Neural Network Approximation-based Near-optimal Motion Planning with Kinodynamic Constraints Using RRT

Yang Li, Rongxin Cui, *Member, IEEE*, Zhijun Li, *Senior Member, IEEE*, and Demin Xu

***Abstract*—In this paper, the problem of near-optimal motion planning for vehicles with nonlinear dynamics in a clustered environment is considered. Based on rapidly-exploring random trees (RRT), we propose an incremental sampling-based motion planning algorithm, i.e., near-optimal RRT (NoD-RRT). This algorithm aims to solve motion planning problems with nonlinear kinodynamic constraints. To achieve the cost/metric between two given states considering the nonlinear constraints, a neural network is utilized to predict the cost function. On this basis, a new reconstruction method for the random search tree is designed to achieve a near-optimal solution in the configuration space (C-space). Rigorous proofs are presented to show the asymptotical near-optimality of NoD-RRT. Simulations are conducted to validate the effectiveness of NoD-RRT through comparisons with typical RRT and kinodynamic RRT*. In addition, NoD-RRT is demonstrated in an experiment using a Pioneer3-DX robot.**

***Index Terms*—Motion planning, Neural networks, Sampling-based algorithms, Kinodynamic constraints.**

I. Introduction

In recent years, robot motion planning problems have received a considerable amount of research attention because of their important applications, such as civilian search/rescue, autonomous exploration, storehouse management, assembly planning, bioengineering, and so forth [1]–[3]. Motion planning is one of the basic issues in industrial processes since most industrial production processes need robotic or mechanical assistance. Robots, vehicles and machines can operate more safely with a good path or motion planning, such as autonomous guided vehicles (AGVs) in automated storage and retrieval systems (ASRS), autonomous robots in assembly processing, mechanical arms in chip production, and so on. In addition, motion planning is also important in industrial processes because of energy optimality or time optimality or because a combination of robots and vehicles is generally considered [4]–[7]. In [4], [5], energy-optimal and time-optimal trajectory generation methods are proposed, respectively. In [6], nonuniform environments are considered during the path planning of unmanned vehicles.

In general, robot motion planning problems can be described as finding a sequence of control inputs that drive the robot from its initial state to its goal state while avoiding obstacles in a complex environment [8]. Sampling-based algorithms, such as RRT and probabilistic roadmaps (PRM) [9], have achieved remarkable success in solving motion planning problems, particularly with a high-dimensional C-space. RRT is a sampling-based algorithm that exhibits highly effective performance in high-dimensional configuration spaces. This algorithm is able to explore and search the configuration space quickly. In addition, RRT can address much broader nonlinear systems, eventually determining a valid path in a complex environment when enough samples are made. Due to the efficiency of RRT, it is widely used in many applications, such as nonholonomic wheeled robots, surgical planning, search and rescue, and so forth. RRT constructs a random search tree in a robotic C-space, and then it incrementally extends the search tree toward the sampling states that are uniformly generated according to some given sampling strategy, such as the goal bias method. RRT has provably probabilistic completeness, i.e., the probability that RRT returns an available solution from the initial state to the goal state is almost surely one as the number of samples tends to infinity if a solution exists. PRM is a multiple-query method that contains all available path sets. Then, the best path is searched from the initial state to the goal state set. In [3], samples are generated in a low-dimensional manifold of high-dimensional C-space such that the connectivity of states is easier to identify. This method is provably more efficient than PRM with a six-dimensional C-space.

RRT has been proven to be asymptotically non-optimal [10]. To achieve optimality, several other sampling-based path planning algorithms have been proposed, such as PRM* and RRT*. RRT* is a typical extension of RRT, which has been applied in several applications [11]–[13]. The solution returned by RRT* converges to optimality when the number of samples tends to infinity. The reason for the asymptotical optimality of RRT* primarily lies in the post processing, i.e., the reconstruction of the search tree. It checks each *near node* neighboring with a new sample, and it reconstitutes the local edges to decrease

Manuscript received June 20, 2017; revised August 18, 2017 and September 28, 2017; accepted March 02, 2018. This work was supported in part by the National Natural Science Foundation of China (NSFC) under grant 61472325, grant 61633002, grant 61751310 and grant 61625303. *(Corresponding author: Rongxin Cui.)*

Y. Li, R. Cui and D. Xu are with the School of Marine Science and Technology, Northwestern Polytechnical University, Xi'an 710072, China. (e-mail: r.cui@nwpu.edu.cn).

Z. Li is with the College of Automation Science and Engineering, South China University of Technology, Guangzhou 510641, China. (e-mail: zjli@ieee.org).

the cost based on the metric between the new sample and each node in the *near node set*. Other asymptotic optimal motion planning methods, such as transition-based RRT* (T-RRT*) and anytime T-RRT (AT-RRT), have also been proposed. In [14], the edges of the search tree are evaluated in the form of discrete cost based on mechanical work. In this way, paths are efficiently planned in a cost space.

There are two important issues from which RRT* suffers when kinodynamic constraints are considered, namely, the local boundary value problem (BVP) and the metric computation problem. During the process of reconstruction in RRT*, a new edge is added to the search tree once it has a potentially lower cost. Due to the kinodynamic constraints, the replacement can be described as a local BVP problem with two states in C-space. Although a BVP solver can provide the optimal or sub-optimal solutions, it is difficult to address the problem in high-dimensional C-space.

A new extension of RRT*, called kinodynamic RRT*, was proposed to address the BVP problem with an optimal controller for linear dynamics [15]. It estimates the fixed final time based on the optimal control policy, and then it computes the local optimal trajectory. This method requires the robot dynamics to be holonomic, and it is still computationally expensive. SPARSE-RRT adopts a *sparse structure* in a forward propagation to replace the BVP solver [16]. The key idea involves a drain of nodes to select a better cost with other new sampling states. In contrast to other sampling-based algorithms, it avoids the steer functions with a sparse data structure, which is adapted to minimize the number of propagations per iteration. The asymptotic completeness and optimality of SPARSE-RRT were also proven in [16]. In [11], a novel motion planning algorithm called RRT$^{\#}$ (RRT "sharp") was proposed. RRT$^{\#}$ combines the concept of lifelong planning A* (LPA*) to guarantee that the solutions contain the optimal one if at least one node of the current tree is within the goal region. In [17], an asymptotically near-optimal RRT, named lower bound tree-RRT (LBT-RRT), was proposed based on RRT* and rapidly-exploring random graph (RRG). This algorithm returns high-quality solutions and uses less running time with a slight trade-off in optimality. RRT$^{\mathrm{X}}$ was described as the first sampling-based replanning algorithm in [18] and was proven to be asymptotical. Moreover, this algorithm performs well when environment changes are detected. This result indicates that RRT$^{\mathrm{X}}$ has good properties in both dynamic environments and static environments.

In [19], the motion planning problem for aerial vehicles is considered as an optimal control problem, i.e., a constrained two-point BVP problem. A potential field is designed to generate the new sample. However, sampling-based algorithms adopt a collision detector other than the explicit representation of the obstacles and build a search tree in obstacle-free robotic C-space. The BVP problem is also avoided due to its forward propagation. In [20], the integration of the system dynamics is computed in the form of spline curve parameterization. In this approach, the differential constraints are always satisfied when the connection/edge between two states in robotic C-space are generated. On this basis, RRT can work well with kinodynamic constraints in finite runtime. However, the performance of this algorithm is sensitive to the choice of the metric in C-space.

The metric of two states in C-space is generally regarded as the effort between them, such as total length, time-to-go, or another combination. A suitable metric can provide a more accurate estimation of the cost. In RRT, the metric is defined as the Euclidean distance or weighted Euclidean distance between the two states in a C-space. Because the reconstruction is sensitive to the metric, the weights have a substantial influence on the convergence efficiency.

Meanwhile, several new types of metrics are approximated based on the finite-horizon linear quadratic regulation (LQR) in [21]. It has good performance with linear systems, as well as with nonlinear systems by linearizing the dynamics. In each iteration, the differential Riccati equation needs to be integrated, which actually has high computational complexity. Dubins' path is designed as the metric in [19], which is not easy to extend to other applications. In [22], the travel time is considered due to the finite energy of a solar-powered robot. The unique feature in [19] is that the environment includes additional solar energy in a known density distribution.

