Hindawi
Discrete Dynamics in Nature and Society
Volume 2020, Article ID 7821942, 11 pages
https://doi.org/10.1155/2020/7821942

Research Article

Research on SBMPC Algorithm for Path Planning of Rescue and Detection Robot

Lin-Lin Wang[1] **and Li-Xin Pan**[2]

[1]*College of Information Engineering, Inner Mongolia University of Technology, Hohhot 010080, China*
[2]*Beijing Institute of Control Engineering, China Academy of Space Technology, Beijing 100190, China*

Correspondence should be addressed to Li-Xin Pan; panlixin1@126.com

Received 28 July 2020; Revised 9 October 2020; Accepted 8 November 2020; Published 23 November 2020

Academic Editor: Emilio Jiménez Macías

This research aims to improve autonomous navigation of coal mine rescue and detection robot, eliminate the danger for rescuers, and enhance the security of rescue work. The concept of model predictive control is introduced into path planning of rescue and detection robot in this paper. Sampling-Based Model Predictive Control (SBMPC) algorithm is proposed basing on the construction of cost function and predictive kinematics model. Firstly, input sampling is conducted in control variable space of robot motion in order to generate candidate path planning solutions. Then, robot attitude and position in future time, which are regarded as output variables of robot motion, can be calculated through predictive kinematics model and input sampling data. The optimum solution of path planning is obtained from candidate solutions through continuous moving optimization of the defined cost function. The effects of the three sampling methods (viz., uniform sampling, Halton's sampling, and CVT sampling) on path planning performance are compared in simulations. Statistical analysis demonstrates that CVT sampling has the most uniform coverage in two-dimensional plane when sample amount is the same for three methods. Simulation results show that SBMPC algorithm is effective and feasible to plan a secure route for rescue and detection robot under complex environment.

1. Introduction

As an important tool for processing coal mine accidents, rescue and detection robot can substitute rescuers to enter the accident area for implementation of detecting and rescuing. Real-time information can be transferred to rescue command center rapidly. Rescue and detection robot can provide scientific basis for the rescue decision, and the rescue plan can be made as accurately as possible. Therefore, it has critical significance for safe production in coal mine and can reduce the loss of public property and lives to develop rescue and detection robot.

For recent years, a large number of research findings are obtained in path planning which is the key technology of robot to realize regional search. To some extent, path planning is regarded as a synthesis of global planning and local planning [1]. In practical application, the robot needs to select an optimal path under some certain criteria such as shortest path, minimal energy consumption, or time consuming at least. In the process of robot exploring along its desired path, local planning is generally implemented according to real-time information from sensor feedback, and motion states are often adjusted for realization of real-time obstacle avoidance.

Path planning method based on random sampling has started in 1990 and originates from Randomized Potential Planner (RPP) algorithm which is firstly proposed by Barraquand and Latombe [2]. RPP is used for solving the problem of local minimum in artificial potential field method and improving planning efficiency in high-dimensional space. Currently, there are three main methods aiming at the problem of path planning for mobile robot with multidegree of freedom, namely, Probabilistic Roadmap Methods (PRM) algorithm [3], Rapidly Exploring Random Tree (RRT) algorithm [4], and Vector Field (VF) algorithm.

The main idea of PRM algorithm is to construct a probabilistic roadmap through local planner firstly, then implement global path planning on this roadmap, and finally attempt to find a collision-free path with graph search algorithm. PRM has the advantage of easy operation, but random sampling points cannot be distributed uniformly in the whole space due to limited sampling in practical application. As a result, connectivity of roadmap is reduced, especially for narrow channel environment, where PRM often fails to find a feasible path. In order to solve this problem, researchers propose many enhanced algorithms for PRM. In [5], the coverage of a graph is defined as a performance index of its optimality as constructed by a sampling-based algorithm and an optimization algorithm is proposed which can maximize graph coverage in the configuration space. The simulation results confirm that the roadmap graph obtained by the proposed algorithm can generate results of satisfactory quality in path-finding tests under various conditions.

The main idea of RRT algorithm is to extend search tree incrementally by browsing the state space in a rapid and uniform way. This method proved to be probability complete; namely, a collision-free path satisfying planning conditions must be found if the amount of nodes is larger than a certain value of threshold. In [6], a new method of dynamic replanning is proposed, which is based on an extended rapidly exploring random tree. An efficient method of node sharing is applied to allow the multirobot team to quickly develop path plans. Various experimental results in both single- and multirobot scenarios show the effectiveness of the proposed methods.

The main idea of VF algorithm is similar to that of potential field algorithm, which has been widely used as a tool for path planning in the robotics community [7]. Vector fields are different from potential fields in that they do not necessarily represent the gradient of a potential [8]. Rather, the vector field simply indicates a desired direction of travel [9]. In [10], a method for accurate path following for miniature air vehicles is developed. The method is based on the notion of vector fields, which are used to generate the desired course inputs to inner-loop attitude control laws. Lyapunov's stability analysis demonstrates asymptotic decay of path-following errors in the presence of constant wind disturbances. This method can be applied for path following of straight-line and circular arcs effectively.

In addition to the above research findings, there are many new methods proposed for path planning [11, 12]. In [13], the theory of charged particles' potential fields is utilized by assigning a potential function for each individual obstacle. The interaction of all scattered obstacles is integrated in a scalar potential surface (SPS) which strongly depends on the physical features of the mobile robot and obstacles. The achieved results demonstrate a feasible, fast, oscillation-free, and collision-free path planning of the proposed method. In [14], a new method called laser simulator (LS) is proposed in order to solve the path planning problem of a nonholonomic three-wheeled mobile robot (WMR). The path planning and roundabout detection are determined based on LS and sensor fusion of a laser range finder, camera, and odometry measurements. Experimental results show the capability of this proposed algorithms to robustly drive the robot.

In the field of Model Predictive Control (MPC) application, there are also successful cases which are worthy of studying. In [15], nonlinear terminal sliding mode control (TSMC) and linear MPC are combined to solve the robust optimal three-axis attitude tracking problem of spacecrafts. The global stability and robust attitude tracking are guaranteed by this method. In [16], a disturbance rejection MPC scheme is developed for tracking nonholonomic vehicle with coupled input constraint and matched disturbances. The whole system is input-to-state stable if no information about the disturbance is available and can reach an offset-free tracking performance if the harmonic frequencies of the disturbance are known. In [17], MPC is used to model and implement controllers for dynamic encirclement of UAVs team. The contributions of this paper lie in the implementation of MPC to solve the problem of dynamic encirclement of a team of UAVs in real time and the application of theoretical stability analysis to the problem.

In this paper, path planning is implemented through Sampling-Based Model Predictive Control (SBMPC) method combined with nonlinear kinematics model of rescue and detection robot. In SBMPC method, cost function with the form used by Model Predictive Control (MPC) [18] is adopted and input of control system is optimized. Different from traditional numerical optimization, SBMPC applies the method of objective-oriented optimization which is widely used in robotics and artificial intelligence field. The process of path planning through SBMPC algorithm is deduced in this paper, and the selection of sampling methods is also analyzed from the viewpoint of effect on planning performance.

