applied sciences

MDPI

Article

Underwater Vehicle Path Planning Based on Bidirectional Path and Cached Random Tree Star Algorithm

Jinxiong Gao, Xu Geng, Yonghui Zhang * and Jingbo Wang

School of Information and Communication Engineering, Hainan University, Haikou 570100, China; 20211108100000007@hainanu.edu.cn (J.G.); xgeng@hainanu.edu.cn (X.G.); 21220854000176@hainanu.edu.cn (J.W.)
* Correspondence: yhzhang@hainanu.edu.cn

Abstract: Underwater autonomous path planning is a critical component of intelligent underwater vehicle system design, especially for maritime conservation and monitoring missions. Effective path planning for these robots necessitates considering various constraints related to robot kinematics, optimization objectives, and other pertinent factors. Sample-based strategies have successfully tackled this problem, particularly the rapidly exploring random tree star (RRT*) algorithm. However, conventional path-searching algorithms may face challenges in the marine environment due to unique terrain undulations, sparse and unpredictable obstacles, and inconsistent results across multiple planning iterations. To address these issues, we propose a new approach specifically tailored to the distinct features of the marine environment for navigation path planning of underwater vehicles, named bidirectional cached rapidly exploring random tree star (BCRRT*). By incorporating bidirectional path planning and caching algorithms on top of the RRT*, the search process can be expedited, and an efficient path connection can be achieved. When encountering new obstacles, ineffective portions of the cached path can be efficiently modified and severed, thus minimizing the computational workload while enhancing the algorithm's adaptability. A certain number of simulation experiments were conducted, demonstrating that our proposed method outperformed cutting-edge techniques like the RRT* in several critical metrics such as the density of path nodes, planning time, and dynamic adaptability.

Keywords: autonomous path planning; rapidly exploring random tree star; path cache; two-way extended random tree; underwater vehicle

Citation: Gao, J.; Geng, X.; Zhang, Y.; Wang, J. Underwater Vehicle Path Planning Based on Bidirectional Path and Cached Random Tree Star Algorithm. *Appl. Sci.* **2024**, *14*, 947. https://doi.org/10.3390/app14020947

Academic Editor: Yutaka Ishibashi

Received: 15 December 2023
Revised: 17 January 2024
Accepted: 19 January 2024
Published: 22 January 2024

1. Introduction

Basic monitoring of the marine environment is essential for early warning and assessment of marine hydro-meteorological conditions, urban climate change, and ecosystem disasters [1,2]. In the present age, we are facing a range of ocean changes that threaten the sustainable development of the human race, such as rising seawater temperatures [3,4], increased acidity in the oceans [5], oxygen reduction [6], coral bleaching [7], declining numbers of sea creatures [8], and various forms of pollution [9], including plastic waste. Over recent decades, growths in ocean farming, mineral extraction, and shipping and tourism, as well as increases in anthropogenic pollution sources (including plastic and chemical pollutants), have led to escalating levels of pollution. By 2030, the global population is expected to reach 8.5 billion, with 40% of this figure residing within 100 km of the coastline, thereby precipitating an increase in demand for marine resources and capacity [10]. Concurrently, as terrestrial resources continue to diminish, it is becoming increasingly urgent to ensure that oceans are protected by formulating measures for the utilization and conservation of marine resources [1]. The core mission of marine conservation consists of observing, monitoring, and understanding the marine environment, which has recently garnered significant global attention in the field of marine science [11,12].

In order to achieve large-scale monitoring and data collection in marine environments at a reduced cost, there are currently two primary approaches: utilizing remote

sensing technology and autonomous underwater vehicles. The collection of data through remote sensing technology is associated with certain drawbacks, including limited temporal coverage, uncertainties in inversion models, and susceptibility to weather conditions. Additionally, satellites have a notable limitation in their inability to acquire information from the seafloor. The other platform is autonomous underwater vehicles (AUVs), which exhibit higher maneuverability in the marine environment [1,13]. As such, they have seen widespread use in marine monitoring and conservation tasks. The adaptability of underwater robots for environmental monitoring allows them to quickly acquire high spatial resolution data, enabling their real-time monitoring of fluctuations in specific oceanic parameters and risks. This has helped to supplement the shortcomings of other platforms [14]. Automatic path planning is a crucial functionality of underwater robots. However, the complex and dynamic nature of marine environments poses challenges distinct from those encountered in terrestrial settings [15,16]. Therefore, this paper places particular emphasis on investigating path-planning algorithms specifically tailored for application in underwater domains.

Generally, the path planning for autonomous underwater vehicle navigation differs from that of land and presents typical challenges in 3D space [17]. Numerous techniques have been proposed for optimizing 3D paths, including statistical optimization-based strategies, for example, the A-star algorithm [18], potential-based methods [19], ant colony algorithms, and genetic algorithms. Despite their demonstrated capability, the majority of these approaches suffer from intrinsic intractability due to their NP-hardness, posing significant computational challenges for optimization [20].

To expedite the search process, an algorithm based on random sampling has been developed, including probabilistic roadmaps (PRMs) [21] and rapidly exploring random trees (RRTs). Sample-based methods efficiently optimize path solutions by eliminating the explicit delineation of impediments [22]. PRMs represent a sparse configuration space of vehicles using a graph model where waypoints serve as nodes, and graph edge reduction is utilized to optimize the path by minimizing the length of the connections between nodes [23]. Conversely, the RRT algorithm is better suited for non-linear dynamic scenes, but it does not guarantee the optimal path [24]. Multiple enhancements have been suggested for the fundamental RRT framework, such as lazy-RRT, RRT-connect, and fixed-node RRT (FN-RRT) [25]. Nonetheless, nearly all RRT-based approaches merely produce non-optimal path solutions. To address this issue, the RRT* algorithm is a variant of the original method, which updates neighboring node routes by adding new nodes to the tree. This approach ensures asymptotic optimization results and is more likely to achieve optimal outcomes [26]. Nevertheless, the RRT* approach exhibits high computational complexity and low efficiency, making it challenging to apply in underwater environments. In recent years, reinforcement learning and deep reinforcement learning methods have been gradually applied to underwater vehicle path planning, but they require a certain amount of computing power [27–29].

Due to the distinctive terrain undulations, relatively sparse obstacles, and the heightened likelihood of novel impediments inherent in marine environments, the aforementioned methods may prove unsuitable for underwater conditions. Path-planning methodologies for underwater environments should be swifter and more adaptive. In response to this challenge, this paper proposes an enhanced RRT* algorithm tailored for underwater path planning. This algorithm expedites the search process and bolsters the adaptability of the algorithm to dynamics.

- This paper presents a novel approach to path planning, leveraging the principles of bidirectional path planning and cached RRT* algorithm. To expedite the search process, we employ a bidirectional search strategy using extended random trees, facilitating the swift identification of both start and target points as well as efficient path connectivity.
- In our proposed methodology, the path computed during the preceding step serves as a cached representation of the optimal route. Subsequently, when novel obsta-

cles emerge, we efficiently modify and sever the invalid portions of the cached path, thereby minimizing computational efforts while simultaneously enhancing the dynamic adaptability of the algorithm.

- A certain amount of simulation experiments have been carried out. This reflects that my proposed method is currently the best method, such as the fast exploration random tree star algorithm, in several critical metrics including path node density, planning time, and dynamic adaptability.

2. Related Works

As previously discussed, the optimal kinodynamic motion planning using incremental sampling-based methods (RRT*) [30] algorithm has emerged as the leading approach for path planning in robotics, owing to its superior performance and other state-of-the-art methodologies. Consequently, recent research endeavors have focused on leveraging the RRT* methodology to expedite the convergence and search processes. Our research is in line with pertinent studies utilizing the RRT* and its derivatives. Figure 1 illustrates the structure of the RRT* algorithm.