Neural networks (NNs) are widely applied in many fields, such as data classification, controller design and black box approximation, among others [23]–[27]. Specifically, NNs have remarkable efficiency for approximation problems. In [28], an adaptive NN is used for performance index approximation. In [29], a multivariate max-product NN is designed for uniform approximation, which is proven to be more accurate. NN approximation-based adaptive control of a discrete time system is presented in [30]. Due to the approximation ability of NNs, we design NNs to approximate the metric in C-space. Compared with the LQR method, NNs are considerably easier in terms of computation. Moreover, the training of NNs can be completed prior to motion planning without considering any environment constraints. Without loss of generality, the cost function can be defined as the total length of the path in this work. Specifically, NNs are utilized to approximate the ratio of the optimal cost of two nodes and corresponding configuration states, based on which the cost function estimation can be derived. The main contributions of this paper can be summarized as follows:

(i) A neural network is designed to approximate the cost function considering nonlinear kinodynamic constraints.
(ii) A new reconstruction method for the random search tree is proposed to guarantee asymptotic optimality.
(iii) The proposed NoD-RRT algorithm can return a valid path within a reasonable time for nonlinear dynamic systems that have an unclosed-form optimal solution. Compared with RRT and kinodynamic RRT*, this algorithm is more effective in terms of computational complexity when handling nonlinear systems.

The remainder of this paper is organized as follows. In section II, the problems, definitions, and notations are described. In section III, the body of NoD-RRT and its main sub-algorithms are presented in detail. Moreover, a rigorous proof is presented in section IV. In section V, several simulations and one real experiment are presented to show the effectiveness of NoD-RRT. Conclusions are drawn in section VI.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2018.2816000, IEEE Transactions on Industrial Electronics

II. Problem Formulation and Notation

A. Notation

Let $\mathcal{X} \subseteq \mathbb{R}^n$ be the $n-$dimensional robotic C-space. $\mathcal{X}_{obs}$ denotes the obstacle region, and it is also a closed subset of $\mathcal{X}$. Thus, the free configuration space can be derived as $\mathcal{X}_{free} = \mathcal{X} \setminus \mathcal{X}_{obs}$. It is clear that $\mathcal{X}_{free}$ is an open subset of $\mathcal{X}$. Let $x \in \mathcal{X}$ denote the robotic configuration state. x_{init} and $\mathcal{X}_{goal}$ are the initial configuration and the goal region, respectively. Let $\mathcal{G} = (V, E)$ denote the spanning tree, where V is the vertex set in robotic C-space and E is the edge set. Given any $v \in V$ in $\mathcal{G}$, successors of v are the vertices that can be visited from v. Meanwhile, the predecessor vertex of v denotes the vertex from which $v \in V$ is visited adjacently.

In this paper, the dynamics of a robot is described as

$$\dot{x} = f(x, u) \tag{1}$$

where $u \in \mathcal{U}$ is the piecewise control vector.

It is assumed that f is smooth with respect to x and u. A valid trajectory is defined as function $\pi(t) : [0, s] \to \mathcal{X}_{free}$, where $\pi(0) = x_1, \pi(s) = x_2$. Meanwhile, the corresponding control input sequence is denoted as $u_\pi(t) : [0, s] \to \mathcal{U}$, which drives the robot from x_1 to x_2. In practice, the rigid robot body must maintain a secure distance from obstacles, also called *clearance distance*. Thus, we define the $\delta-$clearance trajectory in robotic configuration space as follows.

Definition 1 *[10] [δ-clearance Trajectory] Given trajectory $\pi : \mathbb{R} \to \mathbb{R}^n$, π is δ-clearance if for any $x_{obs} \in \mathcal{X}_{obs}$, $|\pi(t) - x_{obs}| \geq \delta$.*

This definition is also typically called strong δ-clearance. If there exists $0 < \delta_a \leq \delta$ such that $\pi(t)$ has δ_a-clearance and there exists a continuous transformation from $\pi(t)$ to $\pi'(t)$ with strong δ-clearance, we say that $\pi(t)$ has weak δ-clearance.

Let $c(\pi) : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ denote the cost function of $\pi(t)$:

$$c(\pi) = \int_0^s [\pi(t)^\top Q \pi(t) + u_{\pi(t)}^\top R u_{\pi(t)}] \mathrm{d}t \tag{2}$$

where Q and R are positive matrices. In general, the Euclidean distance is chosen when $\pi(t)^\top Q \pi(t) + u_{\pi(t)}^\top R u_{\pi(t)} = 1$.

In this work, we select the cost function as the length of the path without loss of generality, i.e., $c(\pi(t)|t \in [0, s]) = \int_0^t \mathrm{d}t$. Let us denote $c(\pi(t_1), \pi(t_2)) = \int_{t_1}^{t_2} \mathrm{d}t$ as the cost of the path from $x_1 = \pi(t_1)$ to $x_2 = \pi(t_2)$. For brevity, we denote $c(\pi(t_1), \pi(t_2))$ as $c(x_1, x_2)$. The cost function is generally more complex in real applications, which even results in the computation being intractable. In this work, we utilize NNs (see section III-A) to address any cost described by (2).

B. Problem Formulation

In this section, we formally present the definitions of the problems considered in this work. The basic path planning problem can be defined as follows.

Problem 1 (Basic Path Planning Problem [9]) *Given an initial state x_{init} and a goal region $\mathcal{X}_{goal}$, find a trajectory $\pi(t) : [0, s] \to \mathcal{X}_{free}$ such that $\pi(0) = x_{init}$ and $\pi(s) \in \mathcal{X}_{goal}$.*

Kinodynamic constraints always exist in real applications. Thus, we define the δ-clearance motion planning problem with kinodynamic constraints as follows.

Problem 2 *Given an initial state x_{init} and a goal region state set $\mathcal{X}_{goal}$, find a trajectory $\pi(t) : [0, s] \to \mathcal{X}_{free}$ and the corresponding control input function $u(t) : [0, s] \to \mathcal{U}$ such that $\pi(t)$ is derived from $u(t)$ and is $\delta-$clearance, which satisfies $\pi(0) = x_{init}$, $\pi(s) \in \mathcal{X}_{goal}$.*

Let us denote the existing available solution set for Problem 2 as Π, and denote $\pi^* \in \Pi$ as the optimal path. In most applications, the objective involves the optimality of the path and the efficiency of the algorithms. We define the asymptotic near-optimality property as follows.

Definition 2 (Asymptotic Near-optimality of Trajectories) *Given the problem described in Problem 2 and the cost function of trajectories $c(\pi)$, an algorithm possesses asymptotic near-optimality if it returns the solution π, whose cost is no more than $(1 + \epsilon)\pi^*$ for any small $\epsilon > 0$, with probability of one when the number of samples tends to infinity.*

III. Main Algorithms

In this section, the proposed NoD-RRT algorithm is described in detail. The NN approximation method is outlined first. Then, NoD-RRT is introduced as a new incremental sampling-based algorithm for Problem 2.

A. Neural Network Approximation for Cost Function

In RRT*, the robot is viewed as a *free-flying* node in C-space. Each obstacle-free edge $e = (x_i, x_j)$, namely, the straight line from x_i to x_j, is regarded as available for robots. This is restricted for real applications with kinodynamic constraints, such as minimum turning radius. It also leads to the fact that the path length between two nodes cannot be calculated in the form of Euclidean distance, as shown in Fig. 1.

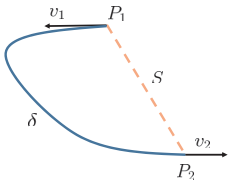


Fig. 1. Considering the robotic kinodynamic constraints, the robot cannot move exactly along the straight line from P_1 to P_2, and one of its acceptable paths is shown as the curve from P_1 to P_2.

The estimation of the real optimal or near-optimal cost is more important for whether and how the new node is added to the search tree, as well as the process of reconstructing the search tree. Let us define the relationship between the cost function and the Euclidean distance with kinodynamic constraints as

$$\mathscr{L}(x_i, x_j) = \frac{c(x_i, x_j) - \rho(x_i, x_j)}{\rho(x_i, x_j)} \tag{3}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2018.2816000, IEEE Transactions on Industrial Electronics

where $\rho(x_i, x_j)$ is the Euclidean distance from x_i to x_j. It is clear that $\mathscr{L} \in [0, 1]$ because $c(x_i, x_j) \leq \rho(x_i, x_j)$. Since the approximation error returned by NNs must exist, we calculate the cost function indirectly if $\mathscr{L} \notin [0, 1]$.

To estimate the unknown function $\mathscr{L}(x_i, x_j) : \mathbb{R}^3 \times \mathbb{R}^3 \rightarrow \mathbb{R}$, we introduce the two-layer feed-forward NN with sigmoid hidden neurons and linear output neurons $\mathcal{N}(x) : \mathbb{R}^3 \rightarrow \mathbb{R}$ as

$$\mathcal{N}(x) = \phi S(\omega, \mathrm{In}(x))^\top \tag{4}$$

where $\mathrm{In}(x) = [b, \theta_s, |v|_2, \rho(x_i, x_j)]^\top \in \mathbb{R}^3$ is denoted as the input vector and the weights ϕ and ω are denoted as $\phi = [\phi_1, \phi_2, \ldots, \phi_l]^\top$ and $\omega = [\omega_{i,j}]_{3\times l}$, respectively. The structure of the NN is shown in Fig. 2.

