Informed Anytime Fast Marching Tree for Asymptotically-Optimal Motion Planning

Jing Xu, Kechen Song, Defu Zhang, Hongwen Dong, Yunhui Yan,
and Qinggang Meng, *Senior Member, IEEE*

***Abstract*—In many applications, it is necessary for motion planning planners to get high-quality solutions in high-dimensional complex problems. In this paper, we propose an anytime asymptotically-optimal sampling-based algorithm, namely Informed Anytime Fast Marching Tree (IAFMT*), designed for solving motion planning problems. Employing a hybrid incremental search and a dynamic optimal search, the IAFMT* fast finds a feasible solution, if time permits, it can efficiently improve the solution toward the optimal solution. This paper also presents the theoretical analysis of probabilistic completeness, asymptotic optimality, and computational complexity on the proposed algorithm. Its ability to converge to a high-quality solution with the efficiency, stability, and self-adaptability has been tested by challenging simulations and a humanoid mobile robot.**

***Index Terms*—Asymptotic optimality, fast marching tree, informed anytime algorithm, motion planning**

I. Introduction

MOTION planning is such a fundamental research topic in robotics that it is widely applied to industrial robots, medical robots, bionic robots, and smart vehicles [1]-[6]. Hence, motion planning methods are still of high scientific interest, particularly for high-dimensional complex motion planning problems. Motion planning aims to search collision-free paths guiding robots from an initial node to a goal region in a configuration-space (C-space) full of obstacles [7], [8]. Arguably, sampling-based methods, such as Rapidly-exploring Random Trees (RRT) [9], Probabilistic Roadmaps (PRM) [10], and their variants [11]-[13], are among the most popular and widespread methods available in practical applications. There are natural advantages for sampling-based methods to solve high-dimensional problems as they avoid the explicit construction of the C-space by randomly sampling [14], [15].

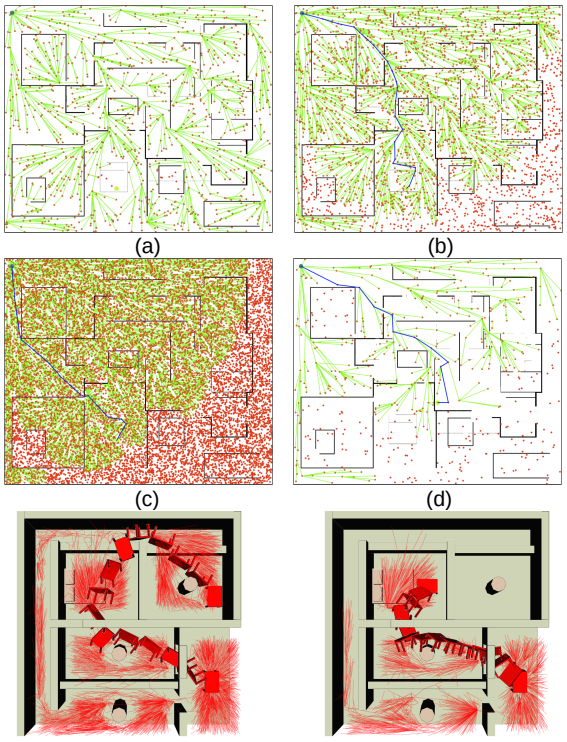


Fig. 1. Solutions of motion planning provided by FMT*. Let n denote the number of samples and J is path cost in C-space. (a) Failed planning: $n = 1000$, $J = \infty$. (b) A feasible path: $n = 3000$, $J = 1783$. (c) A high-quality path: $n = 15000$, $J = 1616$. (d) Change the goal node in the 2D environment: $n = 500$, $J = 1698$. (e) A high-quality path: $n = 15000$, $J = 1295$. (f) Change the goal node in the 3D environment: $n = 5000$, $J = 687$.

Efforts to find high-quality solutions, which quality can be measured according to length, clearance, smoothness, and energy, lead to asymptotically-optimal (AO) methods. A high-quality solution is a path whose cost is lower than the given cost threshold or close to the cost of the optimal solution. The optimal solution is the path with the lowest cost in a C-space. Methods with asymptotic optimality, such as RRT* and PRM* [16], are able to asymptotically converge to the optimal solution as the number of samples goes to infinity. Whether the optimal solution can be converged is usually concerned in theory, while in practical applications we focus on obtaining high-quality solutions, not the optimal solution, because it is difficult for motion planning methods to converge to the optimal solution in limited runtime and computational

This work was supported by the National Natural Science Foundation of China (51805078, 51374063), the National Key Research and Development Program of China (2017YFB0304200). *(Corresponding authors: Yunhui Yan; Kechen Song)*

J. Xu, K. Song, D. Zhang, H. Dong, and Y. Yan are with the School of Mechanical Engineering and Automation, Northeastern University, Shenyang, Liaoning, 110819, China, and the Key Laboratory of Vibration and Control of Aero-Propulsion Systems Ministry of Education of China, Northeastern University, Shenyang, 110819, China. (e-mail: jing_xu@yeah.net, songkc@me.neu.edu.cn, zdf1985681480@163.com, donghongwenliran@163.com, yanyh@mail.neu.edu.cn).

Q. Meng is with the department of computer science, Loughborough University, Loughborough LE11 3TU, U.K. (e-mail: q.meng@lboro.ac.uk).

resources.