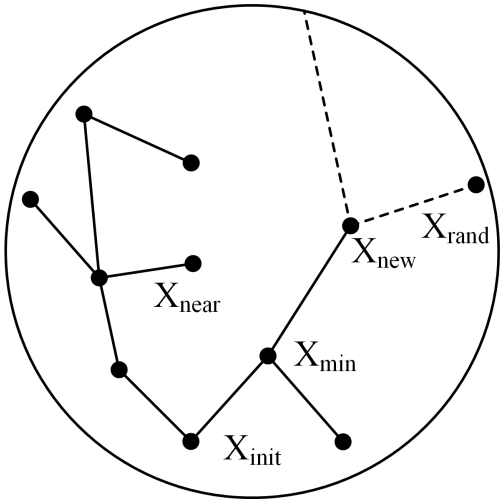

Figure 1. Schematic diagram of optimal kinodynamic motion planning using incremental sampling-based methods (RRT*) [30] . The meaning of X_{init} is initial position. The meaning of X_{min} is the point closest to the initial position. The meaning of X_{near} is the point adjacent to the initial position. The meaning of X_{new} is the new location node explored according to the algorithm. The meaning of X_{rand} is that there is a probability of the most new node.

In the pursuit of enhancing rapidly exploring random tree star (RRT*) performance, two fundamental methods have been proposed. The first approach concentrates on refining the sampling process, while the second involves path optimization strategies. Qureshi and Ayaz proposed an artificial potential field approach using the potential function to guide samples towards the local minima, which correspond to the optimal path [31]. The merits of this method are that it efficiently reduces sample dispersion and accelerates convergence. Kiesel et al. introduced an RRT* heuristic function that learns estimators online to guide motion tree expansion for path optimization [32]. This learning approach generates available guidance, enabling efficient exploration of the optimum solution. Akgun et al. employed a node-rejection criterion to enhance computational efficiency, where the number of iterations for a single state is affected by the sampling domain [33]. Noreen and Khan conducted collaborative research into keypoint suppression and bounded sampling techniques by applying the RRT*-adjustable boundary (RRT*-AB) method [34]. Graph clipping is a strategy that reduces the model size by utilizing heuristic functions. It retains samples that optimize existing solutions while eliminating others during the exploration process. Karaman and Frazzoli present a computational and asymptotically optimal approach to the algorithm, generating RRT* and PRM*, significantly improving sampling-based methods' efficiency.

However, these approaches might exaggerate the heuristic expenses and result in inaccuracies when utilizing graph compression techniques, primarily occurring at the search tree's beginning [35]. By merging RRT-connect and RRT* approaches, Klemm et al. devised a proficient randomized motion planning algorithm that attains a theoretical optimum more rapidly than its RRT* counterpart [36]. Tahir et al. proposed a novel path-planning algorithm consisting of potentially guided intelligent bidirectional RRT* (PIB-RRT*) and potentially guided bidirectional RRT*PB-RRT* [37]. Shen et al. proposed a fast path-planning method for underwater robots that improves RRT* by combining target and search algorithms to realize quick path planning [38]. The aforementioned algorithm mentioned above has achieved remarkable success in various domains. However, due to the sparse visual spatial distribution of submerged impediments, the elevated likelihood of arbitrary emergence of novel impediments, the fluctuating submerged topography, and other factors, there is a requirement for more efficient path-planning algorithms that connect the initial and objective points. Liu et al. [39] devised a virtual force method. The idea of the method is to simulate the environment as an electrostatic field and a real-world fluid to specify the effects of targets and obstacles on the robot, and then plan the channel, but it is easy to fall into the local minima. Das et al. [40] devised the bug algorithm to make the robot move in a straight line towards the goal when it does not encounter an obstacle; otherwise, it goes around the boundary of the obstacle. Although the algorithm is computationally simple, it can be extreme, i.e., if there are a large number of dynamic obstacles, the robot will keep walking around the obstacles and will not even be able to reach the goal. Yang et al. [41] proposed an N-step priority dual DQN (NPDDQN) path-planning algorithm, which effectively achieves obstacle avoidance in complex environments, and designed a filtering mechanism that improves the utilization of a priori knowledge, but still suffers from the drawbacks of more complex computation and reliance on a priori knowledge.

The existing organizational structure of this investigation is outlined in the following manner: Section 2 offers an evaluation of relevant literature, while contingent details and mechanics of the proposed methodology are elucidated in Section 3. Subsequently, experimental evaluation and consequential analysis are presented in Section 4. Section 5 provides a summary of this work along with a discourse on future research trajectories.

3. Bidirectional Cached Rapidly Exploring Random Tree Star

This section will provide a detailed exposition of the designed bidirectional path and cached RRT* algorithm.

3.1. Random Tree Search Strategy of Bidirectionally Extended RRT Algorithm*

During path planning of autonomous underwater vehicles in the ocean, optimization of the planning process can ameliorate both the efficiency and caliber of the resultant path. However, because of the uniform sampling in the basic RRT* algorithm, numerous redundant branches will be generated on the search tree, which are typically planned. Consequently, the paths are suboptimal and comprise many noisy nodes, leading to insufficient smoothness. To tackle this issue, redundant path nodes are smoothed by filtering. The smoothing operation starts from the second node and checks backwardly to determine whether the connection between the starting node and the current node interferes with obstacles. If interference happens, the node is removed; otherwise, it is retained. Following the anomaly detection process, the superfluous nodes along the path can be expeditiously eliminated. Therefore, this study proposes an improved RRT*-based approach for completing underwater path planning. By introducing the Dijkstra algorithm for subsampling the extended random tree and searching for the set matrix of nodes, the exploratory bidirectional tree search methodology is employed to establish an initial pathway. If the cost of a path exceeds a critical value, the pathway is refined and selected as the optimal solution, leading to an update of the initial path to its optimized counterpart. This approach not only improves the search efficiency but also guarantees the quality of the planned path.

The comprehensive workflow of subaquatic trajectory planning utilizing the refined RRT* algorithm is depicted in Figure 2.