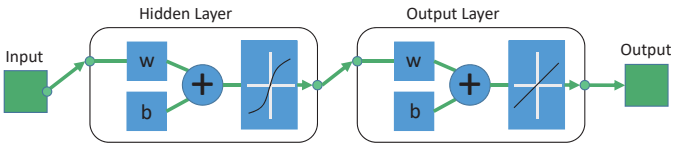
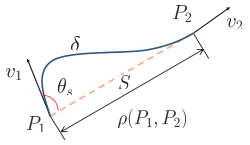
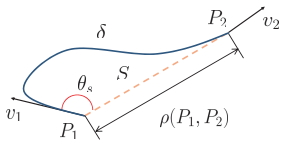


Fig. 2. Structure of the NN.

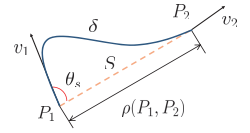
Each element in the input vector of the NN is described as in Fig. 3(a). $\theta_s \in [0, \pi]$ is the absolute value of the angle between the velocity vector of x_i and the straight line through x_i to x_j. $|v|_2 \in [0, v_{max}^2]$ is the Euclidean 2-norm of the velocity vector of x_i, where $v_{\max}$ is the maximum of velocity.



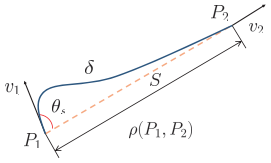
(a) Input vector of NN.



(b) $\mathscr{L}(x_1, x_2)$ mostly increases as θ_s grows.



(c) $\mathscr{L}(x_1, x_2)$ mostly increases as $|v|_2$ increases.



(d) $\mathscr{L}(x_1, x_2)$ mostly decreases as $\rho(x_1, x_2)$ increases.

Fig. 3. Choice for the inputs of NNs and corresponding influences.

Intuitively, all of these three elements have an influence on $\mathscr{L}(x_i, x_j)$, as shown in Fig. 3(b)- Fig. 3(d). Increasing θ_s and $|v|_2$ will clearly lead to an increase in $\mathscr{L}$. Conversely, the Euclidean distance reduces the impact of the kinodynamic constraints. Moreover, $\mathscr{L}$ will tend to 0 when $\rho \rightarrow \infty$. Once we obtain $\mathscr{L}(x_i, x_j)$ after training the NN, cost function $c(x_i, x_j)$ can be approximated as $c'(x_i, x_j) = [1 + \mathscr{L}(x_i, x_j)]\rho(x_i, x_j)$.

The training data for the NN are generated using an offline process. As shown in Fig. 4, a new state x_2 is randomly uniformly generated in robotic C-space. Thus, x_1 is set to be the origin of the plane. Meanwhile, the velocity of x_1 is also randomly selected in $[0, v_{max}]$. Then, the path from x_1 to x_2 is calculated according to the robotic controller. Ideally, the optimal segment path from x_1 to x_2 can be generated based on a linear controller [31] or an optimal controller [32]. However, it is inappropriate for our problems due to the computational cost of the Riccati equation. Here, we use a simple but effective PD controller to drift forward x_2. For testing the training results, we randomly divided the data into 70%, 15% and 15% for training, validating and testing, respectively. For brevity, we use the Levenberg-Marquardt (LM) algorithm [33] to update the weights of the NN.

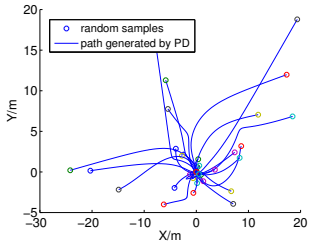


Fig. 4. New states are randomly generated in robotic configuration space. Paths are calculated based on the controller, e.g., PD controller used in this work.

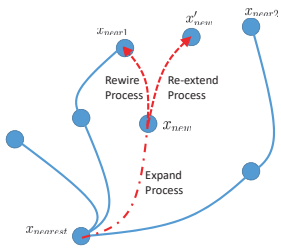


Fig. 5. Reconstruction process: red dot dash line shows the expand process toward x_{new}. x_{near1} is rewired from x_{new}. Green dot line will be removed due to a higher cost. x_{near2} is re-extended, in which x'_{new} is generated with a potentially lower cost.

B. NoD-RRT

Considering the nonlinear robot dynamics described by (1), the main body of the proposed NoD-RRT is described in Algorithm 1. This algorithm can be viewed as a new version of the RRT* algorithm, based on which a new method of reconstruction is proposed to address kinodynamic constraints.

At iteration k, a new sample x_{rand} is generated according to a uniform distribution on C-space (Algorithm 1, Line 3). Indeed, x_{rand} needs to be utilized to expand the search tree in C-space. In typical RRT, new edges are generated toward x_{rand} from the nearest node $x_{nearest}$ that has the least distance from x_{rand}. Due to the kinodynamic constraints, a "distance" estimation is achieved by the NNs trained in section III-A (Algorithm 1, Line 4).

To maintain the asymptotic optimality property, reconstruction must be called after x_{new} is added to $\mathcal{G}(V, E)$. Reconstruction in NoD-RRT includes two parts: the **Re-extend** function and the **Rewire** function. Fig. 5 shows the reconstruction process.

First, the near vertex set X_{near} is obtained using the **Near** function (Algorithm 1, Line 7), and then it is determined whether trajectories with less cost exist. For each vertex $x' \in \mathcal{X}_{near}$, the x'_{parent} of x_{new} is replaced with x' once the cost function of x_{new} is decreased. Due to the unknown local optimal trajectory from x' to x_{new}, it is always impractical to calculate the accurate cost. In contrast to typical RRT*, NoD-RRT *re-extends* toward the x_{new} region from x'. In other words, a new vertex x'_{new} is generated by applying the controller on x', which attempts to re-explore from x'. Note

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2018.2816000, IEEE Transactions on Industrial Electronics

Algorithm 1 Body of Nod-RRT.

$V \leftarrow (x_{init})$; $E \leftarrow \emptyset$;
for $k = 1 \rightarrow K$ **do**
  $x_{rand} \leftarrow$ **SampleFree**$(\mathcal{X})$;
  $x_{nearest} \leftarrow$ **Nearest**$(\mathcal{G} = (V, E), x_{rand})$;
  $x_{new} \leftarrow$ **Steer**$(x_{nearest}, x_{new})$;
  if **CollisionFree**$(x_{nearest}, x_{new})$ **then**
    $X_{near} \leftarrow$ **Near**$(\mathcal{G} = (V, E), x_{new}, \min\{\gamma^*(\frac{\log k}{k})^{\frac{1}{d}}, \eta\})$;
    $V \leftarrow V \bigcup \{x_{new}\}$;
    $c_{\min} \leftarrow$ **Cost**$(x_{nearest})$ + **Cost**$(\overline{x_{nearest}, x_{new}})$;
    $x_{\min} \leftarrow x_{nearest}$;
    for each $y \in X_{near}$ **do**
      if **CollisionFree**(y, x_{new}) && **Cost**$(\mathcal{G} = (V, E), y)$ + **Cost**$(\overline{y, x_{new}}) < c_{\min}$ **then**
        Re-extend(y, x_{new});
        $c_{\min} \leftarrow$ **Cost**$(\mathcal{G} = (V, E), y)$ + **Cost**$(\overline{y, x_{new}})$
      end if
      $E \leftarrow E \bigcup (x_{min}, x_{new})$
    end for
    for each $y \in X_{near}, x_{new} \in X_{new}$ **do**
      if **CollisionFree**(x_{new}, y) && **Cost**$(\mathcal{G} = (V, E), y)$ + **Cost**$(\overline{x_{new}, y}) < c_{\min}$ **then**
        Rewire(x_{new}, y);
      end if
    end for
  end if
end for
return $\mathcal{G} = (V, E)$;

Algorithm 2 Re-extend Function

input $\leftarrow$ **control_policy**(x_{near}, x_{new});
$x'_{new} \leftarrow$ **Driven**(x_{near}, input)
add_vertex$(\mathcal{G} = (V, E), x'_{new})$
$x'_{new}.parent \leftarrow x_{near}$
$x_{near}.u \leftarrow$ input

that if the cost estimation of the trajectory from x' is optimal rather than its prior parent vertex, then it is worth extending the search tree toward x_{new} from x'.

The important advantage of the **Re-extend** function is the avoidance of the complicated local nonlinear BVP problem. It is only required after collision detection compared with other sampling-based algorithms, which certainly leads to a reduction in the computational complexity. In RRT*, the useless edge $e = (x' \in X_{near}, x_{new})$ is added, and $e = (x_{nearest}, x_{new})$ is deleted. Considering kinodynamic constraints, however, it is complicated for calculation.