For high-dimensional complex problems, sampling-based AO methods may not consider the efficiency and stability to achieve high-quality solutions [17], [18]. The informed sampling technique is introduced to AO methods, such as Informed RRT* [19] and Batch Informed Trees (BIT*) [20], to reduce computational burden by defining the sampling region as a hyper-ellipsoid for finding a high-quality solution. Fast Marching Trees (FMT*) [21] performs a "lazy" dynamic programming recursion on a batch of random samples to generate a low-cost tree as its roadmap. It is capable of solving high-dimensional complex problems efficiently, especially in scenarios where collision-checking is expensive, if the number of samples is reasonable. However, when the number of samples is small, FMT* is difficult to obtain high-quality solutions, sometimes it even fails to converge, as shown in Fig. 1(a) and (b). FMT* will lose the advantage of computational efficiency if the number of samples is large, as presented in Fig. 1(c).

For different problems, it is hard for motion planning methods without self-adaptability to fast converge to high-quality solutions, which means that the reasonable parameter values of a method may be completely different when solving simple problems and complex problems respectively. In short, methods with self-adaptability can automatically tune parameters as requirements change. For example, the performance of FMT* and its variants [22], [23] heavily depends on the number of samples, which leads to lack of self-adaptability for the methods. The reasonable number of samples may be several thousand in 2D environments, while that could be tens of thousands in 3D environments, as shown in Fig. 1(e), even in the same environment, the different number of samples are needed to solve problems with different levels of difficulty, as shown in Fig. 1(d) and (f). It is not easy to determine the reasonable number of samples for an unfamiliar problem. Some AO methods have anytime performance leading to self-adaptability, named anytime AO methods, such as RRT*, PRM*, and their variants [24]-[26]. Anytime methods provide any solution as quickly as possible and yield higher-quality solutions if time permits [27], [28]. Anytime AO methods fast search for an initial solution and asymptotically improve the solution toward the optimal solution. Motion Planning using Lower Bounds (MPLB) [29] is a quasi-anytime algorithm by solving a series of independent problems and better solutions are only returned after a sub-problem is solved.

This paper presents an anytime AO algorithm, namely Informed Anytime Fast Marching Tree (IAFMT*). IAFMT* extends FMT* to an anytime algorithm by designing a hybrid incremental search and a dynamic optimal search, in addition, the informed sampling technique is employed to refine efficiency. The hybrid incremental search integrates the batch sampling search, like FMT*, and the single sampling search, like RRT*, to build a low-cost spanning tree, which balances the efficiency of the batch sampling search and the flexibility of the single sampling search. As we know the "non-lazy" search tends to find a high-quality solution by taking lots of computational resources, while the "lazy" search efficiently obtains a feasible solution. The proposed dynamic optimal search considers a tradeoff between the "non-lazy" and "lazy" searches to fast improve a spanning tree and it achieves a high-quality solution. Additionally, this paper gives the theoretical analysis on the IAFMT* in depth, such as the analysis of probabilistic completeness, asymptotic optimality, and computational complexity. This paper also comprehensively evaluates the performance of the proposed algorithm by a series of 2D, 3D simulations, and real-world experimental tests. The experimental results show that the proposed algorithm, IAFMT*, has the ability to converge to a high-quality solution with the efficiency, stability, and self-adaptability when compared with the state-of-the-art algorithms, namely PRM*, Informed RRT*, and FMT*.

Our contribution can be summarized that

1) This paper presents an anytime sampling-based AO algorithm, namely IAFMT*, which is used to solve high-dimensional complex motion planning problems and get high-quality solutions.
2) We propose a strategy that balances the benefits of the batch and single sampling searches, the "non-lazy" and "lazy" searches, which makes IAFMT* efficient, stable, and self-adaptive.
3) In order to support the proposed algorithm in theory, we provide the analysis of probabilistic completeness, asymptotic optimality, and computational complexity.
4) This paper comprehensively evaluates the performance of IAFMT* for solving high-dimensional complex problems by simulations and a humanoid mobile robot.

This paper is organized as follows. In Section II, the problem is formally defined. Section III introduces the proposed algorithm, IAFMT*. The theoretical analysis of IAFMT* is provided in Section IV. Next, Section V presents the results of the simulation and experiments. Finally, Section VI concludes this paper and gives future research directions.

II. Problem Formulation

Let $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{X}_{\text{free}} \subset \mathcal{X}$ be a d-dimensional C-space and the obstacle-free space, respectively. Let $x_{\text{init}} \in \mathcal{X}_{\text{free}}$ denote the initial node and $\mathcal{X}_{\text{goal}} \subset \mathcal{X}_{\text{free}}$ be the set of goal nodes. A feasible path to the motion planning problem can be denoted by $\sigma: [0,1] \mapsto \mathcal{X}$ if $\sigma(\tau) \in \mathcal{X}_{\text{free}}$, where $\forall \tau \in [0,1]$, which means that the initial node x_{init} connects to any node $x_{\text{goal}} \in \mathcal{X}_{\text{goal}}$ through free space. Let Σ be the set of all feasible paths and $J(\sigma)$ be the path cost according to Euclidean metric in $\mathcal{X}$.

