time path planning with self-evolving particle swarm optimization in dynamic scenarios. *Unmanned Systems* doi:[10.1142/S230138502441005X](https://doi.org/10.1142/S230138502441005X).
- Xin, J., Yu, L., Wang, J., Li, N., 2023b. A diversity-based parallel particle swarm optimization for nonconvex economic dispatch problem. *Transactions of the Institute of Measurement and Control* 45, 452–465.