Meanwhile, the reconstruction is also performed by checking the edge set $E = (x_{new}, x' \in X_{near})$. If the cost of $x' \in X_{near}$ decreases when the parent vertex is replaced with x_{new}, then NoD-RRT attempts to set the new edge $e = (x', x_{new})$. Considering the kinodynamic constraints, however, the local BVP problem still exists. Thus, we propose a new method in the **Rewire** function. As shown in Algorithm 3, **Re-extend** is called to extend the search tree toward x' from

Algorithm 3 Rewire Function

input $\leftarrow$ **control_policy**(x_{new}, x_{near});
x'_{near}**(Driven)**(x_{new}, input);
if **Cost**$(x'_{near}, x_{near}) < \epsilon$ **then**
  Delete$(e = (x_{near}.parent, x_{near}))$;
  Delete$(x_{near}.parent.u)$;
else
  add_vertex$(\mathcal{G}(V, E), x'_{near})$;
  $x'_{near}.parent \leftarrow x_{new}$;
  $x_{new}.u \leftarrow$ input
end if

x_{new} if $x' \in \mathcal{X}_{near}$ is the leaf vertex. In other situations, a new state is obtained by applying the control returned by the same policy, and this new state is denoted as x'_{new}. If x'_{new} and x' are almost identical, i.e., **Cost**$(x'_{new}, x') < \epsilon$, then the parent vertex of x' is changed to x_{new}. Otherwise, x'_{new} is added to the search tree.

IV. Analysis

A. Probabilistic Completeness

The probabilistic completeness property of an algorithm is stated in [10] considering the feasible problem. RRT has provably probabilistic completeness, as shown in Lemma 1.

Lemma 1 (Probabilistic Completeness of RRT [34]) *Consider Problem 1; there exists an available path from x_{init} to $\mathcal{X}_{goal}$ and a constant $\eta > 0$ such that the probability that RRT returns failure is at most $\mathrm{e}^{-\eta n}$. Additionally, the probability will almost surely be zero when n tends to infinity.*

The probabilistic completeness of RRT is achieved by a k-length attraction sequence $\{\mathcal{A}_0, \ldots, \mathcal{A}_k\}$, which combines the initial state x_{init} and the goal state set $\mathcal{X}_{goal}$. The attraction sequence $\{\mathcal{A}_i\}$ can intuitively be viewed as a random trail that guides the robot toward the goal if a *basin* set $\mathcal{B}_i \supseteq \mathcal{A}_i$ can be found for each $\mathcal{A}_i$ and two conditions are satisfied: 1. given the metric, each vertex of A_i is closer to each element (vertex) of A_{i+1} than any vertex out of $\mathcal{B}_i$, and 2. if a vertex in $\mathcal{B}_i$ is chosen for extension, a feasible input control sequence will be selected to eventually drive the robot to $\mathcal{A}_i$.

According to [34], the probability that the random search tree of RRT has at least one vertex in the goal state set can be considered as a binomial distribution. Moreover, the upper bound of the probability that RRT fails to find a feasible path is given. Similarly, the probabilistic completeness of NoD-RRT can also be achieved.

Theorem 1 *[Probabilistic Completeness of NoD-RRT] Consider Problem 1 and given $\mathcal{X}_{free}$ and $\mathcal{X}_{goal}$, there exists constants $\eta > 0$ and integer $n_0 > 0$ such that*

$$\mathbb{P}(\textit{NoD-RRT finds a solution in} \mathcal{X}_{free}) > 1 - \mathrm{e}^{-\eta n}, \ \forall n > n_0 \tag{5}$$

Proof: First, it is reasonable to consider that the sampling process of NoD-RRT generates the same vertices when the parameters in NoD-RRT are selected the same as in RRT, such

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2018.2816000, IEEE Transactions on Industrial Electronics

as the goal bias rate. It means that $V_{NoD-RRT}(n) \supseteq V_{RRT}(n)$ for all $n \in \mathbb{N}$. Meanwhile, NoD-RRT absolutely returns a connected graph.

We utilize the attraction sequence $\{\mathcal{A}_i | i = 0, 1, \ldots, k\}$ to show the probabilistic completeness of NoD-RRT. The attraction sequence is generated in the same way as that in RRT, and $\mathcal{A}_0 = x_{init}$, $\mathcal{A}_k = \mathcal{X}_{goal}$. For each $\mathcal{A}_i$, the *basin* set $\mathcal{B}_i$ is selected such that the two constraints are satisfied. Specifically, the second constraint of $\mathcal{B}_i$ needs to consider the kinodynamic constraints of the robot. In NoD-RRT, it is assumed that the controllers are designed to be asymptotically stable. According to Lemma 1, if the attraction sequence exists, then the probability that RRT with asymptotically stable controllers fails to find a solution under kinodynamic constraints tends to zero as the number of samples increases. Because $V_{NoD-RRT}(n) \supseteq V_{RRT}(n)$, the probabilistic completeness of NoD-RRT can be achieved directly from the results of RRT. ■

B. Asymptotic Near-Optimality of NoD-RRT

Asymptotic optimality of a planning algorithm is important in the case of sufficient planning time. In real applications, however, the planning time is also a factor of efficiency for algorithms. An acceptable pattern is intuitively designed to find a solution that has enough "good" property in the least possible planning time. Thus, the near-optimality is proposed as a trade-off between optimality and running efficiency.

Definition 3 (Asymptotic Near-optimality) *Given the planning problem $(\mathcal{X}_{free}, x_{init}, \mathcal{X}_{goal})$ and cost function $c(\pi)$, the planning algorithm is asymptotically optimal if for any $\epsilon > 0$,*

$$\mathbb{P}(\{\lim_{n\to\infty} \sup c(\pi_n) = (1+\epsilon)c^*\}) = 1 \tag{6}$$

Definition 4 *[Asymptotically Stable Lyapunov-based Controller] Denote the equations of motion for the system as $\dot{x} = f(x, u, t)$ and one of the corresponding solutions as $x^*(u^*, t)$, a small positive real number ϵ; then, there exists $\varrho(\epsilon, t_0) > 0$ such that $||x(t) - x^*(t)|| \leq \epsilon$ if $||x(t_0) - x^*(t_0)|| \leq \varrho$.*

In this paper, it is assumed that the robot controller is asymptotically Lyapunov stable, i.e., the robot can follow the path if it exists.

Let σ^* denote the optimal path from x_{init} to $\mathcal{X}_{goal}$. Let $\{\delta_n\}_{n\in\mathbb{N}}$ denote a sequence of positive real numbers and $\lim_{n\to\infty} \delta_n = 0$. Let $\{\sigma_n\}_{n\in\mathbb{N}}$ denote a sequence of paths such that σ_n has strong δ_n-clearance. We will show the existence of $\{\sigma_n\}_{n\in\mathbb{N}}$ in Lemma 2.

Lemma 2 *[Existence of $\{\sigma_n\}_{n\in\mathbb{N}}$] Given that σ^* with corresponding control u^* has weak $\delta-$clearance, sequence $\{\delta_n\}_{n\in\mathbb{N}}$ satisfies $\lim_{n\to\infty} \delta_n = 0$ and $0 < \delta_n \leq \delta$; then, there exists a sequence of paths $\{\sigma_n\}_{n\in\mathbb{N}}$ such that $\lim_{n\to\infty} \sigma_n = \sigma^*$ and σ_n has δ_n-clearance.*

Proof: Motivated by the proof of Lemma 50 in [10], we define a sequence $\{\mathcal{X}_n\}_{n\in\mathbb{N}}$ in the form of $\mathcal{X}_n := \mathrm{cl}(\mathrm{int}_{\delta_n}(\mathcal{X}_{\mathrm{free}}))$, where $cl(\mathcal{A})$ denotes the closure of set $\mathcal{A}$. Note that any node in $\mathcal{X}_n$ has a distance of at least σ_n from any node in obstacles. Because of the weak $\delta-$clearance of σ^*, function $\kappa : [0, 1] \to \Sigma_{free}$ exists such that $\kappa(0) = \sigma^*$. The path sequence is defined as $\beta_n := \min_{\beta\in[0,1]}\{\beta | \kappa(\beta) \in \Sigma_{free}\}$, $\sigma_n := \kappa(\beta_n)$, as shown in Fig. 6.

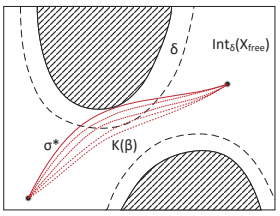


Fig. 6. Path sequence $\{\sigma_n\}_{n\in\mathbb{N}}$, i.e., $\{\kappa(\beta)|\beta \in [0, 1]\}$ can be considered as a homotopy sequence of σ^* to approach σ^* with strong δ_n-clearance.