2. Design of Path Planning Based on SBMPC

2.1. Kinematics Model of Rescue and Detection Robot. In this paper, rescue and detection robot is regarded as the research subject. As shown in Figure 1, xoy denotes global reference frame and x_RCy_R denotes local reference frame. C denotes the center of robot mass, $v(t)$ is robot forward speed, and $\omega(t)$ is robot angular speed.

Kinematics model of rescue and detection robot [19] can be described as

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{r \cdot \cos\theta}{4} & \frac{r \cdot \cos\theta}{4} \\ \frac{r \cdot \sin\theta}{4} & \frac{r \cdot \sin\theta}{4} \\ -\frac{r}{2l} & \frac{r}{2l} \end{bmatrix} \cdot \begin{bmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \end{bmatrix}, \tag{1}$$

where (x, y) is the robot position in global reference frame and θ is the angle deviation between local reference frame and global reference frame. r is the diameter of motivation wheel, l is the distance from left wheel to right wheel, $\dot{\phi}_1$ is

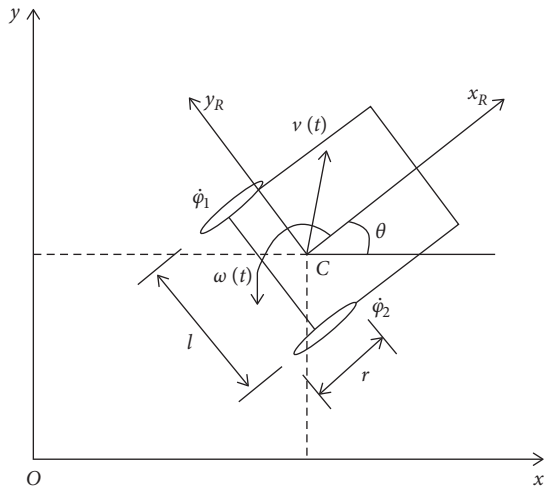

Figure 1: Rescue and detection robot in global reference frame.

the rotation speed of left wheel, and $\dot{\phi}_2$ is the rotation speed of right wheel. In this paper, r is 0.5 m and l is 0.8 m. Equation (1) can be expressed briefly as

$$\dot{X} = B \cdot U, \tag{2}$$

where

$$X = \begin{bmatrix} x & y & \theta \end{bmatrix}^T,$$

$$B = \begin{bmatrix} \frac{r \cdot \cos\theta}{4} & \frac{r \cdot \cos\theta}{4} \\ \frac{r \cdot \sin\theta}{4} & \frac{r \cdot \sin\theta}{4} \\ -\frac{r}{2l} & \frac{r}{2l} \end{bmatrix}, \tag{3}$$

$$U = \begin{bmatrix} \dot{\phi}_1 & \dot{\phi}_2 \end{bmatrix}^T.$$

Equation (2) is the state equation of robot motion. X is the state variable, U is the control input variable, and B is the control input matrix. In the process of path planning, robot attitude and position is regarded as output variable of robot motion, which can be defined as

$$Y = \begin{bmatrix} x & y & \theta \end{bmatrix}^T. \tag{4}$$

Therefore, the output equation of robot motion is expressed as

$$Y = X. \tag{5}$$

Equations (2) and (5) are the kinematics model of rescue and detection robot.

2.2. Predictive Model of Motion Planning. Derivatives of x, y, and θ at the time k can be calculated from equation (1) and expressed as

$$\dot{x}(k) = \frac{r \cdot \cos\theta(k)}{4} \cdot \left(\dot{\phi}_1(k) + \dot{\phi}_2(k)\right), \tag{6}$$

$$\dot{y}(k) = \frac{r \cdot \sin\theta(k)}{4} \cdot \left(\dot{\phi}_1(k) + \dot{\phi}_2(k)\right), \tag{7}$$

$$\dot{\theta}(k) = \frac{r}{2l} \cdot \left(-\dot{\phi}_1(k) + \dot{\phi}_2(k)\right), \quad k = 1, 2, 3, \ldots, \tag{8}$$

where $\dot{\phi}_1(k)$ is the rotation speed of left wheel at the time k and $\dot{\phi}_2(k)$ is the rotation speed of right wheel at the time k. $x(k+1)$, $y(k+1)$, and $\theta(k+1)$ can be calculated through integration and expressed as

$$x(k+1) = x(k) + \dot{x}(k) \cdot T, \tag{9}$$

$$y(k+1) = y(k) + \dot{y}(k) \cdot T, \tag{10}$$

$$\theta(k+1) = \theta(k) + \dot{\theta}(k) \cdot T, \tag{11}$$

where T is the sampling period.

Equations (6)~(11) constitute the predictive model of robot motion and can be used for calculating robot attitude and position in future time through iterative computation.

2.3. Cost Function of Motion Planning. If Shannon's sampling constraints [20] are satisfied, the cost function of robot motion planning based on SBMPC can be described as

$$\min_{\{U(k),\ldots,U(k+N-1)\},N} J = \sum_{i=0}^{N} \|Y(k+i+1) - Y(k+i)\|_{Q(i)} + \sum_{i=0}^{N-1} \|U(k+i) - U(k+i-1)\|_{S(i)}. \tag{12}$$

Variables in equation (12) should satisfy the following inequality conditions; viz.,

$$\left\|Y(k+N) - G_0\right\| \le \varepsilon, \tag{13}$$

$$\begin{aligned} Q(i) &\ge 0, \\ S(i) &> 0. \end{aligned} \tag{14}$$

In equation (12), the term $\|Y(k+i+1) - Y(k+i)\|_{Q(i)}$ represents the edge cost of planning path between the current predictive output $Y(k+i)$ and the next predictive output $Y(k+i+1)$. The term $\|U(k+i) - U(k+i-1)\|_{S(i)}$ represents the energy consumption in robot motion process. In the constraint (13), G_0 denotes the goal state which is used for construction of the terminal constraint of path planning. ε is a small positive number which represents the proximity of robot output to the goal state. The constraint (13), which reflects the requirement that robot should reach the target G_0 with the accuracy ε, is a key condition for ensuring the convergence of planning algorithm. $Q(i)$ and $S(i)$ are the orders of corresponding norms. Therefore, the goal-directed optimization is adopted here, and in this method, the goal is

implicitly considered through the calculation of a rigorous lower bound for the cost from a particular state to G_0.

2.4. Principle of Motion Planning Based on Sampling. Sampling-based motion planning algorithms include Rapidly Exploring Random Trees (RRTs) and randomized A* algorithms [21]. A common feature of these algorithms is that they are applied in the output space of robot and used for generating samples through various strategies. In essence, as shown in Figure 2, sampling-based motion planning methods are applied by using sampling to construct a tree that connects the root (initial state) with a goal region.

Sampling-based path planning algorithm is shown in Figure 3.

Sampling-based path planning algorithm follows the steps below.

Step 1. Initialization: $G(V, E)$ represents a search graph where V denotes the set of nodes and E denotes the set of edges. During the initialization, V only contains start node and E does not contain any edge.

Step 2. Node selection: select a node τ in V, which is regarded as the current node.

Step 3. New edge generation: generate corresponding routes Γ_s for all candidate nodes τ_{new} in C_{free} (the set of candidate nodes). Γ_s satisfies the following constraints:

$$\begin{aligned} \Gamma_s(0) &= \tau, \\ \Gamma_s(1) &= \tau_{\text{new}}. \end{aligned} \tag{15}$$

If the route Γ_s does not exist for a certain candidate node, select the next candidate node to generate its corresponding route. Until above processing covers all candidate nodes, then go to Step 4.