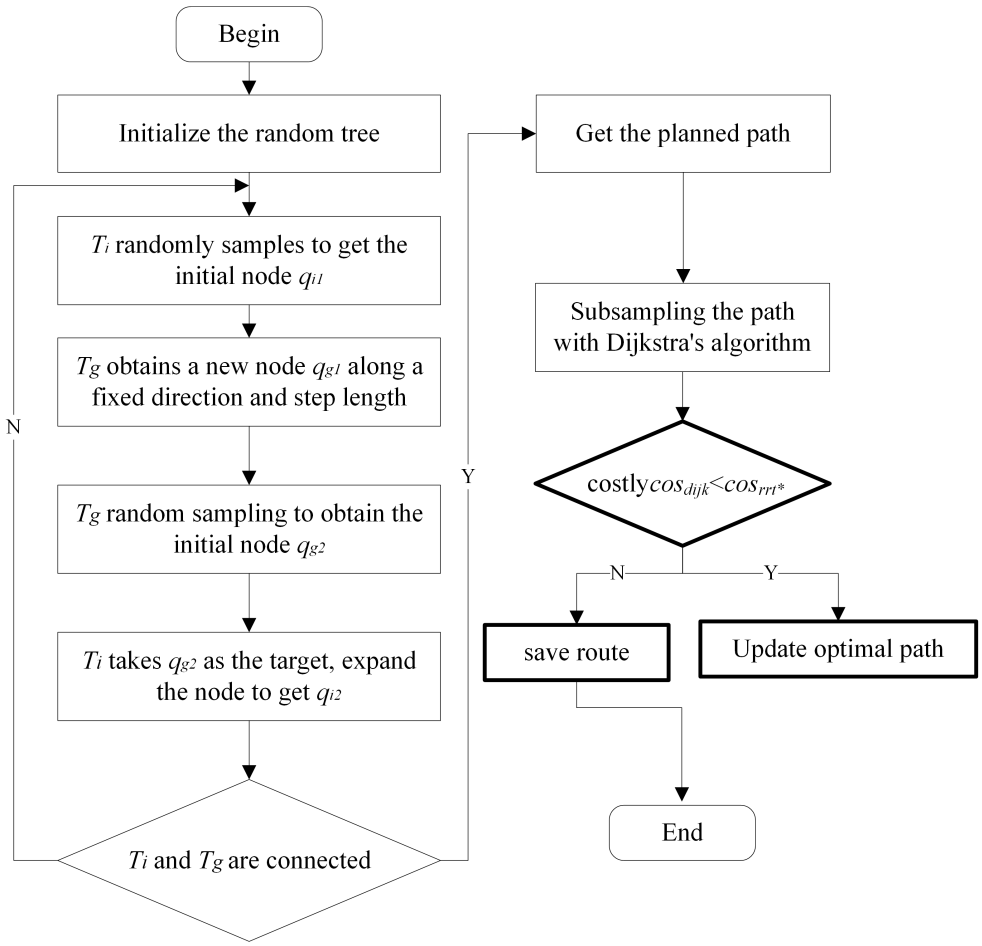

Figure 2. Schematic diagram of two-way extended RRT* algorithm. T_i and T_g mean two random trees. q_i and q_g represent the node of random sampling expansion.

This study aims to address the issue of underwater robot path planning by efficiently connecting the start and target points through a short path. Unlike the basic RRT* algorithm, which relies on random sampling from the initial node to expand search space, this paper proposes the simultaneous growth of fast-expanding random trees from both the start and target points to improve search efficiency and quickly find an initial cable planning path. The parallel search enhances solution efficiency while maintaining a shorter initial path through alternating attraction between the two trees. By employing the dual-aspect enhanced random tree exploration methodology suggested in this study, prompt acquisition of an initial trajectory linking the origin and destination is achievable, to which lies further node-level fine-tuning for optimal path inference.

3.2. Path-Caching Strategy

In the intricate and dynamic milieu of the marine realm, dynamic or novel impediments may necessitate adept modifications to the random tree of the RRT* algorithm. A possible strategy is to promptly re-plan and abandon the original path when encountering previously unknown obstructions. The new route should evade current obstacles while maintaining the continuity of the original trajectory. To realize this continuity bias, a route cache must be established in the planning process. Path caching involves pre-saving nodes on the static plan's original path. In the novel path-planning process, these points have the opportunity to guide the growth of newly generated nodes by serving as reference

points. Subsequently, under consideration of the impact of gravitational offset on the target position, the path cache functions as a directional tool guiding the growth of the random tree. An improved growth function $F(x_{near})$ is proposed for generating new nodes from x_{near}:

$$F(x_{near}) = R(x_{near}) + G(x_{near}) + P(x_{near}) \tag{1}$$

where $R(x_{near})$ represents the stochastic sampling operation performed on the nodes, $G(x_{near})$ stands for the intended directional tendency function, and $P(x_{near})$ signifies the cached path bias function. Furthermore, the gravitational influence exerted by the target vector x_{goal} on x_{near} can be mathematically formulated as:

$$G = k_g \cdot \left\| x_{goal} - x_{near} \right\| \tag{2}$$

The vector x_{goal} denotes the target point's position. The Euclidean norm of the difference between x_{goal} and x_{near}, denoted by $\left\| x_{goal} - x_{near} \right\|$, indicates the distance from the node to the target point. Furthermore, ρ signifies the search step length, while k_p stands for the gravitational coefficient. Consequently, the ensuing equation can be deduced as the objective bias function:

$$G(x_{near}) = \rho \cdot k_g \frac{x_{goal} - x_{near}}{\left\| x_{goal} - x_{near} \right\|} \tag{3}$$

Analogously, the offset function for path caching can be formulated as follows:

$$P(x_{near}) = \rho \cdot k_p \frac{x_{path,i} - x_{near}}{\left\| x_{path,i} - x_{near} \right\|} \tag{4}$$

where k_p denotes the coefficient for path cache offset, and $x_{path,i}$ signifies the positional vector of the i-th node within the pathway directory. Specify the Euclidean distance value between two points (fixed point and node) is denoted by $\left\| x_{path,i} - x_{near} \right\|$ in the equation. The format of the random expansion function in the fundamental RRT* algorithm is represented as follows:

$$R(x_{near}) = \rho \cdot \frac{x_{rand} - x_{near}}{\| x_{rand} - x_{near} \|} \tag{5}$$

By substituting Formulas (3)–(5) into Equation (1), a resultant expression is obtained:

$$F(x_{near}) = \rho \cdot \frac{x_{rand} - x_{near}}{\| x_{rand} - x_{near} \|} + \rho \cdot k_g \frac{x_{goal} - x_{near}}{\left\| x_{goal} - x_{near} \right\|} + \rho \cdot k_p \frac{x_{path,i} - x_{near}}{\left\| x_{path,i} - x_{near} \right\|} \tag{6}$$

The algorithmic expression for generating the novel entity is additionally derived as follows:

$$x_{new} = x_{near} + \rho \cdot \frac{x_{rand} - x_{near}}{\| x_{rand} - x_{near} \|} + \rho \cdot k_g \frac{x_{goal} - x_{near}}{\left\| x_{goal} - x_{near} \right\|} + \rho \cdot k_p \frac{x_{path,i} - x_{near}}{\left\| x_{path,i} - x_{near} \right\|} \tag{7}$$

During implementation, three distinct probabilities are initially defined as p_{goal}, p_{path}, and p_{rand}. p_{goal} represents the probability of selecting the target point (5), (6), or (7) as the new node to guide the growth of the random tree towards the goal. On the other hand, p_{path} corresponds to the probability of choosing a path cache point as a new node, while p_{rand} denotes the probability of selecting a new node at random.

$$p_{goal} + p_{path} + p_{rand} = l \tag{8}$$

The determination of p_{goal} and p_{path} represents a pivotal aspect among the three probabilities, wherein comprehensive simulation experiments are imperative to ascertain based on distinct contextual contingencies. In order to debug, $p_{goal} = 0.3$ and $p_{path} = 0.6$ serve as the preliminary parameters.

This section primarily concerns the enhancement of the node selection function utilized in the RRT* algorithm for selecting the new node, which involves a stochastic process. Specifically, the aforementioned operation entails the selection of an integer value obtained from a pseudo-random number generator in the form of a uniform distribution i, with p being uniformly distributed over the interval $[0, 1]$, while i is uniformly selected from the interval $[1, N]$. Here, N indicates the cardinality of the path cache set. If $p < p_{goal}$, an additional vertex is created as a result of the algorithmic process towards the desired objective point. If $p_{goal} < p < p_{goal} + p_{path}$, a novel vertex is produced in the direction of the i-th point on the stored trajectory; if not, in the absence of this condition a new node is produced towards a randomly selected location. The corresponding sampling formula is thus derived as follows:

$$x_{rand} = \begin{cases} x_{goal} & p < p_{goal} \\ x_{path} & p_{goal} < p < p_{goal} + p_{path} \\ x_{free} & p > p_{goal} + p_{path} \end{cases} \tag{9}$$

Among them, x_{goal} denotes the positional values of the aimed destination, whereas $x_{path,i}$ denotes the coordinate of the i-th node in the stored trajectory. Additionally, x_{free} denotes the positional values of a point randomly selected from the sampled space. Upon examination of Equation (8), it is evident that the summation of the probabilities yields a constant value. Hence, the probability of random sampling is affected by the target offset probability and the path cache offset probability. The significant reduction of irrelevant node exploration during tree traversal hinges on guiding its growth towards more effective paths. This is achieved by incorporating directional cues such as pre-planned path nodes and target points in random tree variations, ultimately diminishing random sampling while eliminating unnecessary node computation. The algorithm proposed in this paper is depicted in the overall flowchart illustrated in Figure 3.