The optimal motion planning problem can be described that given a path cost function $J: \Sigma \mapsto \mathbb{R} \geq 0$, finding the optimal solution σ^* satisfies

$$\sigma^* = \arg\min_{\sigma \in \Sigma} \left\{ J(\sigma) \middle| \begin{matrix} \sigma(0) = \mathrm{x}_{\text{init}}, \sigma(1) = \mathrm{x}_{\text{goal}}, \\ \forall \tau \in [0,1], \sigma(\tau) \in \mathcal{X}_{\text{free}} \end{matrix} \right\}. \quad (1)$$

This paper will interchangeably refer to points in $\mathcal{X}$ as nodes or samples.

III. Informed Anytime Fast Marching Tree

We now present the Informed Anytime Fast Marching Tree algorithm, namely IAFMT*, described in the pseudo-code from Algorithm 1 to 5.

A. Algorithm Overview

It is necessary to introduce some notions and functions before describing IAFMT*. V_i is the set of nodes including x_{init}, x_{goal}, and batch random samples n generated by $\text{SampleFree}(n)$. $T = (V, E)$ denotes a spanning tree, where V and E are the tree-node and tree-edge sets, respectively. V_{u} is the unvisited node set where the nodes are not added to the tree. V_{op} contains the open nodes that may be expanded on the tree. V_{c} is the closed node set where there are no unvisited nodes near each tree node. Let $\text{Near}(V_{\text{space}}, z, r_n)$, i.e., N_{z}, be a function which returns the node set that satisfies

$$\{x \in V_{\text{space}} \big| \|x - z\| < r_n\}, \tag{2}$$

where V_{space} is the set of all nodes in the current C-space, z is an open node selected for expansion, and r_n denotes the search radius.. The search radius r_n [21] is

$$r_n = (1+\eta) \cdot 2\left(\frac{1}{d}\right)^{\frac{1}{d}} \left(\frac{\mu(X_{free})}{\zeta_d}\right)^{\frac{1}{d}} \left(\frac{\log n}{n}\right)^{\frac{1}{d}}, \tag{3}$$

where $\eta > 0$ is a small constant, $\mu(\cdot)$ denotes the Lebesgue measure, and ζ_d is the volume of the unit ball in the d-dimensional Euclidean space. $\text{Cost}_T(x)$ returns the shortest path cost from x_{init} to x on the tree, i.e., the lowest-cost value for x.

We establish the IAFMT* algorithm in Algorithm 1. The IAFMT* algorithm first determines V_i by random uniform sampling, then it initializes a tree and other parameters by the function $\text{Initialize}(V_i, V_{\text{u}}, E)$. $\text{HybridIncrementalSearch}(T, z)$ and $\text{DynamicOptimalSearch}(V_{i-1}, T)$, which are different from FMT*, are introduced to search feasible paths and high-quality paths. $\text{HybridIncrementalSearch}(T, z)$ first uses the batch sampling search to fast expand a low-cost spanning tree and try to find a feasible path σ. And if a feasible path σ is not found, the function will perform the single sampling search to incrementally search for σ. $\text{DynamicOptimalSearch}(V_{i-1}, T)$ employs the informed sampling technique to substantially reduce the search region which exists better paths and it asymptotically improves the tree T to achieve a high-quality path by integrating the "non-lazy" and "lazy" searches. The IAFMT* algorithm returns σ_{HQ} when the cost of the current feasible path is less than or equal to the given cost threshold J_{given}. σ_{HQ} is equal to σ^* when $J_{\text{given}} \leq J(\sigma^*)$ and available time t is enough.

There are three following contributions in IAFMT*: (1) IAFMT* extends FMT* to an anytime AO algorithm which can fast find a high-quality path adaptively in a given runtime; (2) The hybrid incremental search introduces the incremental single sampling search to the "lazy" batch sampling search of FMT* in order to have the ability to quickly get feasible paths as facing different motion planning problems; (3) The dynamic optimal search considers a tradeoff between the "non-lazy" and "lazy" searches to anytime asymptotically improve a spanning tree, which provides low-cost feasible paths and a high-quality path, even the optimal solution if conditions permit.

Algorithm 1 IAFMT*

Require: Query (x_{init}, x_{goal}), Search radius r_n, Sample count n_0, Available time t, Given cost threshold J_{given}

```
V_0 ← {x_init}, n ← n_0, i ← 1
while t do
  if σ = ∅ ∩ σ_HQ = ∅ then
    V_i ← V_{i−1} ∪ SampleFree(n); V_u ← V_i \ V_0; E ← ∅
    {V_op, V_c, T, z, N_z} ← Initialize(V_i, V_u, E)
    {σ, T} ← HybridIncrementalSearch(T, z);
  if σ ≠ ∅ ∩ σ_HQ = ∅ then
    σ ← DynamicOptimalSearch(V_{i−1}, T); i ← i + 1
    if Cost_T(x_goal) ≤ J_given then
      return σ_HQ ← σ
return σ
```