Step 4. New edge insertion: all the routes Γ_s generated in Step 3 are regarded as new edges connecting the node τ to the node τ_{new} and incorporated to E. The node τ_{new} is also incorporated to V if V does not contain τ_{new}.

Step 5. Search for path planning solutions: start from the current node τ in the graph $G(V, E)$ and search for the optimal edge as the best route for node refreshing; meanwhile, the terminal node τ_{new} of the optimal edge is regarded as the starting node of planning path in the next time.

Step 6. Return to Step 2. Repeat the above steps until τ_{new} arrives at the goal node or the path planning time is over.

The novelty of SBMPC method is mainly reflected in three aspects, i.e., input sampling, implicit state grid, and goal-directed optimization, which are described in the following parts.

2.5. Input Sampling. There are two primary disadvantages to use output sampling commonly done in traditional sampling methods. The first limitation is that the algorithm must determine the most ideal node to expand [22]. This selection

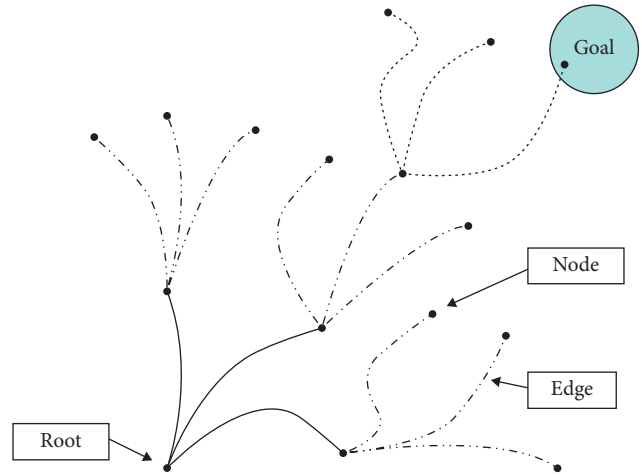

Figure 2: Diagram of tree connecting the root with a goal region.

is typically made based on the proximity of nodes in the graph to a sampled output node and involves a potentially costly nearest-neighbor search. The Local Planning Method (LPM) presents the second and perhaps more troublesome problem, which is determining an input that connects a newly sampled node to the current node [23]. This problem is essentially a two-point boundary value problem that connects one output or state to another. There is no guarantee that such an input exists. Also, for systems with complex dynamics, the search itself can be computationally expensive, which leads to a computationally inefficient planner. In contrast, when the input space is sampled as proposed in this paper, the need for a nearest-neighbor search is eliminated, and the LPM is reduced to the integration of a system model, and therefore, only outputs that are achievable for the system are generated.

In order to visualize this concept, consider a robot that has position (x, y) and orientation θ, which are the outputs of the robot kinematics model. The model restricts the attainable outputs. All the dots in Figure 4 are output nodes obtained from sampling the output space even though only the dots on the mesh surface can physically be achievable by the robot. There are a larger number of dots (sampled outputs) in the output space that do not lie in the achievable region (mesh surface). This means those sampled outputs are not physically achievable, so traditional sampling-based methods would have to finish the search in the whole output space. This leads to an inefficient search that can substantially increase the computational time of the planner. In essence, sampling in the input space leads to more efficient results since each of the corresponding dots in the output space is allowed by the model.

2.6. Implicit State Grid. Although extension from several existing sampling-based paradigms can lead to input sampling algorithms like SBMPC, input sampling has not been used in most planning research. This is most likely due to the fact that input sampling is seen as an inefficient method because it can result in highly dense samples in the output space since input sampling does not inherently lead to a

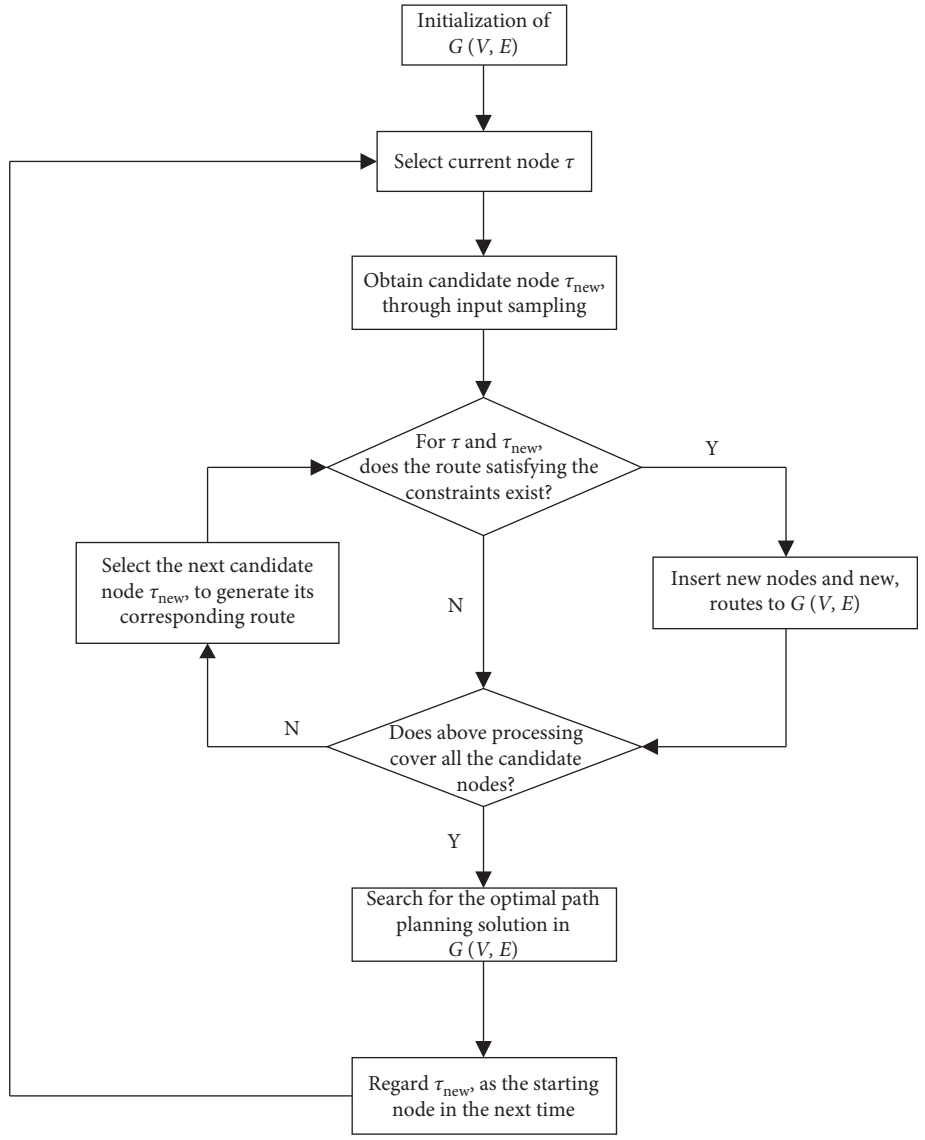

Figure 3: Diagram of sampling-based path planning algorithm.