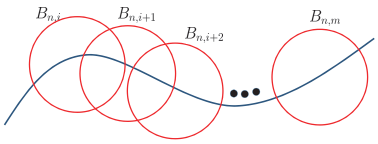


Fig. 7. Ball sequence B_n is constructed to cover all of σ_n.

Because any node of σ_n has a distance of at least σ from any node in $\mathcal{X}_{obs}$ and $\delta_n \leq \delta$, it is clear that σ_n has strong δ_n-clearance.

Meanwhile, $\forall\beta \in (0, 1]$, $\exists\delta_\beta > 0$ such that $\kappa(\beta)$ has strong $\delta_\beta-$clearance. Thus, because $\lim_{n\to\infty}$, we obtain $\lim_{n\to\infty} \beta_n = 0$, which implies that $\lim_{n\to\infty} \sigma_n = \sigma^*$. This completes the proof. ■

We construct the ball sequence along σ^* as $\{B_{n,1}, B_{n,2}, \ldots, B_{n,m}\}$. The center of each ball is located at $\sigma(t), t \in [0, 1]$, and they are at least $\delta_n\varepsilon/(1+\varepsilon), \varepsilon \in (0, 1)$, apart from each other as shown in Fig. 7. The radius of each ball is also defined as $\frac{\delta_n}{1+\varepsilon}$.

Lemma 3 *If $\gamma^* > 3(\frac{1}{d})^{\frac{1}{d}}(\frac{\mu(\mathcal{X}_{free})}{\zeta_d})^{\frac{1}{d}}$, then there exists a positive constant $0 < \varepsilon < 1$ such that for sufficiently large n, each ball in B_n contains at least one vertex.*

Proof: This is a direct conclusion of Lemma 52 in [10]. The main difference is the radius of the reconstructed ball. Let F_n and F_n^c denote the event that each ball of B_n contains at least one vertex and its complement set, respectively. Meanwhile, let $F_{n,i}^c$ denote the event that ball $B_{n,i}$ contains no vertices. The bound of the probability of $F_{n,i}^c$ can be calculated as

$$\mathbb{P}(F_{n,i}^c) = \left(1 - \frac{\zeta_d}{\mu(\mathcal{X}_{free})}(\frac{\gamma^*}{2+\varepsilon})^{\frac{1}{d}}\frac{\log n}{n}\right)^n \tag{7}$$

Due to the inequality $(1 - 1/f(n))^r \leq \exp^{-r/f(n)}$, $\mathbb{P}(F_{n,i}^c)$ can be bounded as $\mathbb{P}(F_{n,i}^c) \leq n^{-\frac{\zeta_d}{\mu(\mathcal{X}_{free})}}$.

Thus, $\mathbb{P}(F_n^c)$ can be bounded as

$$\mathbb{P}(F_n^c) = \mathbb{P}(\bigcup_{i=1}^{M} F_{n,i}^c) \leq Cn^{\left(-\frac{\zeta_d}{\mu(\mathcal{X}_{free})}(\frac{\gamma^*}{2+\varepsilon})^d - \frac{1}{d}\right)} \tag{8}$$

Note that $\varepsilon < 1$ and $\gamma^* > 3(\frac{1}{d})^{\frac{1}{d}}\left(\frac{\mu(\mathcal{X}_{free})}{\zeta_d}\right)^{\frac{1}{d}}$, and $-\frac{\zeta_d}{\mu(\mathcal{X}_{free})}(\frac{\gamma^*}{2+\varepsilon})^d - \frac{1}{d} < 0$ is always satisfied. Because $\mathbb{P}(F_n^c) < \infty$ holds, $\mathbb{P}(\lim_{n\to\infty} F_n^c) = 0$. Thus, we have $\mathbb{P}(\lim_{n\to\infty} F_n) = 1$, which implies that each ball of B_n contains at least one vertex. ■

Once NoD-RRT runs n iterations, we denote the path set Σ_n as all paths returned by NoD-RRT, and we denote the closest path $\sigma^c \in \Sigma_n$ to σ_n, i.e., $\sigma_c := \min_{\sigma_c\in\Sigma_n} |\sigma_c - \sigma_n|$. Then,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2018.2816000, IEEE Transactions on Industrial Electronics

we will show that the bound of variance of $\sigma_c - \sigma_n$ is almost surely finite, which is formally described as Lemma 4.

Lemma 4 *$|c(\sigma_c) - (\sigma_n)| < \epsilon c(\sigma_n)$ is almost surely satisfied with sufficiently large n.*

Proof: This proof can be followed by Lemma 55 in [35]. The main difference lies in the process of extending the edges in the random search tree. An asymptotically stable Lyapunov-based controller in Definition 4 is used to generate the new vertex x_{new}. According to the definition of the controller, x_{new} will finally be added such that $|x_{new} - x_{sample}| < \epsilon$ for any small positive $\epsilon > 0$ if a segment path from $x_{nearest}$ to x_{sample} exists. It means that if samples are randomly selected in each ball of B_n, i.e., $\bigcap_{i=1}^{M} \left(V_{NoD-RRT} \bigcap B_{n,i}\right) \neq \emptyset$. Thus, the conclusion could be achieved. This completes the proof. ■

Lemma 5 *[Asymptotic Near-optimality of NoD-RRT] Given any planning problem $(\mathcal{X}_{free}, x_{init}, \mathcal{X}_{goal})$ and the cost function $c(\pi)$, for any positive constant $\epsilon > 0$, the solution returned by NoD-RRT has at most $(1+\epsilon)c(\sigma^*)$.*

Proof: Under the assumption of Lemma 2, we know that $\lim_{n\to\infty} \sigma_n = \sigma^*$. Additionally, $\lim_{n\to\infty} c(\sigma_c) < (1+\epsilon)c(\sigma_n)$ is satisfied from Lemma 4. Thus, we obtain $\mathbb{P}(\{\lim_{n\to\infty} c(y^{NoD-RRT}) < (1+\epsilon)c(\sigma^*)\}) = 1$. ■

V. Applications

In this section, several numerical simulations are conducted to evaluate our motion planning algorithms. We provide two applications with the comparison with other algorithms, i.e., typical RRT, RRT* and kinodynamic RRT*. Autonomous underwater vehicles (AUVs) are widely applied both in civil and military applications, such as salvage operations, ocean surveying and field monitoring, among others. To perform underwater tasks safely and effectively, considering motion planning for AUVs is necessary. In addition, the dynamics of AUVs are nonlinear and have many constraints, such as turning radius, velocity saturation, and so on. Moreover, the optimal solution of the BVP for AUVs is difficult to be solved in closed-form. The dynamics of an underactuated system is described as

$$\begin{aligned} \dot{x} &= R(\psi)v \\ \dot{v} &= -M^{-1}[(C(v) + D(v))v + g(x) + \Delta(x,v)] \quad (9) \end{aligned}$$

where the parameters are chosen the same as in [36]. An AUV is set to be in a square space $\mathbb{R}^2$. The initial state of the AUV is $[-9m, -9m, \pi/2rad, 0, 0, 0]^\top$, and the goal state set is set to be $\{x||x-[9,9]^\top|_2 \leq 0.5\}$. For simplicity, the PD controller is utilized as the AUV's controller, where the parameters are designed as $K_p = [3,3,1]^\top$ and $K_d = [7,7,3]^\top$.

First, the NN is trained for the approximation of $\mathscr{L}(x_i, x_j)$, and the training results are shown in Figs. 8-10. In this case, we set 10 hidden neurons with a sigmoid function, i.e., $S(x) = \frac{1}{1+e^{-x}}$. The gradient descent of $\mathscr{L}(x_i, x_j)$ describes the potential area where the cost function may decrease with high probability.

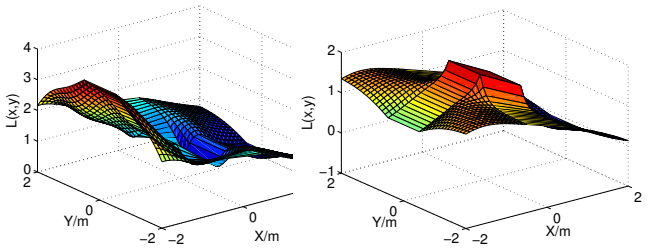


Fig. 8. $\mathscr{L}(x_i, x_j)$ when $v = [1,0,0]^\top$.

Fig. 9. $\mathscr{L}(x_i, x_j)$ when $v = [3,0,0]^\top$.

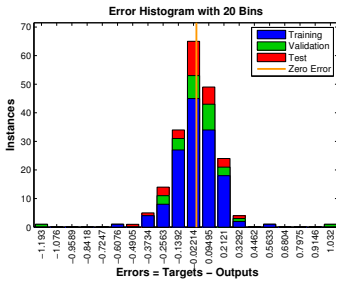


Fig. 10. Fitting results.