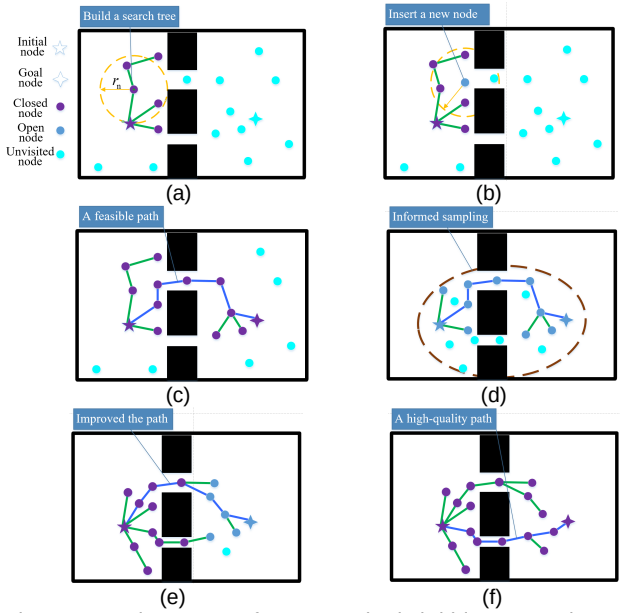


Fig. 2. Search process of IAFMT*. The hybrid incremental search is presented in (a)-(c). The dynamic optimal search is shown in (d)-(f). (a) IAFMT* builds a search tree not reaching the goal node. (b) A new node is inserted to help the tree grow. (c) A feasible path is found. (d) IAFMT* prunes the tree and performs the informed sampling. (e) The tree is improved dynamically. (f) A high-quality path is got finally.

B. Hybrid Incremental Search

The hybrid incremental search aims to find an initial path efficiently and flexibly by integrating the batch sampling search and the single sampling search, as shown in Fig. 2(a)-(c). In Algorithm 2, based on the batch random samples n in Algorithm 1, the function $\text{ExpandTree}(z)$ efficiently builds a spanning tree, where z consistently denotes the open node selected for expansion. $\text{Path}(x_{\text{goal}}, T)$ returns the lowest-cost feasible path from x_{init} to x_{goal} on the tree. However, the batch sampling search $\text{ExpandTree}(z)$ cannot always obtain a path. The single sampling search $\text{InsertNode}(T)$ flexibly connects one new node to the tree in order to assist the $\text{ExpandTree}(z)$ in finding an initial path.

Algorithm 2 HybridIncrementalSearch(T, z)

```
while z ≠ x_goal do
    {z, T} ← ExpandTree(z); σ ← Path(x_goal, T)
    if σ ≠ ∅ then break
    if σ = ∅ ∩ V_op = ∅ then T ← InsertNode(T)
return {σ, T}
```

Algorithm 3 ExpandTree(z)

```
1  V'_op ← ∅; X_near ← N_z ∩ V_u
2  for x ∈ X_near do
3      N_x ← Near(V\{x}, x, r_n); Y_near ← N_x ∩ V_op
4      y_min ← arg min_{y ∈ Y_near} {Cost_T(y) + Cost(y, x)}
5      if CollisionFree(y_min, x) then
6          T.parent(x) ← y_min
7          V'_op ← V'_op ∪ {x}; V_u ← V_u\{x}
8  V_op ← (V_op ∪ V'_op)\{z}; V_c ← V_c ∪ {z}
9  if σ ≠ ∅ then T ← RewireConnection(x, Y_near)
10 if V_op = ∅ ∩ σ = ∅ then return Failure
11 z ← arg min_{y ∈ V_op} {Cost_T(y)}
12 return {z, T}
```

```
1  function RewireConnection(x, Y_near)
2  H_near ← ∅
3  for y ∈ Y_near do
4      if T.parent(y) ≠ T.parent(x) then
5          H_near ← H_near ∪ {y}
6      for h ∈ H_near do
7          if Cost_T(x) + Cost(x, h) < Cost_T(h) then
8              if CollisionFree(x, h) then
9                  T.parent(h) ← x; UpdateChildCosts(h)
10 return T
```

The ExpandTree(z), as shown in Algorithm 3, is introduced from FMT* [21] except for RewireConnection(x, Y_{near}) used in the dynamic optimal search in Algorithm 5. The lowest-cost open node z is selected to search for its neighbor unvisited node set X_{near}. For each $x \in X_{\text{near}}$, its neighbor open node set Y_{near} is got to determine the lowest-cost node $y_{\min}$. If the connection between $y_{\min}$ and x is valid, the x with $y_{\min}$ as its parent is sent to the set V'_{op} which will be added to the tree T and V_{op} once all x samples in X_{near} have been considered. It is noteworthy that the x samples in V'_{op}, not in V_{op}, will not be connected to the remaining samples in X_{near}. Thereafter, the node z is removed from the open set V_{op} and added to the closed set V_{c}. RewireConnection(x, Y_{near}) is able to improve the spanning tree if a path σ exists, we will explain it later. The batch sampling search returns Failure when V_{op} and σ are empty.

Algorithm 4 InsertNode(T)

```
1  s ← SampleFree(1); W_near ← Near(V_i, s, r_n) ∩ V_c
2  while W_near ≠ ∅ do
3      x_min ← arg min_{x ∈ W_near} {Cost_T(x) + Cost(x, s)}
4      if CollisionFree(x_min, s) then
5          T.parent(s) ← x_min
6          V_op ← V_op ∪ {x_min}; z ← x_min
7          break
8      else then W_near ← W_near\{x_min}
9  return T
```

The InsertNode(T) shown in Algorithm 4 determines a new node s from $\mathcal{X}_{\text{free}}$ and activates the neighbor closed nodes W_{near}. The new node is added to the existing tree in an optimal way. In addition, the new node can be regarded as a bridge between the closed nodes and the unvisited nodes to help the ExpandTree(z) expand the tree again.