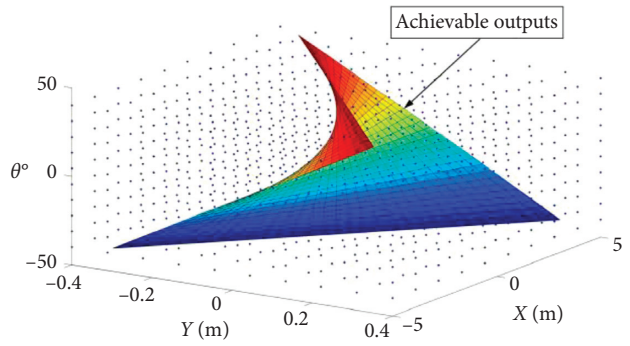

Figure 4: Potential problems in output space sampling.

uniformly discrete output space, such as a uniform grid. This problem is especially evident when encountering a local minimum problem associated with the A* algorithm [24], which can occur when planning in the presence of a large concave obstacle while the goal is on the other side of the obstacle. This situation is considered in depth for discrete 2D path planning in the work of this paper, which discusses that the A* algorithm must explore all the states in the neighborhood of the local minimum, shown as the shaded region of Figure 5, before progressing to the final solution. The issue presented to input sampling methods is that the number of states within the local minimum is infinite due to the lack of a discrete output space.

The concept of an implicit state grid is introduced as a solution to both of the challenges generated by input sampling. The implicit grid ensures that the graph generated by the SBMPC algorithm is constructed such that only one active output (state) exists in each grid cell, limiting the number of nodes that can exist within any finite region of the output space. In essence, the implicit state grid provides a discrete output space. It also allows for the efficient storage of potentially infinite grids by only storing the grid cells that

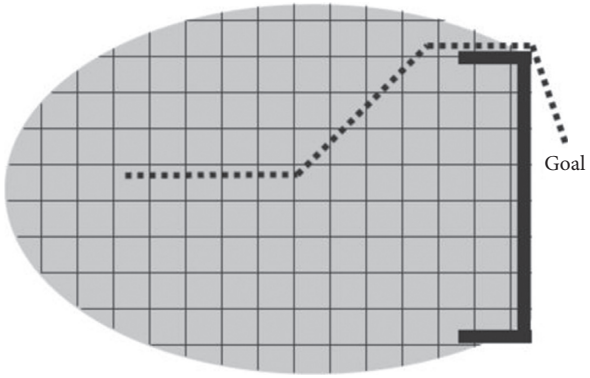

Figure 5: Diagram of implicit state grid.

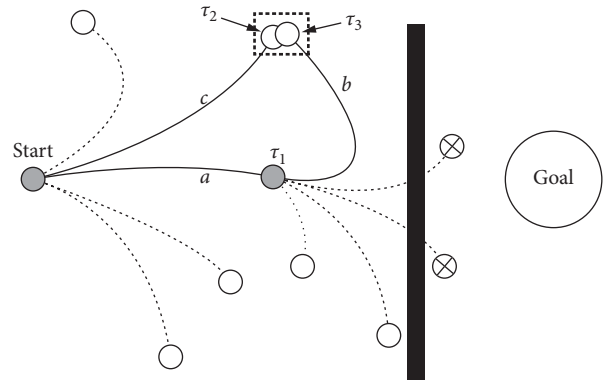

Figure 6: Necessity for introducing implicit state grid.

contain nodes, which is increasingly important for higher-dimensional problems.

As shown in Figure 6, a, b, and c denote the edge length of planning path. Node τ_1 is selected for expansion after which the lowest-cost node is τ_3. The implicit state grid then recognizes that τ_2 and τ_3 are close enough to be considered the same and updates the path to their grid cell; then the modified planning path is c since $c < a + b$.

2.7. Goal-Directed Optimization. There is a class of discrete optimization techniques that have their origin in graph theory and have been further developed in the path planning literature. In this study, these techniques will be called goal-directed optimization and refer to graph search algorithms such as Dijkstra's algorithm [25], A* algorithm, D* algorithm [26], and LPA* algorithm [27]. Given a graph, these algorithms find a path that optimizes some cost of moving from a start node to some given goal. In contrast to discrete optimization algorithms such as branch-and-bound optimization, which is used for solving continuous optimization problems, the goal-directed optimization methods are inherently discrete and usually used for real-time path planning.

Although not commonly recognized, goal-directed optimization methods are capable of solving control theory problems for which the ultimate objective is to plan an optimal trajectory and control inputs to reach a goal (or set point) while optimizing a cost function. Hence, graph search algorithms can be applied to terminal constraint optimization problems and set point control problems. To observe this, consider the tree graph of Figure 2. Each node of this tree can correspond to a system state and the entire tree may be generated by integrating sampled inputs to a system model. Assume that the trajectory cost is given by the sum of corresponding edge (i.e., branch) cost, where each edge cost is dependent not only on the states it connects but also on the inputs that are used to connect those states. The use of the system kinematics model can be viewed simply as a means to generate the directed graph and associated edge cost. Figure 7 shows the block diagram of the optimal node selection based on SBMPC.

2.8. Rescue and Detection Robot Path Planning Based on SBMPC. In this section, path planning algorithm based on SBMPC is presented. Firstly, variables used in this algorithm are defined. Sampling-Based Model Predictive Optimization (SBMPO) is similar to LPA* algorithm, and the difference between the two methods lies in the way neighborhood solution is generated. In this paper, states for future time are generated by introducing predictive model of robot motion with certain constraints shown in Part B of Section 2.

SBMPC operates on a dynamic directed graph G which is a set of all nodes and edges currently in the graph. Node τ denotes the motion states of rescue and detection robot, namely, $\tau = [x, y, \theta]^T$. $\mathrm{SUCC}(\tau)$ represents the set of successors (children) of node $\tau \in G$ while $\mathrm{PRED}(\tau)$ denotes the set of all predecessors (parents) of node $\tau \in G$. The cost of traversing from node τ to node $\tau' \in \mathrm{SUCC}(\tau)$ is denoted by $c(\tau, \tau')$, where τ satisfies $0 < c(\tau, \tau') < 1$. The optimization component of SBMPC is SBMPO which is an algorithm that determines the optimal cost (i.e., shortest path, shortest time, least energy, etc.) from a start node $\tau_{\mathrm{start}} \in G$ to a goal node $\tau_{\mathrm{goal}} \in G$. The start distance of node $\tau \in G$ is given by $g^*(\tau)$ which is the cost of the optimal path from the given start node τ_{start} to the current node τ. This paper will focus on determining the shortest path through SBMPC algorithm.

SBMPC maintains two estimates of $g^*(\tau)$. The first estimate $g(\tau)$ is essentially the current cost from τ_{start} to the node τ while the second estimate $\mathrm{rhs}(\tau)$ is a one-step look-ahead estimate based on $g(\tau')$ for $\tau' \in \mathrm{PRED}(\tau)$ and provides more information than the estimate $g(\tau)$. The $\mathrm{rhs}(\tau)$ value satisfies

$$\mathrm{rhs}(\tau) = 0, \quad \text{if } \tau = \tau_{\mathrm{start}}, \tag{16}$$

$$\mathrm{rhs}(\tau) = \min_{\tau' \in \mathrm{PRED}(\tau)} \left(g(\tau') + c(\tau', \tau)\right), \quad \text{otherwise} \tag{17}$$

A node τ is locally consistent if $g(\tau) = \mathrm{rhs}(\tau)$ and locally inconsistent if $g(\tau) \neq \mathrm{rhs}(\tau)$. If all nodes are locally consistent, then $g(\tau)$ satisfies equation (17) for all $\tau \in G$ and is therefore equal to the start distance. This enables the ability to trace the shortest path from τ_{start} to any node τ by starting at τ and traversing to any predecessor τ' that minimizes $g(\tau') + c(\tau', \tau)$ until τ_{start} is reached.