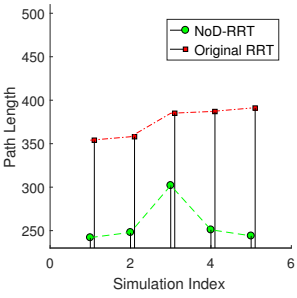


Fig. 11. Five additional simulations for an AUV are run with the same parameters.

$\mathscr{L}(x_i, x_j)$ is shown in Fig. 9 when $v = [3,0,0]^\top$, which describes how the velocity affects the cost function due to the kinodynamic constraints. Note that $\mathscr{L}(x_i, x_j)$ is large when x_j is located behind the orientation of the velocity $v(x_j)$. For example, from Fig. 8, $\mathscr{L}(x_i, x_j)$ is higher if x_j lies on the left plane. In addition, if $|v|_2 = 3$ and x_j locates near x_i and behind the orientation of the velocity, then $\mathscr{L}(x_i, x_j)$ has a clear increase.

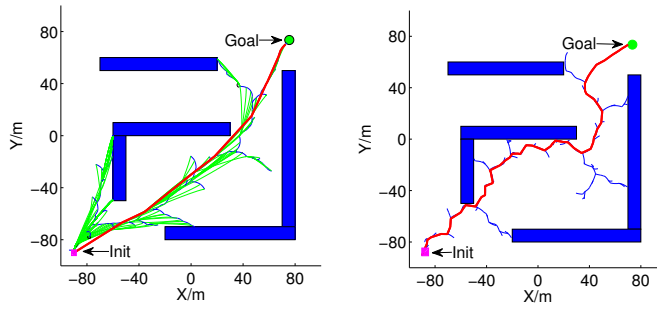
A. Comparison with typical RRT

NoD-RRT and original RRT are both applied on the AUV that is described above with the same environments, including the same obstacles and controllers. The range of the environment is set to be $\mathcal{X} \in [-100, 100]^\top \times [-100, 100]^\top$. The bounds for the velocity of the AUV are set to be $[2m/s, 2m/s, 0.18rad/s]$. Moreover, the parameters are set to be the same as in [37]. The results are shown in Fig. 12, from which we can observe that the cost function (i.e., the total path length from the start node to the goal node) decreases more by NoD-RRT than that by RRT.

In fact, the sampling sequence is different between each simulation, even when the same parameters are adapted. Thus, we test the performance of NoD-RRT with multiple simulations under the same situations and parameters. The results are shown in Fig. 11, from which we can observe that NoD-RRT returns better solutions than the original RRT for an AUV.

In addition, we also present a numerical simulation of NoD-RRT with data from national oceanic and atmospheric

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2018.2816000, IEEE Transactions on Industrial Electronics



(a) NoD-RRT planning for an AUV. (b) RRT planning for an AUV.

Fig. 12. Comparison between NoD-RRT and typical RRT. The iterations of NoD-RRT and typical RRT are 242 and 391, respectively.

administration (NOAA). This simulation could serve as a guide for implementation in physical settings. We apply NoD-RRT to search the planar trajectory for AUVs. We select the parameters for the AUV the same as in (9). The digital elevation model (DEM) contains a raster grid covering Pearl Harbor, Oahu, Hawaii, at a 1/3-arc-second (10 m) resolution. Without loss of generality, we clip a rectangle from Pearl Harbor, as shown in Fig. 13. Considering the safety of the AUV, we set regions deeper than 5 meters as C_{free}. The results of this simulation are obtained using MATLAB with an Intel(R) Core i7-4720.

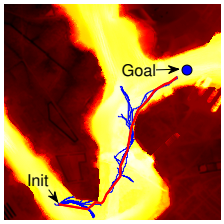


Fig. 13. Selected region of Pearl Harbor. Lower left point locates at $(157°58'48''\mathrm{W}, 21°20'24''\mathrm{N})$. Upper right point locates at $(157°57'36''\mathrm{W}, 21°21'36''\mathrm{N})$. Blue lines describe the expanding tree. Red line shows the available trajectory from x_{init} to X_{goal}. Initial state is located at $(157°58'30''\mathrm{W}, 21°20'29''\mathrm{N})$, and goal region set is defined as a circle whose center locates at $(157°57'47''\mathrm{W}, 21°21'14''\mathrm{N})$.

B. Comparison with RRT* and kinodynamic RRT*

In this section, we present a comprehensive comparison with RRT* and kinodynamic RRT*. First, RRT* considers the path as a series of line segments. Since it is not concerned about the dynamic constraints directly, the path should be smoothed using an interpolation method, such as B-spline. Meanwhile, kinodynamic RRT* is proposed to refine the path toward optimality using a fixed-final-state-free-final-time controller. As described in [15], nonlinear dynamics must be linearized in the form of first-order approximations as

$$\dot{x} = Ax + Bu + c \tag{10}$$
$$A = \frac{\partial f}{\partial \bar{x}} \quad B = \frac{\partial f}{\partial \bar{u}} \quad c = f - A\bar{x} - B\bar{u}$$

where $\bar{x}$ and $\bar{u}$ are the state and control input where the system dynamics is linearized. In kinodynamic RRT*, "weighted controllability Gramian", i.e., $G(t)$ (see (11)) needs to be calculated for an optimal BVP problem given two states. If the closed-form of $G(t)$ is available, the computation time can be remarkably reduced. If not, a numerical integration, such as 4-order Runge Kutta needs to be conducted.

$$G(t) = \int_0^t \exp[A(t-t')BR^{-1}B^\top \exp[A^\top(t-t')]]\mathrm{d}t' \tag{11}$$

It certainly results in considerably more computational time, which does not exist for the systems with linear dynamics. A summary of the comparison between RRT* and kinodynamic RRT* is presented in Table I.

TABLE I
COMPARISON BETWEEN NOD-RRT AND RRT*, KINODYNAMIC RRT*

robot dynamics	linear	nonlinear (closed-form of Gramian available)	nonlinear (closed-form of Gramian available)
RRT*	Yes(complex)	No	No
kinodynamic RRT*	Yes	Yes(effective)	Yes(complex)
NoD-RRT*	Yes	Yes	Yes(effective)

To show the difference in performance between NoD-RRT and kinodynamic RRT*, we conducted simulations on both algorithms with linear and nonlinear systems in the same scenarios. A linear system can be described as an integrator robot that operates in the plane, which is formulated as in

$$\dot{x} = Ax + Bu; x \in \mathbb{R}^4; A = [0\ I; 0\ 0]; B = [0; I] \tag{12}$$

Meanwhile, we have also conducted simulations of kinodynamic RRT* with a nonlinear system the same as in Fig. 12(a). Furthermore, we set the initial and goal to be located similarly as $[-90, -90]^\top$ and $[80, 80]^\top$, respectively. As shown in Fig. 14, kinodynamic RRT* returns both the near-optimal path and the optimal path successfully, although with different computation times, as shown in Table II. We can observe that kinodynamic RRT* has remarkable efficiency for the systems with linear dynamics compared with NoD-RRT. Moreover, NoD-RRT performs better for systems with nonlinear dynamics, particularly when the closed-form Gramian matrix is unavailable.

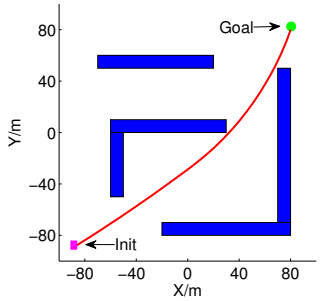
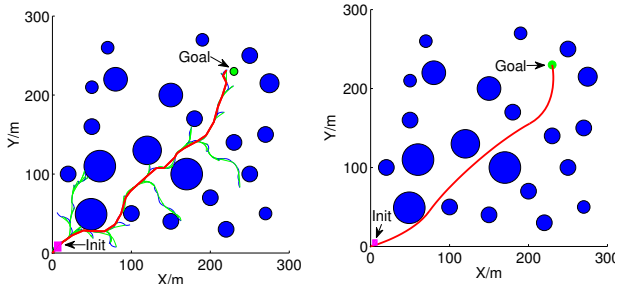


Fig. 14. Path generated by kinodynamic RRT* with linear system. Iteration is 210. Path length is 241.1 m.