C. Dynamic Optimal Search

The proposed dynamic optimal search, balancing the features of the "lazy" and "non-lazy" searches, asymptotically improves the existing tree to find a better path until the optimal path is obtained, as shown in Fig. 2(d)-(f). We briefly list new functions and notions in Algorithm 5. Let Prune$(V_{i-1}, \text{Cost}_T(x_{\text{goal}}))$ be the function that prunes the nodes and edges in V_{i-1} by the informed technique determining a hyper-elliptic region in C-space, where V_{i-1} only contains closed nodes and unvisited nodes. Better paths may exist in the hyper-elliptic region that satisfies

$$\|x - x_{\text{init}}\| + \|x - x_{\text{goal}}\| \le J(\sigma_{i-1}). \tag{4}$$

$\text{Cost}_{\text{ig}}(v)$ denotes the sum of the lowest cost from x_{init} to v and from v to x_{goal}, ignoring obstacles. The function InformedSampleFree$(n, \text{Cost}_T(x_{\text{goal}}))$ samples n nodes in the hyper-elliptic region and the nodes not on the tree in V_i are added to V_{u} by UnconnectNodes(V_i).

Algorithm 5 shows the dynamic optimal search for IAFMT*. Prune$(V_{i-1}, \text{Cost}_T(x_{\text{goal}}))$ deletes the nodes and edges outside the hyper-elliptic sub-region given by the informed technique and InformedSampleFree$(n, \text{Cost}_T(x_{\text{goal}}))$ provides n new nodes in the sub-region. V_{u} includes the new nodes in the *i*'th iteration and the unvisited nodes in the (*i*-1)'th iteration after executing UnconnectSamples(V_i). It is noted that all the closed nodes in V_i are reopened when the dynamic optimal search is initialized. The dynamic optimal search asymptotically improves the tree and tries to find a lower-cost path at any time. The function RewireConnection(x, Y_{near}) in Algorithm 3 makes the ExpandTree(z) can improve the searching tree. The ExpandTree(z) employs the "lazy" search, a neighbor unvisited node x is simply skipped if a connection between the x and its neighbor open node intersects an obstacle, to connect the nodes in V_{u} to the tree (Alg. 3, line 2-8) as the dynamic optimal search is executed. Thereafter, the "non-lazy" search is conducted by the RewireConnection(x, Y_{near}) to rewire every neighbor open node h, which connects the lowest-cost node as its parent node.

Algorithm 5 DynamicOptimalSearch(V_{i-1}, T)

```
V_p ← Prune(V_{i-1}, Cost_T(x_goal)); n ← (1/2) × Size(V_p)
V_i ← V_p ∪ InformedSampleFree(n, Cost_T(x_goal))
V_u ← UnconnectSamples(V_i)
{V_op, V_c, T, z, N_z} ← Initialize(V_i, V_u, E)
while V_op ≠ ∅ do
    {z, T} ← ExpandTree(z); σ ← Path(x_goal, T)
return σ
```

```
function Prune(V_{i-1}, Cost_T(x_goal))
    V_p ← ∅
    for v ∈ V_{i-1} do
        if Cost_ig(v) ≤ Cost_T(x_goal) then V_p ← V_p ∪ {v}
return V_p
```

IV. Analysis

A. Probabilistic Completeness

The proposed algorithm IAFMT* ensures probabilistic completeness which means that the probability of solving a problem goes to 1 as the number of samples approaches infinity.

Theorem 1. IAFMT* is a probabilistically complete algorithm. For a given motion planning problem, the probability of searching for a feasible path is as follows.

$$\lim_{n\to\infty} \mathrm{P}\left[\left\{x_{\mathrm{goal}} \in V_i \cap \mathcal{X}_{\mathrm{goal}} \text{ in } T\right\}\right] = 1 \tag{5}$$

Proof. The following three arguments are used to prove the theorem 1: (1) A batch of nodes including x_{init} and x_{goal} are randomly sampled from $\mathcal{X}_{\mathrm{free}}$; (2) The batch sampling search starts to build a searching tree from x_{init} and expands the tree with the lowest-cost node in T as the selected node z; (3) The single sampling search helps the batch sampling search explore the area where the distance from any node to the tree nodes is longer than r_n. Therefore, IAFMT* performs the hybrid incremental search to steadily grow a searching tree outward from x_{init}. The probability of finding a feasible path approaches to one as the number of samples n goes to infinity. So it is stated that IAFMT* can guarantee probability completeness.

B. Asymptotic Optimality

Let $\sigma^{\mathcal{T}}: [0,1] \longmapsto \mathcal{X}$ be a tracing path that traces the path σ. The cost of $\sigma^{\mathcal{T}}$ is bounded as

$$J(\sigma^{\mathcal{T}}) \leq (1+\varepsilon)J(\sigma), \tag{6}$$

when $\sigma^{\mathcal{T}}$ approaches σ, where ε is a given small constant.