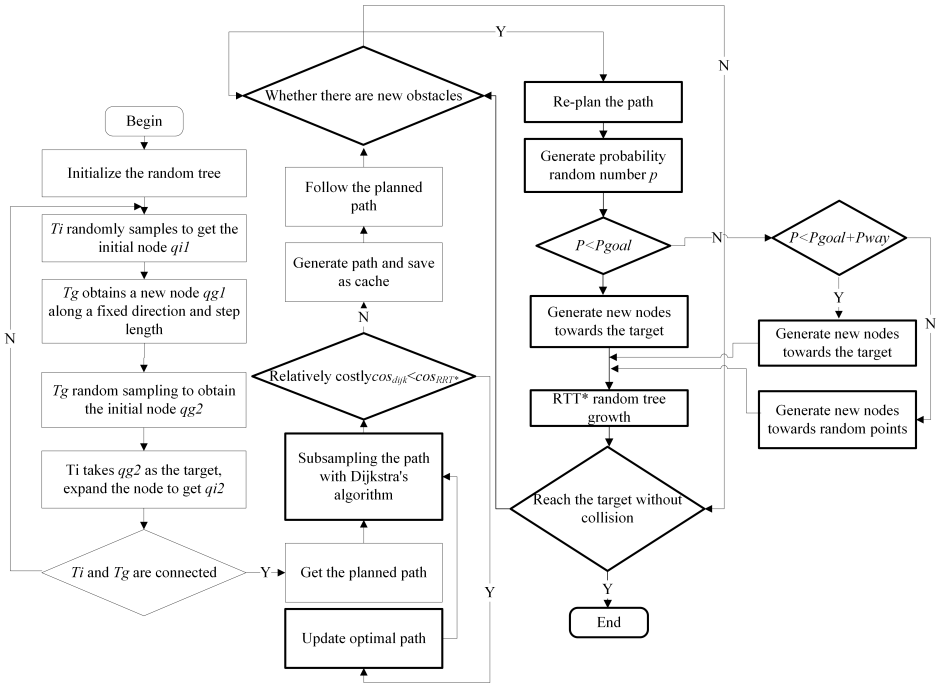

Figure 3. Schematic diagram of two-way extended RRT* algorithm. T_i and T_g mean two random trees. q_i and q_g represent the node of random sampling expansion. The bolded boxes highlight points of divergence between the methodology proposed in this paper and other existing approaches.

3.3. Algorithm

The preceding sections delineate the procedural schema of the refined algorithm. The ensuing script depicts the pseudocode for devising a trajectory for a subaqueous vehicle predicated on the BCRRT* algorithm (Algorithm 1):

Algorithm 1: Bidirectional Path Planning and Cache RRT* Algorithm

T ← initializeTree() for k = 1 to k do If $p < p_{goal}$
  $X_{rand} = X_{goal}$
 else if $p_{goal} < p < p_{goal+}p_{path}$
  $X_{rand} = X_{path,i}$
 else
  $X_{rand} = X_{free}$
 $X_{near} \leftarrow$ Nearst_Neighbor(X_{rand},T);
 $X_{new} \leftarrow$ Steer(X_{rand}, X_{near})
  u ← Select_Input(X_{rand}, X_{near});
  $X_{new} \leftarrow$ New_State(X_{near}, u, Δt);
  if obstacle free(X_{new}) then
  return X_{new};
 $X_{near\_neighbor} \leftarrow find\_near\_neighbor$(T, X_{new}, r);
if obstacle free(X_{new}, T, r) then
 T ← Choose_parent(X_{new}, $X_{near\_neighbor}$, T);
 for each $X_{near\_neighbor}$ calculate(dist(X_{new}, $X_{near\_neighbor}$) + $cost$ ($X_{near\_neighbor}$, X_{init}))
 $X_{new\_parent} \leftarrow X_{near\_neighbor}$, min$cost$;
 return X_{new_parent};
T ← rewire(T, X_{new}, $X_{near\_neighbor}$);
 for each $X_{near\_neighbor}$ calculate(dist($X_{near\_neighbor}$, X_{new}) + $cost$(X_{new}, X_{init}))
 X_{new} min$cost$ ← min(dist($X_{near\_neighbor}$, X_{new}) + $cost$(X_{new}, X_{init}));
 $X_{near\_neighbor}$, newparent ← $X_{xnewmincost}$;
 return $X_{near\_neighbor}$, newparent;
end

4. Experiments

This section introduces the experimental platform and conducts an analysis of the experimental results. Sections 4.1 and 4.2 provide a detailed exposition of these aspects. The core algorithm code link designed in this article is https://github.com/Fngyang/BCRRT. (accessed on accessed on 22 January 2024).

4.1. Implementation and Testbeds

During the experimental phase, a comparative study was conducted between the rudimentary RRT* algorithm and the improved approach proposed in this manuscript. To construct the simulated space with obstacles we refer to the work of Shen et al. [38]. The evaluation was performed using Python 3.7 and Matplotlib on an Intel Core i5-6500 processor with a clock speed of 3.20 GHz, a memory capacity of 4 GB, and the Windows 7 operating system. For this purpose, the search step was set to a fixed value of 5 cm, while the loop was allowed to terminate after 4000 iterations. The testbed comprised a three-dimensional map with dimensions of 100 × 100 × 100, where the starting and ending points were (5,5,5) and (95,95,95), respectively. The display modules were responsible for generating both the random tree and path information. The red lines denoted the branches of the random tree, whereas the black lines represented the output path. To ensure consistency in results, the same map was deployed for algorithmic validation in this research work.

4.2. Experiment Analysis

Figure 4 portrays the path obtained through the basic RRT* algorithm. Owing to the stochasticity of sampling, the trajectory generated may differ despite having identical obstacle attributes and locations on the map. Notably, p has been assigned a fixed value of 5 in this study. The ellipsoidal entity highlighted in a brownish-red hue represents an obstacle, while the corresponding path is depicted in a highlighted brownish color. The green depictions signify nodes, whereas the expansion of nodes leads to connections with other pre-existing nodes within the field, as represented by thin red lines.

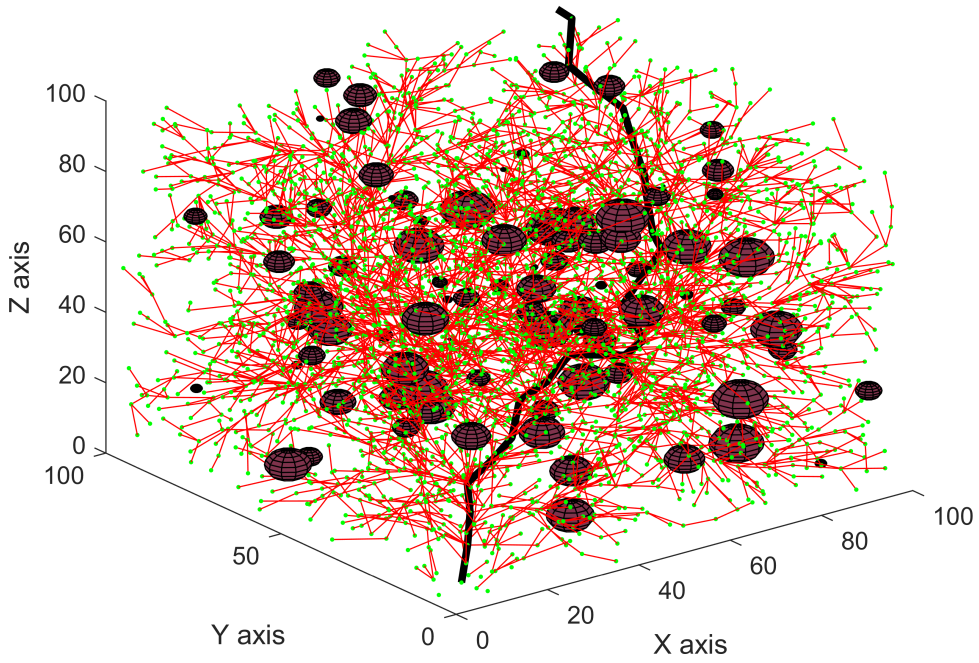

Figure 4. Experimental results of RRT* algorithm. The three-dimensional space depicted in the figure represents a synthetically simulated underwater environment. The variably sized reddish-brown spheres in the illustration simulate underwater obstacles. The thin red lines and green dots within the figure denote the exploration and planning processes. The thicker reddish-brown lines in the figure represent the ultimately planned optimal path.