TABLE II
RUNNING TIMES OF KINODYNAMIC RRT* AND NOD-RRT

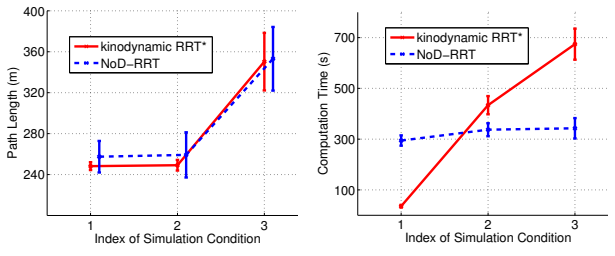
	linear dynamics	nonlinear (closed-form of Gramian unavailable)
kinodynamic RRT*	35.7s	433.8s
NoD-RRT	294.5s	337.1s

To show the advantages of NoD-RRT, we also conduct a simulation in a scenario with a large number of obstacles as in [20]. The start and goal locate at $[0,0]^\top$ and $[230,230]^\top$, respectively, which are denoted as a pink rectangle and a green circle. We attempt to drive the AUV toward the goal with many clustered circular obstacles with NoD-RRT and kinodynamic RRT*. Fig. 15 shows the results, from which it can be observed that both algorithms can search a valid path, although with different computation time. In addition, the performance comparison (path length and computation time) is described in Fig. 16, including the simulations for systems with linear and nonlinear dynamics.



(a) NoD-RRT planning for an AUV. (b) Kinodynamic RRT* planning for an AUV.

Fig. 15. Paths returned by NoD-RRT and kinodynamic RRT*. The iterations of both NoD-RRT and kinodynamic RRT* are 300.



(a) Path lengths returned by NoD-RRT and kinodynamic RRT* (b) Computation times of NoD-RRT and kinodynamic RRT*

Fig. 16. Comparison of the results (path length and computation time) in NoD-RRT and kinodynamic RRT*. Index means (1) linear systems (linear double integrator model), (2) nonlinear systems (AUV model), and (3) AUV model in clustered environment.

In contrast to the typical RRT algorithm, kinodynamic RRT* attempts to connect the random sampled nodes directly. Although the asymptotic convergency is indeed ensured, useful extending attempts may be ignored by kinodynamic RRT*, particularly for scenarios with a greater number of obstacles. Compared with kinodynamic RRT*, NoD-RRT avoids the computation of the connection directly. The computation of NoD-RRT mainly lies in the neural network, such as the numbers of hidden layers and neurons and the activation function of neurons.

In NoD-RRT, the neural network is composed of a hidden layer and a linear output layer. Suppose that the numbers of neurons in the hidden layer and output layer are n_h and n_o, respectively, and the number of training samples is m. The computational complexity of the neural network can be denoted as $O(m(3n_h + n_h n_o))$. Since $n_o = 1$ in our neural network, the computational complexity can be obtained as $O(mn_h)$. Thus, the time complexity of the prediction is $O(n_h)$. Consequently, NoD-RRT will be a better choice when systems have nonlinear dynamics and the closed-form of Gramian is unavailable.

C. Experiments with a Pioneer3-DX robot

To demonstrate the effectiveness of NoD-RRT, we apply NoD-RRT with a Pioneer3-DX robot, which was used in our previous researches [38]. A neural network with 10 sigmoid neurons is trained in advance. The robot uses sonar to detect obstacles. The robot used in the experiment has 8 forward-facing ultrasonic (sonar) sensors, and 8 rear-facing sonar. As we restrict the robot moving forward only, the obstacles are detected by the forward-facing sonar, which is uniformly distributed on the front side as shown in Fig. 17 [39]. As shown in Fig. 18, the node of random searching tree, x_{rand}, which needs to be tested whether it is feasible, has a distance of $d_{x_{rand},robot}$ from the robot. The corresponding sonar, which measures the sector where x_{rand} locates, returns the minimum distance $d_{\min}$ from obstacles. If $d_{\min} - d_{x_{rand},robot} < \delta$, then x_{rand} is unfeasible and is regenerated until $x_{rand} \in C_{free}$.

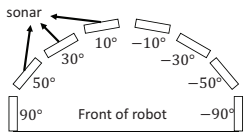


Fig. 17. 8 forward-facing ultrasonic (sonar) sensors on the Pioneer3-DX robot.

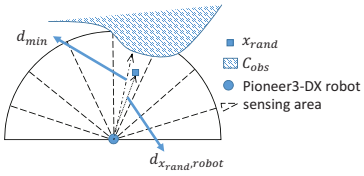


Fig. 18. Collision test of x_{rand}.

Control inputs are calculated according to NoD-RRT per $0.1s$. The boundary of the environment is $\{x \in \mathcal{X} : x_1 \in [0, 4.2m], x_2 \in [0, 3.5m]\}$. There exist four obstacles in the environment of this experiment. The initial state and the goal region locate at $[0,0,0]^\top$ and $[3.8, 2.6, \pi/4]^\top$, respectively. Pioneer3-DX used in the experiment has a maximum velocity of 1.2m/s. Because of the limited environment boundary, we restrict the velocity of each wheel as $v \in (0, 0.3m/s]$. It also means that the robot can only move forward. The distance between two wheels is $0.34m$. Accordingly, the maximum turning radius is $0.17m$. The experiment result is shown in Fig. 19, in which the planning trajectory returned using NoD-RRT and actual trajectory are presented with red and green

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2018.2816000, IEEE Transactions on Industrial Electronics

line, respectively. Snapshots of the experiment process are presented in Fig. 20. From the results we can observe that the Pioneer3-DX robot avoids the obstacles and reaches the goal region along a near-optimal path. The length of the path is 3.49m.

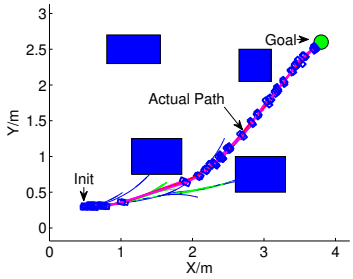
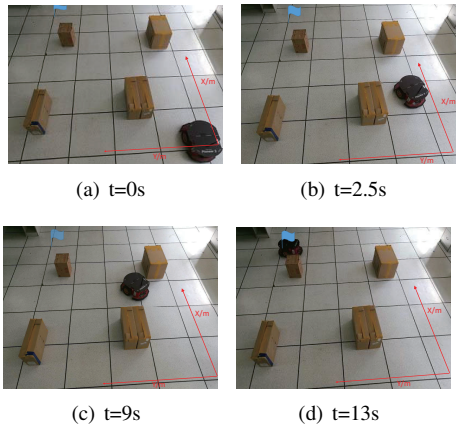


Fig. 19. Actual path of the Pioneer3-DX robot.



(a) t=0s (b) t=2.5s

(c) t=9s (d) t=13s

Fig. 20. Snapshots of the path planning for Pioneer3-DX robot. X and Y axis are denoted with red line. Goal region is denoted as a blue flag.

Here, we present several notes on the practical application of the proposed algorithms. i) Since NoD-RRT is a hybrid of a sampling-based planning algorithm with NNs, the computational complexity is also affected by the quality of the random sampling process. In the case of large-scale environments, the goal bias sampling strategy is simple and effective. ii) Indeed, NoD-RRT can also be applied on real-time tasks. For real-time implementations, we generate training data randomly based on the kinodynamic constraints of vehicles according to the method that we described above. To return a valid path online, the number of hidden layers and neurons needs to be restricted due to the limited computational capability. The approximation error of the NNs may increases slightly. In this case NoD-RRT will need more samples to search a reasonable δ-clearance near-optimal path.

VI. Conclusions

In this paper, we have presented a new motion planning approach, namely, NoD-RRT, for the purpose of addressing kinodynamic constraints. NoD-RRT utilizes NNs to approximate the cost function. Moreover, a new reconstruction of the search tree has been proposed based on RRT*. By using the tools *attraction sequence*, theoretical analysis has indicated that NoD-RRT has the properties of probabilistic completeness and asymptotic near-optimality. Moreover, NoD-RRT translates the nonlinear constraints into the form of cost/metric approximated by NNs. It leads to the computational time being independent of the robot or vehicle dynamics that are considered in the motion planning. Thus, NoD-RRT can return an acceptable near-optimal solution within a reasonable computational time, particularly for the systems with nonlinear dynamics that can not be solved in unclosed-form.

References

[1] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[2] A. A. Agha-Mohammadi, S. Chakravorty, and N. M. Amato, "Firm: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements," *International Journal of Robotics Research*, vol. 33, no. 2, pp. 268–304, 2014.

[3] O. Salzman, M. Hemmer, and D. Halperin, "On the power of manifold samples in exploring configuration spaces and the dimensionality of narrow passages," *IEEE Transactions on Automation Science & Engineering*, vol. 12, no. 2, pp. 529–538, 2014.

[4] Y. Wang, Y. Zhao, S. A. Bortoff, and K. Ueda, "A real-time energy-optimal trajectory generation method for a servomotor system," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 2, pp. 1175–1188, 2015.

[5] Y. Kim and B. K. Kim, "Time-optimal trajectory planning based on dynamics for differential-wheeled mobile robots with a geometric corridor," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 7, pp. 5502–5512, 2017.