Theorem 2. IAFMT* is an AO algorithm. For an optimal motion planning problem as defined in Section II, IAFMT* makes a feasible path σ converge in probability to the optimal path σ^* as $n \to \infty$. Specifically, for any $\varepsilon > 0$,

$$\lim_{n\to\infty} \mathrm{P}[J(\sigma^{\mathcal{T}}) > (1+\varepsilon)J(\sigma^*)] = 0 \tag{7}$$

Proof. The proof is based on [21] that probability is bounded as $O(n^{-\frac{\eta}{d}}\log^{-\frac{1}{d}} n)$ if $\sigma^{\mathcal{T}}$ cannot approach σ as $n \to \infty$. Finding a tracing path $\sigma^{\mathcal{T}'}$ by IAFMT* approximates σ^* with

$$J\left(\sigma^{\mathcal{T}'}\right) > (1+\varepsilon/3)J(\sigma^*). \tag{8}$$

Obviously, for any $\varepsilon > 0$, we can obtain the inequality

$$(1+\varepsilon/3)J\left(\sigma^{\mathcal{T}'}\right) > (1+\varepsilon/3)^2 J(\sigma^*). \tag{9}$$

As n is large enough, a tracing path $\sigma^{\mathcal{T}}$ approaching $\sigma^{\mathcal{T}'}$ can be found and deduce the following inequality

$$\mathrm{P}[J(\sigma^{\mathcal{T}}) > (1+\varepsilon/3)^2 J(\sigma^*)] < \mathrm{P}\left[J(\sigma^{\mathcal{T}}) > (1+\varepsilon/3)J\left(\sigma^{\mathcal{T}'}\right)\right], \tag{10}$$

where

$$\mathrm{P}\left[J(\sigma^{\mathcal{T}}) > (1+\varepsilon/3)J\left(\sigma^{\mathcal{T}'}\right)\right] = O(n^{-\frac{\eta}{d}}\log^{-\frac{1}{d}} n) \tag{11}$$

by [21]. Let $\sigma^{\mathcal{T}}$ returned by IAFMT* approximate σ^*, for any $\eta \geq 0$,

$$\lim_{n\to\infty} \mathrm{P}[J(\sigma^{\mathcal{T}}) > (1+\varepsilon/3)^2 J(\sigma^*)] < \lim_{\mathrm{n}\to\infty} \mathrm{O}\left(n^{-\frac{\eta}{d}}\log^{-\frac{1}{d}} n\right) = 0. \tag{12}$$

If $\varepsilon \leq 3$, the inequality can be obtained

$$(1+\varepsilon/3)^2 \leq (1+\varepsilon), \tag{13}$$

which follows that

$$\{J(\sigma^{\mathcal{T}}) > (1+\varepsilon)J(\sigma^*)\} \subset \{J(\sigma^{\mathcal{T}}) > (1+\varepsilon/3)^2 J(\sigma^*)\}. \tag{14}$$

Hence,

$$\lim_{n\to\infty} \mathrm{P}[J(\sigma^{\mathcal{T}}) > (1+\varepsilon)J(\sigma^*)] \leq \lim_{\mathrm{n}\to\infty} \mathrm{P}[\mathrm{J}(\sigma^{\mathcal{T}}) > (1+\varepsilon/3)^2 \mathrm{J}(\sigma^*)] = 0. \tag{15}$$

If $\varepsilon > 3$, the above statement still holds due to the probability is monotone-decreasing about ε. Therefore, IAFMT* is an AO algorithm, as claimed.

C. Computational Complexity

The space complexity $\mathrm{SC}_n^{\mathrm{IAFMT*}}$ as the amount of memory space occupied by IAFMT* in iteration *n*. The time complexity $\mathrm{TC}_n^{\mathrm{IAFMT*}}$ is defined as the number of calls to the most time-consuming IAFMT* function in iteration *n*.

Theorem 3. $\mathrm{SC}_n^{\mathrm{IAFMT*}} \in O(n\log n)$.

Proof. V_i, V_{u}, V_{op}, V_{c}, V_{p}, E and T require $O(n)$ space, respectively. N_x, N_z, X_{near}, Y_{near}, H_{near} and W_{near} require $O(\log n)$ space for one node, respectively. $O(n\log n)$ space will be occupied to save these variables for up to n nodes. Therefore, $\mathrm{SC}_n^{\mathrm{IAFMT*}} \in O(n\log n)$.

Theorem 4. $\mathrm{TC}_n^{\mathrm{IAFMT*}} \in O(n\log n)$.

Proof. Lines 2-8 are run $O(\log n)$ times [21] in Algorithm 3 and RewireConnection(·) takes $O(\log n)$ time for one node, so executing ExpandTree(·) once takes $O(\log n)$ time. Similarly, InsertNode(·) takes $O(\log n)$ time. $O(\log n)$ time is required to run HybridIncrementalSearch(·) and DynamicOptimalSearch(·) once, respectively. Hence, IAFMT* takes $O(n\log n)$ time for n nodes.