The figure above demonstrates that the RRT* algorithm, which utilizes random uniform sampling, produces points uniformly distributed in the free space. While this approach explores the entirety of the spatial region, a majority of the randomly sampled points are not helpful for final path computation. To address this issue, the RRT* algorithm employs parent node optimization at every sampled point, causing more nodes within the vicinity of the new node to undergo optimization as the random tree expands. This leads to a considerably higher computational complexity and greater memory consumption.

As delineated in Figure 5, the crimson trace represents the preliminary planning trajectory swiftly acquired via a bidirectional expansion of the random tree approach, while the highlighted tawny contour corresponds to the supremum trajectory fine-tuned by means of the Dijkstra algorithm. In comparison with the initial trajectory, the trajectory generated by the Dijkstra algorithm manifests a reduced number of sampling nodes.

Table 1 displays the computational performance comparison of diverse techniques within a Python-based simulation setting while maintaining an equal number and location of obstacles, and consistent starting and ending points. Each procedure was implemented in 50 repetitions, generating 50 unique datasets of trajectory length and time allocation values that were subsequently combined using mean averaging. The table reveals that the RRT algorithm is characterized by high computational overhead and low node utilization. In contrast, the advanced RRT* algorithm proposed in this study, while mitigating

planning time, also significantly improves the quality of planned paths, thus optimizing search efficiency.

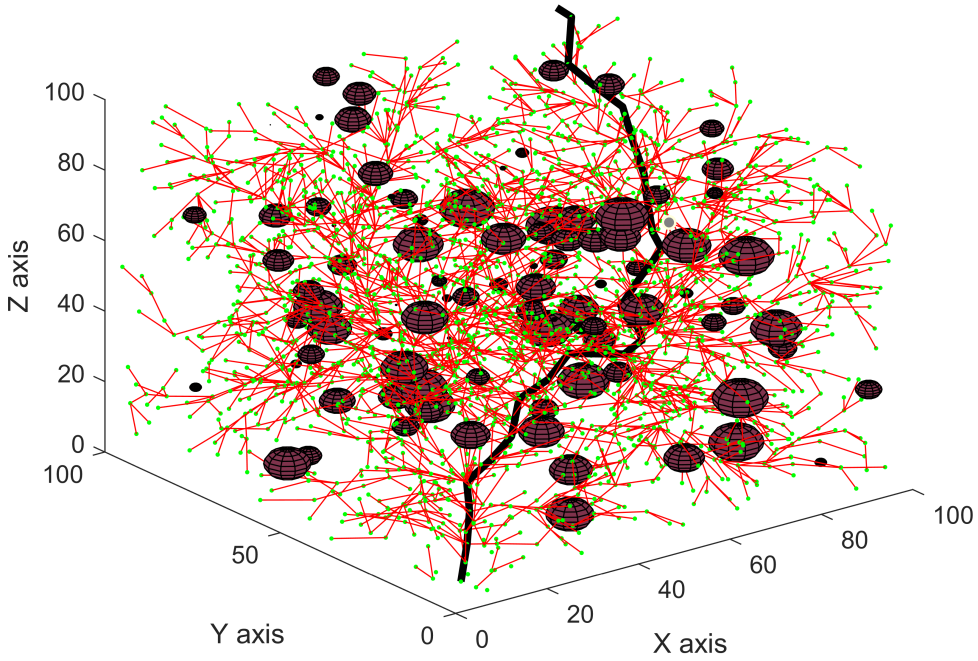

Figure 5. Two-way extended RTT* algorithm effect diagram. The three-dimensional space depicted in the figure represents a synthetically simulated underwater environment. The variably sized reddish-brown spheres in the illustration simulate underwater obstacles. The thin red lines and green dots within the figure denote the exploration and planning processes. The thicker reddish-brown lines in the figure represent the ultimately planned optimal path.

Table 1. An evaluation of various techniques was conducted by comparing the path length and planning time produced by each method. RRT means rapidly exploring random trees. RRT* means optimal kinodynamic motion planning using incremental sampling-based methods.

Method	Planning Trajectory Length/cm	Time Allocation/s
RRT	144.16	12.75
Basic RRT*	137.43	9.21
Dijkstra algorithm optimizes RRT*	101.20	4.07

Owing to the inherent nature of the RRT* algorithm, ensuring identical path planning under the same constraint conditions remains a challenging task. To address this limitation, We use the initial result as a path cache for subsequent iterations, thus reducing the randomness associated with each planning cycle and mitigating the generation of superfluous nodes. Furthermore, as the RRT* algorithm can determine the optimal path solution within the current random tree configuration, coupling it with the path cache can substantially enhance the quality of re-planned paths that are closer to achieving global optimality. In order to evaluate the efficacy of the path cache, we conducted multiple computations on the same map in an unvarying environment. Our results, presented in Figure 6 and Table 2, demonstrate the effectiveness of using the path cache to obtain more efficient path-planning outcomes while maintaining low node utilization.

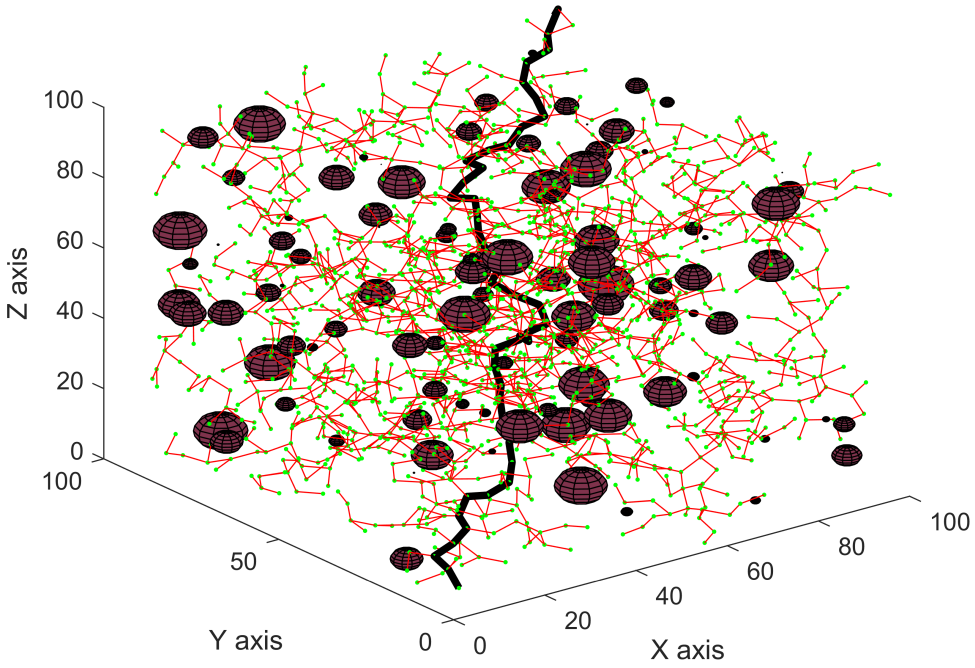

Figure 6. Two-way expansion and path cache optimization RTT* algorithm effect diagram. The three-dimensional space depicted in the figure represents a synthetically simulated underwater environment. The variably sized reddish-brown spheres in the illustration simulate underwater obstacles. The thin red lines and green dots within the figure denote the exploration and planning processes. The thicker reddish-brown lines in the figure represent the ultimately planned optimal path.