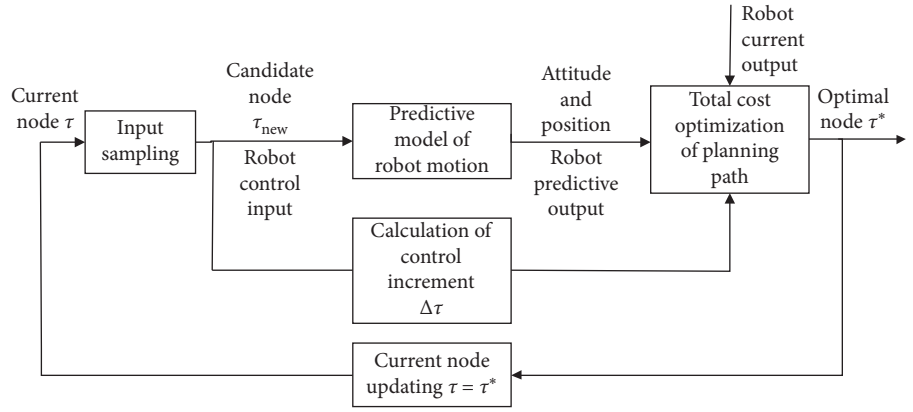

Figure 7: Optimal node selection based on SBMPC.

To facilitate fast replanning, SBMPO does not make every node locally consistent after an edge cost changes and instead it uses a heuristic function $h(\tau, \tau_{goal})$ to focus the search so that it only updates $g(\tau)$ for nodes necessary to obtain the shortest path. The heuristic is used to approximate the goal distances and must follow the triangle inequality $h(\tau_{goal}, \tau_{goal}) = 0$ and $h(\tau, \tau_{goal}) \leq c(\tau, \tau') + h(\tau', \tau_{goal})$ for all nodes $\tau \in G$ and $\tau' \in \mathrm{SUCC}(\tau)$. SBMPO employs the heuristic function along with the start distance estimates to rank the priority queue containing the locally inconsistent nodes and thus all the nodes that need to be updated in order for them to be locally consistent. The priority of a node is determined by a two-component key vector

$$\mathrm{key}(\tau) = \begin{pmatrix} k_1(\tau) \\ k_2(\tau) \end{pmatrix} = \begin{pmatrix} \min(g(\tau), \mathrm{rhs}(\tau)) + h(\tau, \tau_{goal}) \\ \min(g(\tau), \mathrm{rhs}(\tau)) \end{pmatrix}, \tag{18}$$

where the keys are ordered lexicographically with the smaller key values having a higher priority.

The SBMPC algorithm contains three main functions: SBMPC, SBMPO, and Neighbor Generation. The main SBMPC algorithm follows the general structure of MPC where SBMPO repeatedly computes the optimal path between the current state $\tau_{current}$ and the goal state τ_{goal}. After a single path is generated, $\tau_{current}$ is updated to reflect the implementation of the first control input and the graph G is updated to reflect any system changes. These steps are repeated until the goal state is reached.

The second algorithm SBMPO repeatedly generates the neighbors of locally inconsistent nodes until τ_{goal} is locally consistent or the key of the next node in the priority queue is not smaller than key (τ_{goal}). The node τ_{best} with the highest priority (lowest key value) is on top of the priority que. The algorithm then deals with two potential cases based on the consistency of the expanded node τ_{best}. If the node is locally overconsistent, $g(\tau) > \mathrm{rhs}(\tau)$, the g-value is set to $\mathrm{rhs}(\tau)$ making the node locally consistent. The successors of τ are then updated.

The node updating process includes recalculating $\mathrm{rhs}(\tau)$ and key values, checking for local consistency and either adding or removing the node from the priority queue accordingly. For the case when the node is locally underconsistent, $g(\tau) < \mathrm{rhs}(\tau)$, the g-value is set to ∞ making the node either locally consistent or overconsistent. This change can affect the node along with its successors which then goes through the node updating process.

The Neighbor Generation algorithm determines the successor nodes of the current node. In the input space, a set of quasirandom samples are generated that are then used with the predictive model of robot motion to calculate a new set of outputs (states) with $\tau_{current}$ being the initial condition. The branching factor N_b determines the number of paths that will be generated. The path is represented by a sequence of states $\tau(k)$ for $k = k_1, k_1 + \Delta k, \ldots, k_2$, where Δk is the model step size. The set of states that do not violate any state or obstacle constraints is called τ_{free}. If $\tau(k) \in \tau_{free}$, then the new neighbor node τ_{new} and the connecting edge can be added to the graph. If $\tau_{new} \in$ STATE_GRID, then the node currently exists in the graph and only the new path to reach the existing node needs to be added.

3. Simulation

In order to verify the effectiveness of SBMPC algorithm proposed in this paper, path planning simulation of rescue and detection robot is conducted under the conditions of uniform sampling [28], Halton's sampling [29, 30], and Centroidal Voronoi Tessellation (CVT) sampling [31, 32], respectively. Simulation results show that local minimum value is avoided successfully by using SBMPC algorithm. Specific conditions relating to simulation are described as follows.

3.1. Uniform Sampling. Figure 8 shows the path planning simulation of SBMPC algorithm with uniform sampling, where obstacles are denoted by red circles and robot dimensions are considered for the safety of obstacle avoidance. In Figure 8, the start position is (1, 1) and the goal position is (19, 19). The sample size is 441 at each time k. The total length of planning path in Figure 8 is 30.15 m. Time consumption from start position to goal position is 150.75 s. Figure 9 shows the variation of direction angle θ during the process.

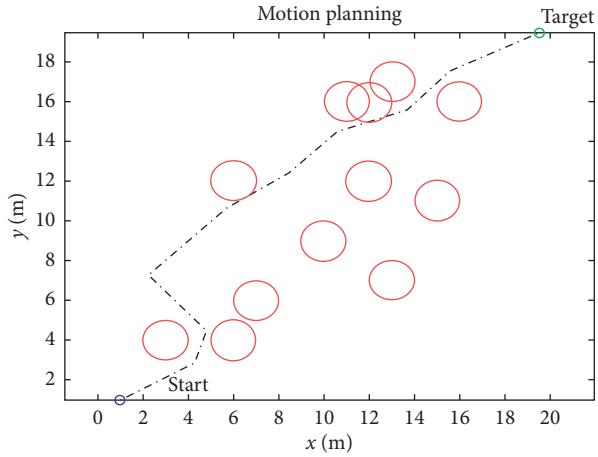

Figure 8: Simulation of SBMPC algorithm with uniform sampling.