V. Experiments

A. Experiments in OMPL Benchmark

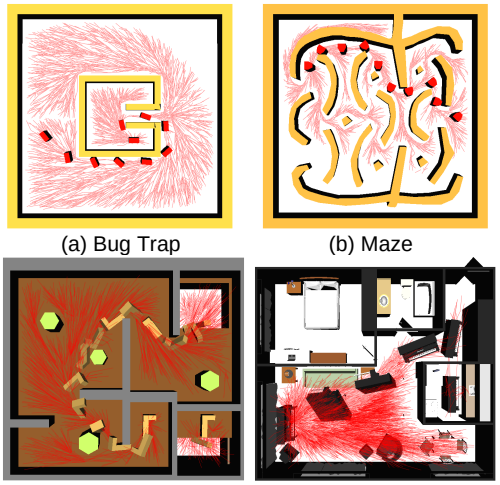
1) Simulation Setup

We provide numerical experiments to evaluate the

performance of IAFMT* by comparing it with three state-of-the-art algorithms, namely PRM*, Informed RRT*, and FMT*. All algorithms are run on a 3.7GHz Intel Core i7-8700K CPU with 16GB of memory and tested in the Open Motion Planning Library (OMPL) [30] v1.4.0 benchmark. Four OMPL test scenarios are considered: two 2D scenarios, "Bug Trap" and "Maze", in the $\mathbb{SE}(2)$ C-space as well as two 3D scenarios, "Cubicles" and "Apartment", in the $\mathbb{SE}(3)$ C-space, as illustrated in Fig. 3. Setting runtime t_{given} and cost threshold J_{given} in each scenario, as shown in Table I, can assess the ability of converging to a high-quality solution in limited resources by test algorithms. The default OMPL settings are employed for PRM* and Informed RRT*. We use the default value $r_n = 1.1$ for FMT* and IAFMT*. $n = 1000$ is given as the initial number of samples for IAFMT* in all scenarios.

TABLE I
RUNTIME AND COST THRESHOLD

Scenario	t_{given}(s)	J_{given}
Bug Trap	10	130
Maze	10	130
Cubicles	100	1800
Apartment	300	500



(a) Bug Trap (b) Maze (c) Cubicles (d) Apartment

Fig. 3. The planning paths generated by IAFMT* in OMPL test scenarios.

TABLE II
SIMULATION RESULTS FOR ALL ALGORITHMS

Scenario	Algorithm	t_{avg}(s)	J_{avg}	Sol (%)	Opt (%)
Bug Trap	PRM*	4.53	129.21	98.0	98.0
	I-RRT*	1.40	129.29	100.0	100.0
	FMT*	0.55	127.38	90.0	79.5
	IAFMT*	0.41	128.91	100.0	100.0
Maze	PRM*	3.01	123.84	100.0	90.0
	I-RRT*	3.70	123.25	100.0	98.0
	FMT*	1.09	111.19	100.0	95.0
	IAFMT*	0.99	115.71	100.0	100.0
Cubicles	PRM*	100.11	1914.18	100.0	0.0
	I-RRT*	50.01	1796.74	100.0	98.0
	FMT*	12.13	1794.43	99.6	72.3
	IAFMT*	15.86	1795.66	100.0	100.0
Apartment	PRM*	199.86	497.86	60.0	36.0
	I-RRT*	150.80	419.97	30.0	30.0
	FMT*	59.78	449.10	87.5	79.1
	IAFMT*	43.69	471.78	100	100

In order to compare the non-anytime AO algorithm FMT* with the other three anytime AO algorithms, we vary the given number of samples for FMT* in different scenarios. Sample counts are varied from 1000 to 10000 points in "Bug Trap" and "Maze", from 1000 to 50000 points in "Cubicles", from 1000 to 80000 points in "Apartment". FMT* runs 20 times at every sample counts. Other anytime AO algorithms run 50 times in each scenario.

2) *Results and Discussion*

The simulation results are shown in Table II, where t_{avg} is the average runtime, J_{avg} is the average path cost, and I-RRT* denotes Informed RRT*. In terms of computational efficiency, i.e., t_{avg}, the performance of FMT* and IAFMT* substantially outperform PRM* and Informed RRT*, and that of IAFMT* is slightly better than FMT* in general. All algorithms can give planning paths with lower J_{avg} than J_{given} in each scenario except J_{avg} of PRM* is 1914.18 in "Cubicles" scenario, which means that PRM* cannot find low-cost paths. Let Sol in Table II be the success rate of obtaining the feasible path by the algorithms. And Opt is the success rate of getting the high-quality path satisfying the given cost threshold J_{given} by the algorithms. The success rate of obtaining the feasible path and high-quality path by IAFMT* is obviously higher than the other algorithms in all scenarios. In the most challenging "Apartment" scenario, the success rate of obtaining the high-quality path by IAFMT* is 64%, 70%, and 20.9% higher than PRM*, Informed RRT*, and FMT*, respectively, which demonstrates the advantages of IAFMT* in the capability of solving a high-quality solution. IAFMT* also shows the self-adaptive ability due to the success rates of achieving the feasible path and the high-quality path are 100% in all scenarios.

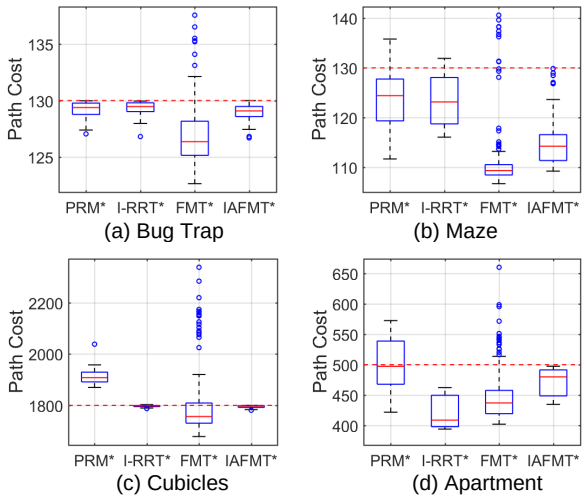


Fig. 4. The path cost given by each algorithm.