Table 2. Node usage of path planning with path cache.

	Total Node Count Employed	Total Path Node Count Employed	Node Utilization/%
Initial planning	1902	42	2.20
First	1530	36	2.28
Second	1430	31	2.09
Third	1007	28	2.87
Fourth	871	24	2.86
Fifth	634	21	3.26

Based on the analysis of Figure 6 and Table 2, it can be observed that utilizing path cache in path planning results in a high level of consistency in generating effective paths. Additionally, by reducing the randomness during sampling, fewer useless nodes are explored. However, the number of nodes does not decrease with the number of replans; this is dependent on the distance between the expansion step and start/end points. Under equal conditions and a certain step length, the path's length and nodes cannot be reduced infinitely, but rather tend towards an optimal solution. Although the number of path nodes has experienced fluctuations, the effectiveness of node utilization has increased significantly attributable to the concomitant decrease in their application.

According to the two-way path planning and cached RRT* algorithm employed by the underwater robot, the alterations of the coordinates pertaining to a specific addressing pathway are visually depicted in Figure 7.

After a comprehensive understanding of the efficacy of the enhanced algorithms, we conducted 100 simulation experiments under randomly generated obstacle environments for both algorithms. Concurrently, we documented the usage of nodes to scrutinize their stability and other parameters. The results, as depicted in Figure 8, illustrate line graphs representing the runtime data derived from the aforementioned simulations. The two

curves in Figure 8 reveal that, through numerous iterations of random obstacle experiments, the algorithm devised in this paper consistently requires fewer nodes to re-plan new paths when obstacles undergo changes.

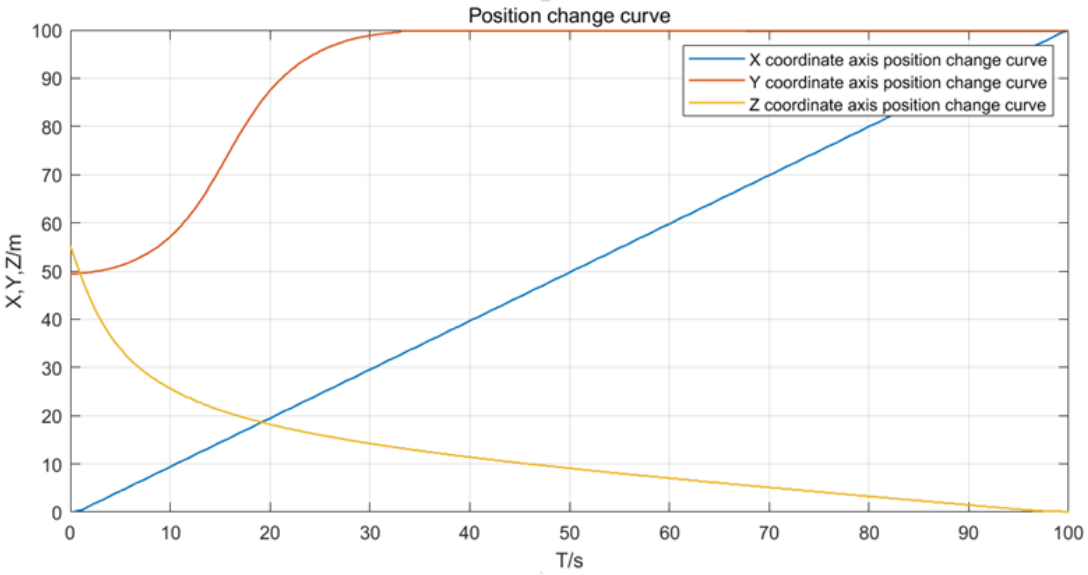

Figure 7. Position change curve. Three curves represent the trajectory from the starting point to the endpoint in the three-dimensional coordinates of X, Y, and Z.

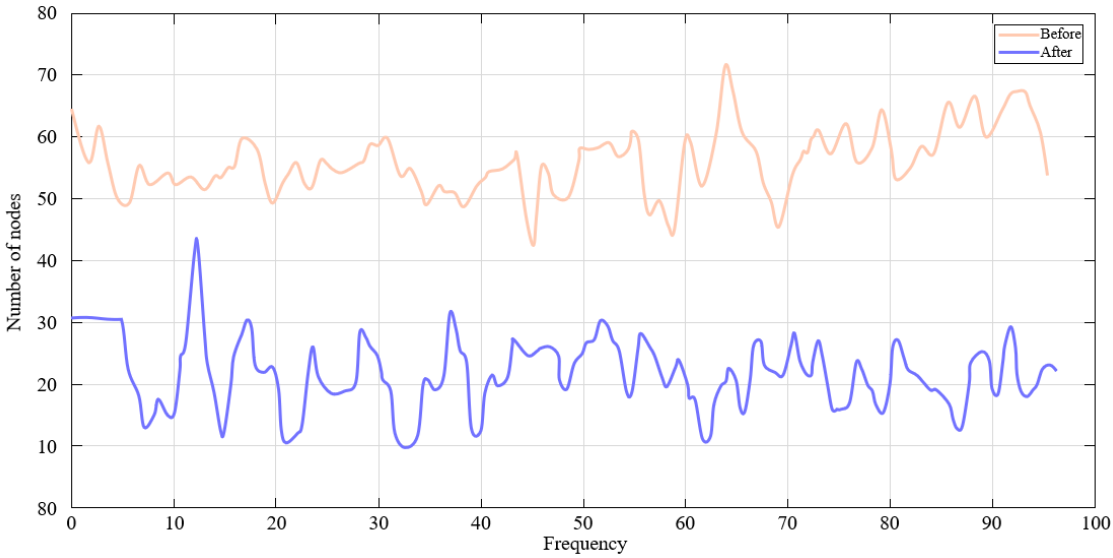

Figure 8. The number of nodes in path planning before and after changes in obstacles. The figure depicts 100 iterations of repeated experiments with randomly changing obstacles. The orange curve represents the number of nodes in path planning before obstacles undergo changes, while the blue curve represents the number of nodes in path planning after obstacle modifications.

The enhanced algorithm incorporates a path cache offset to mitigate the likelihood of random sampling. This restriction on the number of nodes utilized curtails redundant calculations and diminishes memory consumption. Fundamentally, the execution time is predicated on the number of iterations, and therefore, a reduction in the quantity of nodes employed corresponds to a proportional diminution in running time. The RRT* algorithm introduces additional superfluous computations to attain improved outcomes. Nevertheless, the computational overhead incurred by this aspect has been partially mitigated via the utilization of path buffer offset sampling. Subsequently, the average computational

time under both environments registered a 58.79% and 61.99% decline correspondingly. In general, the enhanced algorithm economizes around 60% of computation time, thereby demonstrating the interrelationship between node count and computation time. Additionally, since the quantum of calculations is directly proportional to the size of the random tree, declining node numbers further alleviates the dimensions of the random tree, culminating in a greater reduction in computation time relative to node count.

To substantiate the efficacy of the design methodology proposed in this paper, empirical investigations were conducted utilizing three sophisticated optimization algorithms: the classical ant colony algorithm, the particle swarm algorithm, and the genetic algorithm. Comparative analyses were performed in simulated scenarios, examining performance metrics such as turning points and iteration counts. Figure 9 delineates the number of operations executed by the four algorithms to ascertain the optimal path. Despite the basic ant colony algorithm requiring 18 iterations for convergence, the particle swarm algorithm requiring 13 iterations, and the genetic algorithm necessitating 14 iterations, the optimization algorithm proposed in this study achieves convergence in only 8 iterations, vividly showcasing its prowess in identifying the shortest path.