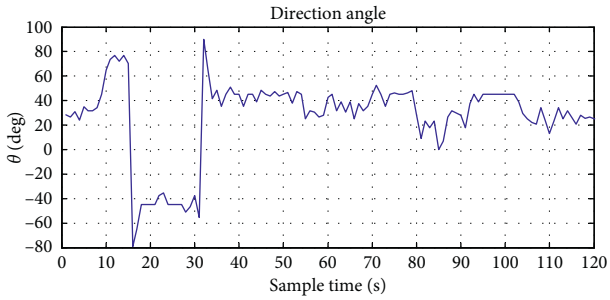

Figure 9: Curve of direction angle while uniform sampling.

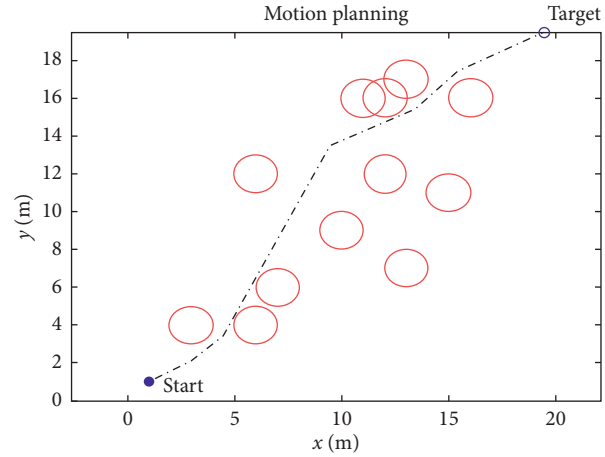

Figure 10: Simulation of SBMPC algorithm with Halton's sampling.

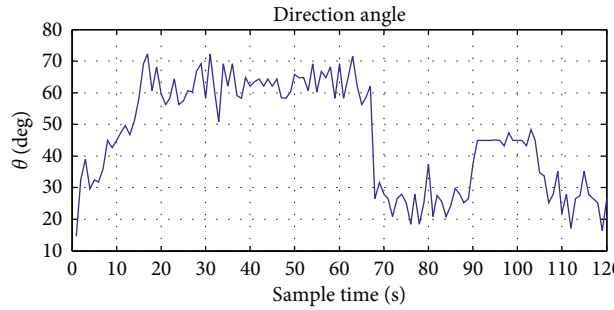

Figure 11: Curve of direction angle while Halton's sampling.

It can be seen from Figures 8 and 9 that the direction angle varies sharply when the sample time is near 15 sec and 30 sec. The simulation result of uniform sampling is not optimal although robot can avoid all the obstacles and attain the goal position along the planning path. The reason for this problem is characteristic information of control input space cannot be extracted completely and accurately through uniform sampling and the candidate solutions are not selected within the global scope of solution space, which leads to the nonideal planning path.

3.2. Halton's Sampling. Figure 10 shows the path planning simulation of SBMPC algorithm with Halton's sampling. In Figure 10, the start position is (1, 1) and the goal position is (19, 19). The sample size is 441 at each time k. The total length of planning path in Figure 10 is 26.82 m. Time consumption from start position to goal position is 134.10 s. Figure 11 shows the variation of direction angle θ during the process.

It can be seen from Figures 10 and 11 that the planning path based on SBMPC algorithm with Halton's sampling is not smooth but the variation of direction angle is relatively steady compared with the simulation of uniform sampling. The robot can avoid all the obstacles and attain the goal position along the planning path. Since the distribution distance between any two points of Halton's stochastic sequence is not limited and the sample point cannot be inserted randomly between any two points of uniform sampling sequence, the point position information obtained by Halton's sampling is more comprehensive and the point coverage is more reasonable than uniform sampling when the sample size is same.

3.3. CVT Sampling. Figure 12 shows the path planning simulation of SBMPC algorithm with CVT sampling. In Figure 12, the start position is (1, 1) and the goal position is (19, 19). The sample size is 441 at each time k. The total length of planning path in Figure 12 is 26.18 m. Time consumption from start position to goal position is 130.90 s. Figure 13 shows the variation of direction angle θ during the process.

It can be seen from Figures 12 and 13 that the planning path based on SBMPC algorithm with CVT sampling is more smooth and the variation of direction angle is more steady compared with the simulation of Halton sampling.

As shown in Table 1, fitting performances of sampling points generated by uniform sampling, Halton's sampling, and CVT sampling are compared and analyzed through fitting toolbox in MATLAB software. SSE denotes the sum of squares for error between fitting data and corresponding original data. SSE indicates better fitting performance when it approaches zero. RMES denotes the square root of SSE mean, and R-square denotes the certain coefficient, which

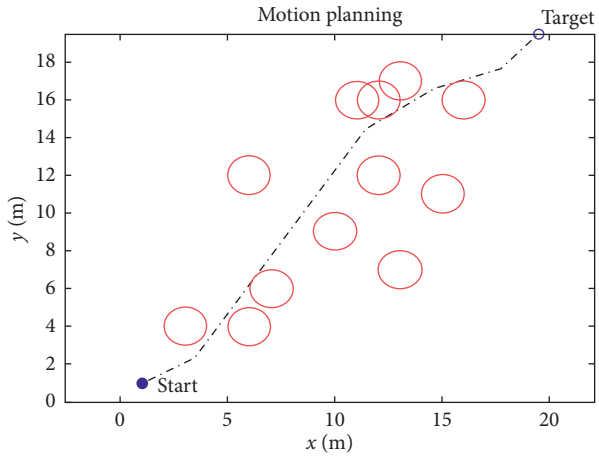

Figure 12: Simulation of SBMPC algorithm with CVT sampling.

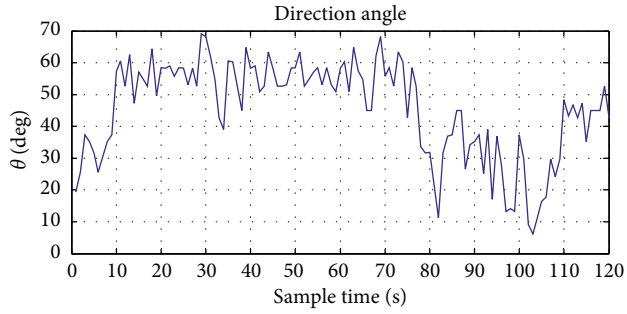

Figure 13: Curve of direction angle while CVT sampling.

Table 1: Fitting performance comparison of three methods.

	Uniform	Halton	CVT
SSE	0.06125	0.04307	0.01518
RMES	0.02144	0.01682	0.01256
R-square	0.9991	0.9994	0.9999

approaches 1 when fitting performance is improved. The plane generated by CVT sampling points is closest to the original plane, and this demonstrates that CVT sampling points can accurately reflect the shape information of a measured plane. Therefore, the performance of SBMPC algorithm is more ideal when the control input variable is sampled through CVT method.

Figure 14 shows the path planning simulation of SBMPC algorithm while the robot is encountering U-shaped obstacle. In Figure 14, the start position is (16.5, 11.5) and the goal position is (14, 27.5). The sample size is 441 at each time k. The total length of planning path in Figure 14 is 34.53 m. Time consumption from start position to goal position is 172.65 s. Figure 15 shows the variation of direction angle θ during the process.

It can be seen from Figures 14 and 15 that the robot can avoid U-shaped obstacle safely and reach the goal position quickly through path planning based on SBMPC algorithm.