[6] B. Zhang, L. Tang, J. Decastro, M. J. Roemer, and G. Kai, "A recursive receding horizon planning for unmanned vehicles," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 5, pp. 2912–2920, 2015.

[7] M. Mansouri, M. A. Shoorehdeli, and M. Teshnehlab, "Integer GA for mobile robot path planning with using another GA as repairing function," in *IEEE International Conference on Automation and Logistics*, pp. 135–140, 2008.

[8] S. M. Lavalle, *From Dynamic Programming to RRTs: Algorithmic Design of Feasible Trajectories*. Springer Berlin Heidelberg, 2003.

[9] Lavalle and S. M, "Rapidly-exploring random trees: A new tool for path planning," *Algorithmic Computational Robotics New Directions*, pp. 293–308, 1999.

[10] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[11] O. Arslan and P. Tsiotras, "Use of relaxation methods in sampling-based algorithms for optimal motion planning," in *IEEE International Conference on Robotics and Automation*, pp. 2421–2428, 2013.

[12] F. S. Hover, R. M. Eustice, A. Kim, B. J. Englot, H. Johannsson, M. Kaess, and J. J. Leonard, "Advanced perception, navigation and planning for autonomous in-water ship hull inspection," *International Journal of Robotics Research*, vol. 31, no. 12, pp. 1445–1464, 2012.

[13] V. Narayanan, M. Phillips, and M. Likhachev, "Anytime safe interval path planning for dynamic environments," in *IEEE/RSJ International Conference on Intelligent Robots & Systems*, pp. 5628–5635, 2011.

[14] D. Devaurs, T. Siméon, and J. Cortés, "Optimal path planning in complex cost spaces with sampling-based algorithms," *IEEE Transactions on Automation Science & Engineering*, vol. 13, no. 2, pp. 415–424, 2016.

[15] D. J. Webb and J. V. D. Berg, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," in *IEEE International Conference on Robotics and Automation*, pp. 5054–5061, 2013.

[16] Y. Li, Z. Littlefield, and K. E. Bekris, *Sparse Methods for Efficient Asymptotically Optimal Kinodynamic Planning*. Springer International Publishing, 2015.

[17] O. Salzman and D. Halperin, "Asymptotically near-optimal RRT for fast, high-quality, motion planning," in *IEEE International Conference on Robotics and Automation*, pp. 4680–4685, 2014.

[18] M. Otte and E. Frazzoli, "RRTX: Asymptotically optimal single-query sampling-based motion planning with quick replanning," *International Journal of Robotics Research*, 2015.

[19] P. Pharpatara, B. Hérissé, and Y. Bestaoui, "3-D trajectory planning of aerial vehicles using RRT," *IEEE Transactions on Control Systems Technology*, 2016.
[20] K. Yang, S. Moon, S. Yoo, J. Kang, N. L. Doh, H. B. Kim, and S. Joo, "Spline-based RRT path planner for non-holonomic robots," *Journal of Intelligent & Robotic Systems*, vol. 73, no. 1, pp. 763–782, 2014.
[21] G. Goretkin, A. Perez, R. Platt, and G. Konidaris, "Optimal sampling-based planning for linear-quadratic kinodynamic systems," in *IEEE International Conference on Robotics and Automation*, pp. 2429–2436, 2013.
[22] A. Kaplan, N. Kingry, P. Uhing, and R. Dai, "Time-optimal path planning with power schedules for a solar-powered ground robot," *IEEE Transactions on Automation Science & Engineering*, pp. 1–10, 2016.
[23] Y. J. Liu, S. Tong, C. L. Chen, and D. J. Li, "Adaptive NN control using integral barrier lyapunov functionals for uncertain nonlinear block-triangular constraint systems," *IEEE Transactions on Cybernetics*, vol. PP, no. 99, pp. 1–11, 2016.
[24] S. S. Ge, C. C. Hang, T. H. Lee, and T. Zhang, "Stable adaptive neural network control," *Springer International*, vol. 13, 2001.
[25] W. He, A. O. David, Z. Yin, and C. Sun, "Neural network control of a robotic manipulator with input deadzone and output constraint," *IEEE Transactions on Systems Man & Cybernetics Systems*, vol. 46, no. 6, pp. 759–770, 2016.
[26] K. Hornik, "Some new results on neural network approximation," *Neural Networks*, vol. 6, no. 9, pp. 1069–1072, 1993.
[27] W. He, Y. Chen, and Z. Yin, "Adaptive neural network control of an uncertain robot with full-state constraints." *IEEE Transactions on Cybernetics*, vol. 46, no. 3, pp. 620–629, 2016.
[28] Q. Wei and D. Liu, "Neural-network-based adaptive optimal tracking control scheme for discrete-time nonlinear systems with approximation errors," *Neurocomputing*, vol. 149, no. PA, pp. 106–115, 2015.
[29] D. Costarelli and G. Vinti, "Pointwise and uniform approximation by multivariate neural network operators of the max-product type," *Neural Networks*, vol. 81, p. 81, 2016.
[30] Y. J. Liu and S. Tong, "Adaptive NN tracking control of uncertain nonlinear discrete-time systems with nonaffine dead-zone input." *IEEE Transactions on Cybernetics*, vol. 45, no. 3, pp. 497–505, 2014.
[31] M. Mansouri, M. Teshnehlab, and M. A. Shoorehdeli, "Indirect adaptive hierarchical fuzzy sliding mode controller for a class of nonlinear systems," *Journal of Intelligent & Fuzzy Systems*, vol. 30, no. 3, pp. 1377–1391, 2016.
[32] K. Ito and K. Kunisch, *Optimal Control*. John Wiley & Sons, Inc., 1999.
[33] R. Battiti, "First- and second-order methods for learning: between steepest descent and newton's method," *Neural Computation*, vol. 4, no. 2, pp. 141–166, 1992.
[34] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.
[35] S. Karaman and E. Frazzoli, "Sampling-based motion planning with deterministic μ-calculus specifications," in *Proc. the 48th IEEE Conference on Decision and Control, 2009.*, pp. 2222–2229. IEEE, 2009.
[36] W. Dong and Y. Guo, "Global time-varying stabilization of underactuated surface vessel," *IEEE Transactions on Automatic Control*, vol. 50, no. 6, pp. 859–864, 2005.
[37] R. Cui, C. Yang, Y. Li, and S. Sharma, "Adaptive neural network control of AUVs with control input nonlinearities using reinforcement learning," *IEEE Transactions on Systems Man & Cybernetics Systems*, vol. 47, no. 6, pp. 1019–1029, 2017.
[38] R. Cui, B. Gao, and J. Guo, "Pareto-optimal coordination of multiple robots with safety guarantees," *Autonomous Robots*, vol. 32, no. 3, pp. 189–205, 2012.
[39] *Pioneer 3DX Operational Manual*, http://www.mobilerobots.com/Mobile_Robots.aspx, Mobile Robot Inc., 2007.

Yang Li received the B.Eng. and degree in automatic control from Northwestern Polytechnical University, Xi'an, China, in 2012. He is currently pursuing the Ph.D. degree in navigation, guidance and control with the School of Marine Science and Technology, Northwestern Polytechnical University. His current research interests include cooperative path planning and control of multiple robots.

Rongxin Cui (M'09) received B.Eng. and Ph.D. degrees from Northwestern Polytechnical University, Xi'an, China, in 2003 and 2008, respectively. From August 2008 to August 2010, he worked as a Research Fellow at the Centre for Offshore Research & Engineering, National University of Singapore, Singapore. Currently, he is a Professor with the School of Marine Science and Technology, Northwestern Polytechnical University, Xi'an, China. His current research interests are control of nonlinear systems, cooperative path planning and control for multiple robots, control and navigation for underwater vehicles, and system development.

Dr. Cui servers as an Editor for the Journal of Intelligent & Robotic Systems.

Zhijun Li (M'07-SM'09) received the Ph.D. degree in mechatronics, Shanghai Jiao Tong University, P.R. China, in 2002. From 2003 to 2005, he was a postdoctoral fellow in Department of Mechanical Engineering and Intelligent systems, The University of Electro-Communications, Tokyo, Japan. From 2005 to 2006, he was a research fellow in the Department of Electrical and Computer Engineering, National University of Singapore, and Nanyang Technological University, Singapore. Since 2012, he is a Professor in College of Automation Science and Engineering, South China university of Technology, Guangzhou, China.

He is serving as an Editor-at-large of Journal of Intelligent & Robotic Systems, and Associate Editors of several IEEE Transactions. Dr. Li's current research interests include service robotics, tele-operation systems, nonlinear control, neural network optimization, etc.

Demin Xu received the M.Eng degree in underwater weapon from Northwestern Polytechnical University, Xi'an, China, in 1964. He is currently a Professor with the School of Marine Science and Technology, Northwestern Polytechnical University. His current research interests include underwater vehicles and control.