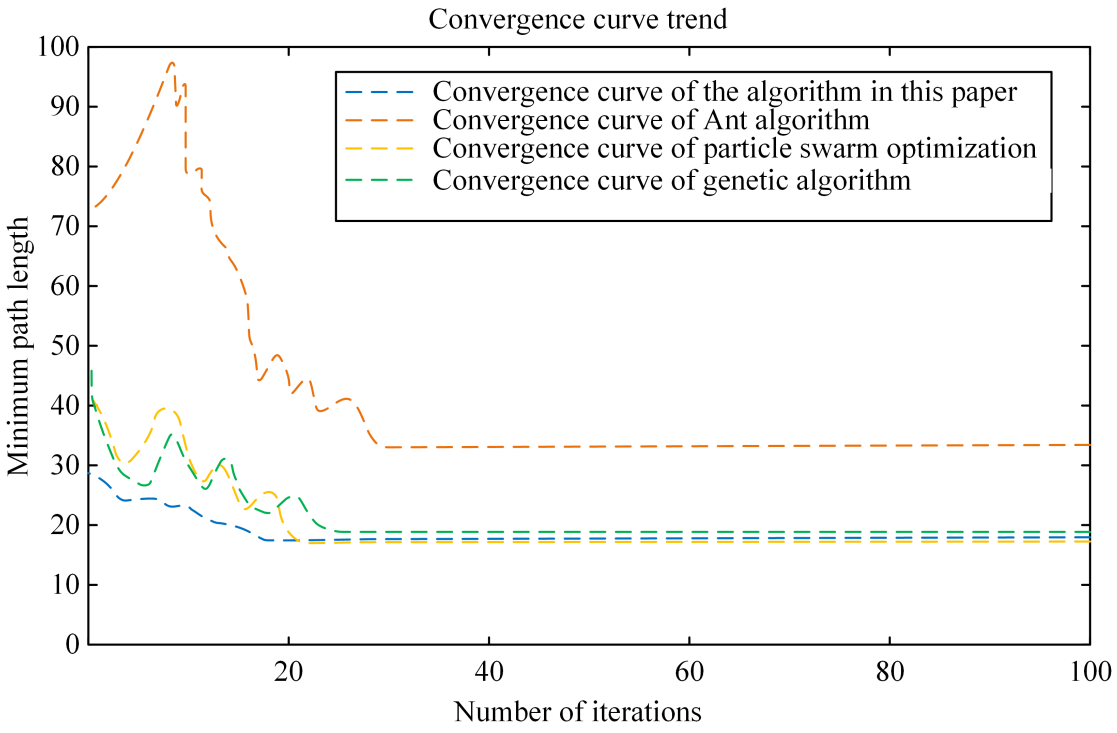

Figure 9. Minimum path length and number of iterations planned by different algorithms.

Table 3 presents a comparative analysis of simulation results for four algorithms. The research findings indicate that the algorithm proposed in this paper exhibits rapid and efficient capabilities in optimal path search. Diverging from ant colony algorithms, this method outperforms the basic ant colony algorithm, particle swarm algorithm, and genetic algorithm, reducing path lengths by 31% and 10%, respectively. Moreover, it achieves the fastest convergence rate while enhancing global optimization capabilities. In summary, the algorithm presented in this paper offers unique advantages in terms of minimum path length, iteration count, and turning point count.

Table 3. A comparative analysis was carried out to evaluate the findings obtained from the three algorithms.

Evaluating Indicator	Ant Colony Algorithm	Particle Swarm Algorithm	Genetic Algorithm	Our algorithm in this Article
Optimal path length	32.97	30.38	30.86	30.38
Number of iterations	18	13	14	8
Number of inflection points	17	13	14	13

5. Conclusions

To enhance the efficacy of cable planning and layout search, as well as improve wiring quality in digital design, this paper proposes an optimized RRT algorithm. The proposed strategy utilizes a bidirectional extended random tree to achieve fast global search between the start and target points. To further optimize the shortest distance calculation between path nodes, a Dijkstra path is introduced, which eliminates redundant nodes in the initial path. Additionally, the path buffer offset is implemented in order to correct inconsistencies caused by random sampling, effectively obtaining the optimal path. Furthermore, a Python static simulation experiment is conducted to validate the efficacy of two strategies; two-way extended sampling and path caching. In comparison with alternative approaches, the underwater robotic path-planning method proposed in this study exhibits superior planning efficacy within simulated underwater environments. This accomplishment not only establishes a benchmark for practical applications in underwater robotic path planning but also underscores its potential value. It is noteworthy that while the distribution of obstacles has been considered in this study, attributes such as material composition and volume of objects in underwater path planning were not accounted for in the assessment of planning effectiveness. In future endeavors, additional factors, including the motion states of underwater devices and the presence of mobile obstacles, will be incorporated into algorithmic design to enhance the comprehensiveness of the analysis.

Author Contributions: Conceptualization, J.G. and Y.Z.; funding acquisition, X.G.; investigation, J.W.; resources, Y.Z.; supervision, Y.Z.; writing—original draft, J.G.; writing—review and editing, Y.Z. and X.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research is funded by the Innovation project of Hainan Province of China (NO. Qhyb2022-82), Innovation project of Hainan Province of China (NO. Qhys2022-154), Key research and development planned project of China (NO. 2018YFB1700501) and Key Development Project of Hainan Province of China (No. ZDYF2019024).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors upon request.

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

References

1. Yuan, S.; Li, Y.; Bao, F.; Xu, H.; Yang, Y.; Yan, Q.; Zhong, S.; Yin, H.; Xu, J.; Huang, Z.; et al. Marine environmental monitoring with unmanned vehicle platforms: Present applications and future prospects. *Sci. Total Environ.* **2023**, *858*, 159741. [CrossRef] [PubMed]
2. Gao, J.; Geng, X.; Zhang, Y.; Tang, H.; Bhatti, U. PE-Transformer: Path enhanced transformer for improving underwater object detection. *Expert Syst. Appl.* **2024**, 123253. [CrossRef]
3. Tu, Q.; Hao, Z. Validation of Sea Surface Temperature Derived From Himawari-8 by JAXA. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2020**, *13*, 448–459. [CrossRef]