Figure 16 shows the path planning simulation of SBMPC algorithm while the robot is encountering cross obstacle. In

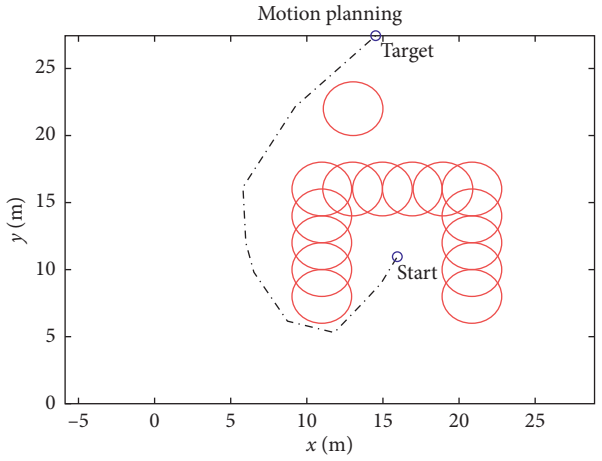

Figure 14: Simulation of SBMPC algorithm with U-shaped obstacle.

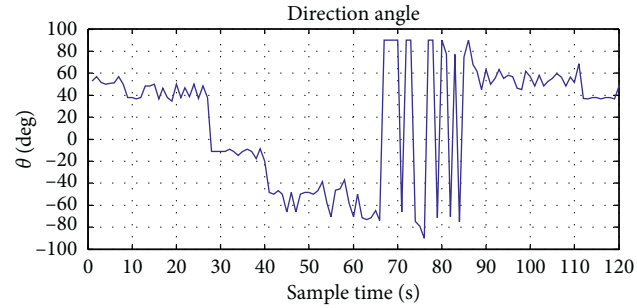

Figure 15: Curve of direction angle while encountering U-shaped obstacle.

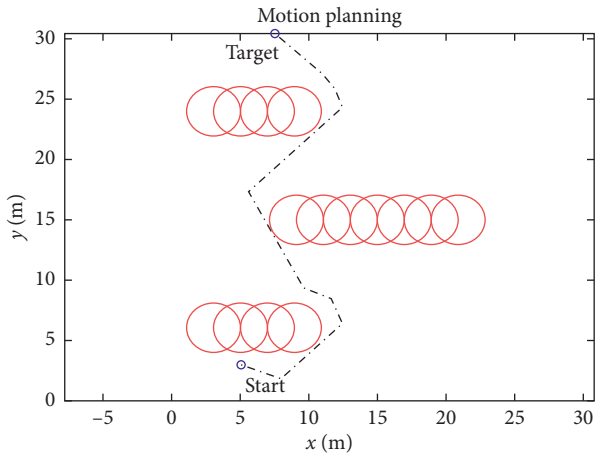

Figure 16: Simulation of SBMPC algorithm with cross obstacle (sample size is 441).

Figure 16, the start position is (5, 2.5) and the goal position is (7.5, 31). The sample size is 441 at each time k. The total length of planning path in Figure 16 is 41.39 m. The time consumption from start position to goal position is 206.94 s. Figure 17 shows the variation of direction angle θ during the process.

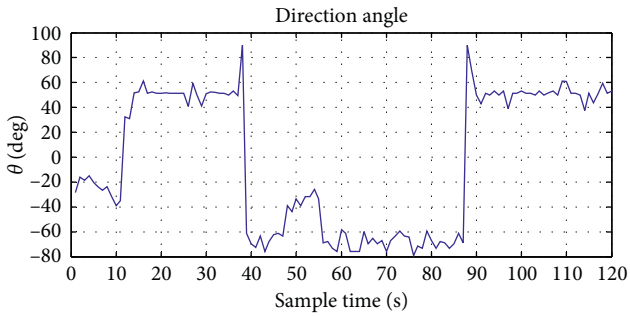

Figure 17: Curve of direction angle while encountering cross obstacle (sample size is 441).

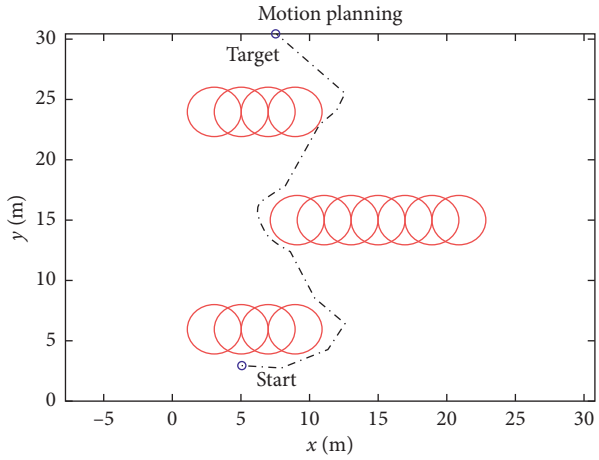

Figure 18: Simulation of SBMPC algorithm with cross obstacle (sample size is 1681).

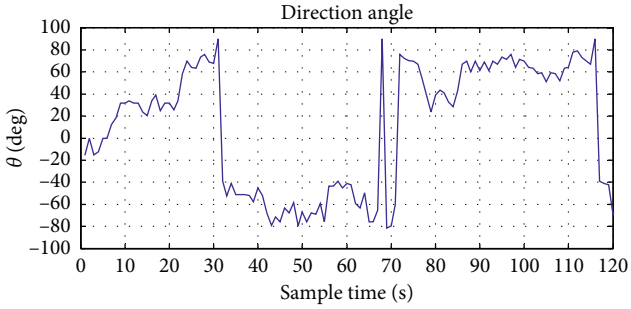

Figure 19: Curve of direction angle while encountering cross obstacle (sample size is 1681).

Figure 18 shows the path planning simulation of SBMPC algorithm while the robot is encountering cross obstacle. In Figure 18, the start position is (5, 2.5) and the goal position is (7.5, 31). The sample size is 1681 at each time k. The total length of planning path in Figure 18 is 40.18 m. Time consumption from start position to goal position is 200.93 s. Figure 19 shows the variation of direction angle θ during the process.

Figures 16 and 18 demonstrate that the robot can avoid cross obstacle safely through path planning based on SBMPC algorithm. The planning path in Figure 18 is more smooth than the planning path in Figure 16. It can be seen from the total length of planning path and time consumption from start position to goal position that path planning performance is improved when the sample size increases.

4. Conclusion

Path planning principle of rescue and detection robot based on SBMPC is mainly analyzed in this paper. SBMPC algorithm and its detailed steps for implementation are specified by analyzing construction of cost function and predictive model.

Simulation results prove the effectiveness of this algorithm when it is applied in the case of encountering intensive obstacles, U-shaped obstacle and cross obstacle. The effects of three sampling methods (viz., uniform sampling, Halton's sampling, and CVT sampling) on path planning performance are compared. Statistical analysis demonstrates that CVT sampling points have the most uniform coverage in two-dimensional plane when the amount of sampling points is the same for the three methods. If path planning of rescue and detection robot is designed through SBMPC algorithm proposed in this paper, the optimal path can be obtained when the control input variable is sampled through CVT method.

In this paper, SBMPC algorithm is mainly used for path planning research and its application of rescue and detection robot. If this algorithm becomes maturer in algorithm structure and process mode, it can also be applied for path planning of domestic service robot and used for life improvement in the future.