4. Gao, J.; Geng, X.; Zhang, Y.; Wang, R.; Shao, K. Augmented weighted bidirectional feature pyramid network for marine object detection. *Expert Syst. Appl.* **2024**, *237*, 121688. [CrossRef]
5. Visbeck, M. Ocean science research is key for a sustainable future. *Nat. Commun.* **2018**, *9*, 690. [CrossRef] [PubMed]
6. Xu, H.; Tang, D.; Sheng, J.; Liu, Y.; Sui, Y. Study of dissolved oxygen responses to tropical cyclones in the bay of Bengal based on Argo and satellite observations. *Sci. Total Environ.* **2019**, *659*, 912–922. [CrossRef]
7. Levy, J.; Hunter, C.; Lukacazyk, T.; Franklin, E.C. Assessing the spatial distribution of coral bleaching using small unmanned aerial systems. *Coral Reefs* **2018**, *37*, 373–387. [CrossRef]
8. Ani, C.J.; Robson, B. Responses of marine ecosystems to climate change impacts and their treatment in biogeochemical ecosystem models. *Mar. Pollut. Bull.* **2021**, *166*, 112223. [CrossRef]
9. Oberbeckmann, S.; Labrenz, M. Marine microbial assemblages on microplastics: Diversity, adaptation, and role in degradation. *Annu. Rev. Mar. Sci.* **2020**, *12*, 209–232. [CrossRef]
10. Alexander, K.A.; Fleming, A.; Bax, N.; Garcia, C.; Jansen, J.; Maxwell, K.H.; Melbourne-Thomas, J.; Mustonen, T.; Pecl, G.T.; Shaw, J.; et al. Equity of our future oceans: practices and outcomes in marine science research. *Rev. Fish Biol. Fish.* **2021**, *32*, 297–311. [CrossRef]
11. Shu, Q.; Wang, Q.; Song, Z.; Qiao, F. The poleward enhanced Arctic Ocean cooling machine in a warming climate. *Nat. Commun.* **2021**, *12*, 2966. [CrossRef] [PubMed]
12. Hucke-Gaete, R.; Viddi, F.A.; Simeone, A. Marine mammals and seabirds of Chilean Patagonia: Focal species for the conservation of marine ecosystems. In *Conservation in Chilean Patagonia: Assessing the State of Knowledge, Opportunities, and Challenges*; Springer: Cham, Switzerland, 2024; pp. 233–261.
13. Groshev, M.; Baldoni, G.; Cominardi, L.; de la Oliva, A.; Gazda, R. Edge robotics: Are we ready? An experimental evaluation of current vision and future directions. *Digit. Commun. Netw.* **2023**, *9*, 166–174. [CrossRef]
14. Kieu, H.T.; Law, A.W.K. Remote sensing of coastal hydro-environment with portable unmanned aerial vehicles (pUAVs) a state-of-the-art review. *J. Hydro-Environ. Res.* **2021**, *37*, 32–45. [CrossRef]
15. Bhatti, U.A.; Mengxing, H.; Neira-Molin, H.; Marjan, S.; Baryalai, M.; Hao, T.; Wu, G.; ullah Bazai, S. MFFCG–Multi feature fusion for hyperspectral image classification using graph attention network. *Expert Syst. Appl.* **2023**, 229, 120496. [CrossRef]
16. Bhatti, U.A.; Tang, H.; Wu, G.; Marjan, S.; Hussain, A. Deep learning with graph convolutional networks: An overview and latest applications in computational intelligence. *Int. J. Intell. Syst.* **2023**, *2023*, 8342104. [CrossRef]
17. Chow, S.; Chang, D.; Hollinger, G.A. Parallelized Control-Aware Motion Planning with Learned Controller Proxies. *IEEE Robot. Autom. Lett.* **2023**, *8*, 2237–2244. [CrossRef]
18. Erke, S.; Bin, D.; Yiming, N.; Qi, Z.; Liang, X.; Dawei, Z. *An Improved A-Star Based Path Planning Algorithm for Autonomous Land Vehicles*; SAGE Publications Sage: London, UK, 2020; Volume 17, p. 1729881420962263.
19. Rostami, S.M.H.; Sangaiah, A.K.; Wang, J.; Kim, H.j. Real-time obstacle avoidance of mobile robots using state-dependent Riccati equation approach. *EURASIP J. Image Video Process.* **2018**, *2018*, 79. [CrossRef]
20. Ichter, B.; Pavone, M. Robot Motion Planning in Learned Latent Spaces. *IEEE Robot. Autom. Lett.* **2019**, *4*, 2407–2414. [CrossRef]
21. Tsardoulias, E.G.; Iliakopoulou, A.; Kargakos, A.; Petrou, L. A review of global path planning methods for occupancy grid maps regardless of obstacle density. *J. Intell. Robot. Syst.* **2016**, *84*, 829–858. [CrossRef]
22. Ichter, B.; Schmerling, E.; Lee, T.W.E.; Faust, A. Learned critical probabilistic roadmaps for robotic motion planning. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 9535–9541.
23. Short, A.; Pan, Z.; Larkin, N.; Van Duin, S. Recent progress on sampling based dynamic motion planning algorithms. In Proceedings of the 2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM), Banff, AB, Canada, 12–15 July 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 1305–1311.
24. Yang, Y.; Ivan, V.; Merkt, W.; Vijayakumar, S. Scaling sampling-based motion planning to humanoid robots. In Proceedings of the 2016 IEEE International Conference on Robotics and Biomimetics (ROBIO), Qingdao, China, 3–7 December 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 1448–1454.
25. Kuffner, J.J.; Lavalle, S.M. RRT-Connect: An Efficient Approach to Single-Query Path Planning. In Proceedings of the 2000 ICRA, Millennium Conference, IEEE International Conference on Robotics and Automation, San Francisco, CA, USA, 24–28 April 2000.
26. Schmitt, P.S.; Neubauer, W.; Feiten, W.; Wurm, K.M.; Wichert, G.V.; Burgard, W. Optimal, sampling-based manipulation planning. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 3426–3432.
27. Chu, Z.; Wang, F.; Lei, T.; Luo, C. Path Planning Based on Deep Reinforcement Learning for Autonomous Underwater Vehicles Under Ocean Current Disturbance. *IEEE Trans. Intell. Veh.* **2023**, *8*, 108–120. [CrossRef]
28. Yang, Y.; Juntao, L.; Lingling, P. Multi-robot path planning based on a deep reinforcement learning DQN algorithm. *CAAI Trans. Intell. Technol.* **2020**, *5*, 177–183. [CrossRef]
29. Kyaw, P.T.; Paing, A.; Thu, T.T.; Mohan, R.E.; Vu Le, A.; Veerajagadheswar, P. Coverage Path Planning for Decomposition Reconfigurable Grid-Maps Using Deep Reinforcement Learning Based Travelling Salesman Problem. *IEEE Access* **2020**, *8*, 225945–225956. [CrossRef]
30. Yu, J.; Chen, C.; Arab, A.; Yi, J.; Pei, X.; Guo, X. RDT-RRT: Real-time double-tree rapidly-exploring random tree path planning for autonomous vehicles. *Expert Syst. Appl.* **2024**, *240*, 122510. [CrossRef]

31. Qureshi, A.H.; Ayaz, Y. Potential functions based sampling heuristic for optimal path planning. *Auton. Robot.* **2016**, *40*, 1079–1093. [CrossRef]
32. Kiesel, S.; Gu, T.; Ruml, W. An effort bias for sampling-based motion planning. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots & Systems, Vancouver, ON, Canada, 24–28 September 2017; pp. 2864–2871.
33. Akgun.; Stilman. Sampling heuristics for optimal motion planning in high dimensions. In Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, CA, USA, 25–30 September 2011.
34. Noreen, I.; Khan, A.; Ryu, H.; Doh, N.L.; Habib, Z. Optimal path planning in cluttered environment using RRT*-AB. *Intell. Serv. Robot.* **2018**, *11*, 41–52. [CrossRef]
35. Janson, L.; Ichter, B.; Pavone, M. Deterministic sampling-based motion planning: Optimality, complexity, and performance. *Int. J. Robot. Res.* **2018**, *37*, 46–61. [CrossRef]
36. Klemm, S.; Oberlnder, J.; Hermann, A.; Roennau, A.; Dillmann, R. RRT*-Connect: Faster, Asymptotically Optimal Motion Planning. In Proceedings of the 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO 2015), Zhuhai, China, 6–9 December 2015.
37. Zaid, T.; Qureshi, A.H.; Yasar, A.; Raheel, N. Potentially Guided Bidirectionalized RRT* for Fast Optimal Path Planning in Cluttered Environments. *Robot. Auton. Syst.* **2018**, *108*, 13–27.
38. Shen, J.; Fu, X.; Wang, H.; Shen, S. Fast path planning for underwater robots by combining goal-biased Gaussian sampling with focused optimal search. *Comput. Electr. Eng.* **2021**, *95*, 107–122. [CrossRef]
39. Liu, Y.; Huang, P.; Zhang, F.; Zhao, Y. Distributed formation control using artificial potentials and neural network for constrained multiagent systems. *IEEE Trans. Control. Syst. Technol.* **2018**, *28*, 697–704. [CrossRef]
40. Das, S.K.; Roy, K.; Pandey, T.; Kumar, A.; Dutta, A.K.; Debnath, S.K. Modified Critical Point–A Bug Algorithm for Path Planning and Obstacle Avoiding of Mobile Robot. In Proceedings of the 2020 International Conference on Communication and Signal Processing (ICCSP), Melmaruvathur, India, 28–30 July 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 351–356.
41. Yang, J.; Ni, J.; Xi, M.; Wen, J.; Li, Y. Intelligent path planning of underwater robot based on reinforcement learning. *IEEE Trans. Autom. Sci. Eng.* **2022**, *20*, 1983–1996. [CrossRef]