Data Availability

The simulation data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this paper.

Acknowledgments

Portions of this work were performed under the projects supported by the Natural Science Foundation of Inner Mongolia, China (Grant no. 2016MS0607), Science Research Foundation of Inner Mongolia University of Technology (Grant no. X201520), and 2019 Talent Development Foundation of Inner Mongolia.

References

[1] X. M. Zhou, Y. B. Gao, and L. W. Guan, "Towards goal-directed navigation through combining learning based global and local planners," *Sensors*, vol. 19, no. 1, pp. 1–20, 2019.

[2] J. Barraquand and J.-C. Latombe, "Robot motion planning: a distributed representation approach," *The International Journal of Robotics Research*, vol. 10, no. 6, pp. 628–649, 1991.

[3] K. Cao, Q. Cheng, S. Gao et al., "Improved PRM for path planning in narrow passages," in *Proceedings of the 2019 IEEE International Conference on Mechatronics and Automation*, pp. 45–50, Tianjin, China, August 2019.

[4] T. Lai, F. Ramos, and G. Francis, "Balancing global exploration and local-connectivity exploitation with rapidly-exploring random disjointed-trees," in *Proceedings of the 2019 IEEE International Conference on Robotics and Automation*, pp. 5537–5543, Montreal, Canada, May 2019.

[5] J.-H. Park and T.-W. Yoon, "Maximizing the coverage of roadmap graph for optimal motion planning," *Complexity*, vol. 2018, Article ID 9104720, 23 pages, 2018.

[6] D. Connell and H. M. La, "Extended rapidly exploring random tree-based dynamic path planning and replanning for mobile robots," *International Journal of Advanced Robotic Systems*, vol. 15, no. 3, pp. 1–15, 2018.

[7] D. R. Nelson, D. B. Barber, T. W. McLain et al., "Vector field path following for small unmanned air vehicles," in *Proceedings of the American Control Conference*, pp. 5788–5794, Minneapolis, MN, USA, June 2006.

[8] J. Yang, X. Wang, S. Baldi, S. Singh, and S. Fari, "A software-in-the-loop implementation of adaptive formation control for fixed-wing UAVs," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 5, pp. 1230–1239, 2019.

[9] S. Farí, X. Wang, S. Roy, and S. Baldi, "Addressing unmodeled path-following dynamics via adaptive vector field: a UAV test case," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, no. 2, pp. 1613–1622, 2020.

[10] D. R. Nelson, D. B. Barber, T. W. McLain, and R. W. Beard, "Vector field path following for miniature air vehicles," *IEEE Transactions on Robotics*, vol. 23, no. 3, pp. 519–529, 2007.

[11] A. Pandey, V. S. Panwar, M. E. Hasan, and D. R. Parhi, "V-REP-based navigation of automated wheeled robot between obstacles using PSO-tuned feedforward neural network," *Journal of Computational Design and Engineering*, vol. 7, no. 4, pp. 427–434, 2020.

[12] A. Pandey, A. K. Kashyap, D. R. Parhi, and B. K. Patle, "Autonomous mobile robot navigation between static and dynamic obstacles using multiple ANFIS architecture," *World Journal of Engineering*, vol. 16, no. 2, pp. 275–286, 2019.

[13] F. Bayat, S. Najafinia, and M. Aliyari, "Mobile robots path planning: electrostatic potential field approach," *Expert Systems with Applications*, vol. 100, pp. 68–78, 2018.

[14] M. A. H. Ali and M. Mailah, "Path planning and control of mobile robot in road environments using sensor fusion and active force control," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2176–2195, 2019.

[15] F. Bayat, "Model predictive sliding control for finite-time three-Axis spacecraft attitude tracking," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 10, pp. 7986–7996, 2018.

[16] Z. Sun, Y. Xia, L. Dai, K. Liu, and D. Ma, "Disturbance rejection MPC for tracking of wheeled mobile robot," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 6, pp. 2576–2587, 2017.

[17] A. T. Hafez, A. J. Marasco, S. N. Givigi, M. Iskandarani, S. Yousefi, and C. A. Rabbath, "Solving multi-UAV dynamic encirclement via model predictive control," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 6, pp. 2251–2265, 2015.

[18] D. C. Fernandez and G. A. Hollinger, "Model predictive control for underwater robots in ocean waves," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 88–95, 2017.

[19] W. Zhou, *Study on the path planning and trajectory tracking of the coal mine detecting and rescuing robot*, Ph.D. dissertation, College of Mechanical and Vehicle Engineering, Taiyuan University of Technology, Taiyuan, China, 2011.

[20] H. Bray, K. Mccormick, R. O. Wells et al., "Wavelet variations on the Shannon sampling theorem," *Biosystems*, vol. 34, no. 1–3, pp. 249–257, 1995.

[21] B. Fu, L. Chen, Y. T. Zhou et al., "An improved A ∗ algorithm for the industrial robot path planning with high success rate and short length," *Robotics and Autonomous Systems*, vol. 106, pp. 26–37, 2018.

[22] M. A. Hasan and M. J. Zaki, "Output space sampling for graph patterns," *Very Large Data Bases*, vol. 2, no. 1, pp. 730–741, 2009.

[23] Y. J. Kanayama and B. I. Hartman, "Smooth local-path planning for autonomous vehicles," *The International Journal of Robotics Research*, vol. 16, no. 3, pp. 263–284, 1997.

[24] R. Song, Y. Liu, R. Bucknall et al., "Smoothed A ∗ algorithm for practical unmanned surface vehicle path planning," *Applied Ocean Research*, vol. 83, pp. 9–20, 2019.

[25] Y. D. Rosita, E. E. Rosyida, and M. A. Rudiyanto, "Implementation of Dijkstra algorithm and multi-criteria decision-making for optimal route distribution," *Procedia Computer Science*, vol. 161, pp. 378–385, 2019.

[26] I. Maurovic, M. Seder, K. Lenac, and I. Petrovic, "Path planning for active SLAM based on the D* algorithm with negative edge weights," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 8, pp. 1321–1331, 2018.

[27] S. Yoon and D. H. Shim, "SLPA*: shape-aware lifelong planning A* for differential wheeled vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 730–740, 2015.

[28] H. Guo, M. Jerrum, and J. Liu, "Uniform sampling through the lovász local lemma," *Journal of the Acm*, vol. 66, no. 3, pp. 1–31, 2019.

[29] B. Robertson, T. Mcdonald, C. Price et al., "Halton iterative partitioning: spatially balanced sampling via partitioning," *Environmental & Ecological Statistics*, vol. 25, no. 8, pp. 1–19, 2018.

[30] L. L. Wang, *Research on the methods for motion planning and path tracking motion control of AUV*, Ph.D. dissertation, Department of Automation, Harbin Engineering University, Harbin, China, 2015.

[31] Z. Chen, T. Zhang, J. Cao, Y. J. Zhang, and C. Wang, "Point cloud resampling using centroidal voronoi tessellation methods," *Computer-Aided Design*, vol. 102, pp. 12–21, 2018.

[32] J. C. Hateley, H. Wei, and L. Chen, "Fast methods for computing centroidal voronoi tessellations," *Journal of Scientific Computing*, vol. 63, no. 1, pp. 185–212, 2015.