Fig. 4 and Fig. 5 use the box plots to visually show the statistical information for computation time and path cost given by each algorithm, where the red dashed lines denote J_{given} and the small circles are data outliers. As presented in Fig. 4, data points of path cost generated by IAFMT* are all under the line of J_{given}, and the points are closer to the given line except in "Maze" scenario due to the low-cost paths are easy to be found

by IAFMT. Hence, IAFMT* can save lots of computational resources while guaranteeing high-quality solutions. In addition, the data points of path cost by IAFMT* is denser in general, which indicates that the performance of obtaining high-quality paths is more stable than the other algorithms. The same features of data points are also shown in Fig. 5. The lower and denser data points of computation time given by IAFMT* show that IAFMT* has the capability to converge to a high-quality solution rapidly and stably.

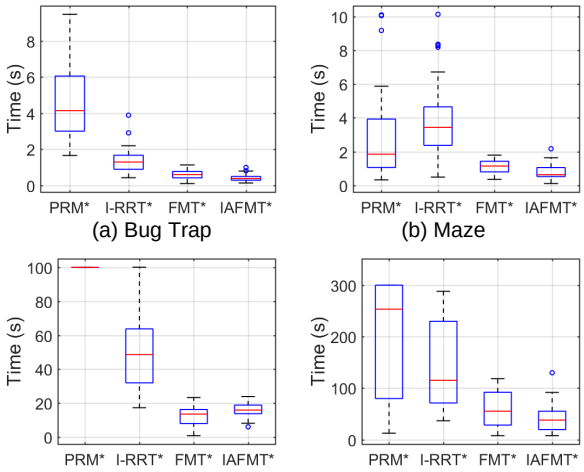


Fig. 5. The computation time consumed by each algorithm.

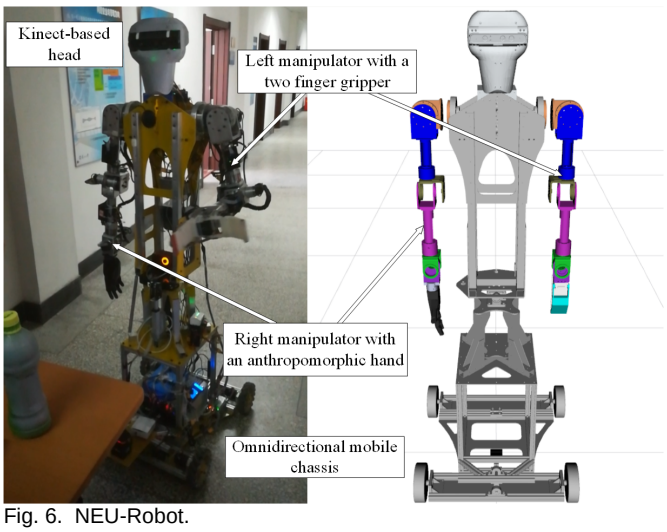


Fig. 6. NEU-Robot.

B. Experiments on Robot Planning Tasks

1) Experiment Setup

A simulation and a pick-and-place experiment on NEU-Robot equipped with 6-DOF manipulators, as shown in Fig. 6, are carried out to evaluate the performance of IAFMT* in application. We also use the four algorithms, run on a 2.3GHz Intel Core i7-3610QE CPU with 8GB of memory, to plan the trajectory of the manipulators. For the end effectors of the manipulators, position precision and orientation precision are 0.01 meters and 0.1 radians. It is the same as before for the parameters of PRM* and Informed RRT*. The search radius r_n is also 1.1 for FMT* and IAFMT*. Each algorithm runs 10 times to plan the trajectory for the manipulators. We do not set the cost threshold J_{given} in all experiments so as to test the ability of the algorithms to explore the high-quality trajectory.

In the simulation of the right manipulator, as shown in Fig. 7(a), the given runtime t_{given} is 5 seconds and 50 randomized boxes are added to the environment as fixed obstacles. We give variable sample counts of FMT* from the order of 1000 to 5500 points and set initial sample counts of IAFMT* to 1000 points. In addition, initial sample count n is set from 1000 to 5000 in order to investigate the influence of variable n on the performance of IAFMT*. The pick-and-place process of the left manipulator consists of four phases, as shown from (b) to (e) in Fig. 7. The given runtime is set to 5 seconds in the pick and reset phases, it is 1 second in the hold and place phases, and the total planning time is limited in 12 seconds. The sample counts for FMT* are given from 1000 to 10000 points and the initial sample counts for IAFMT* are always 1000 points.

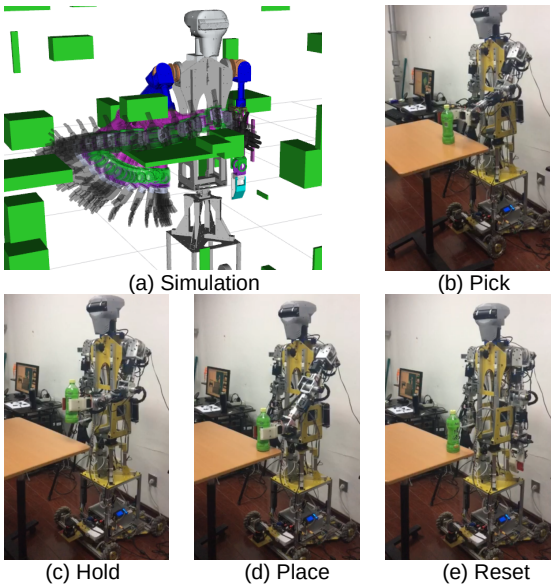


Fig. 7. NEU-Robot motion planning realized by IAFMT*.

TABLE III
EXPERIMENT RESULTS FOR ROBOT MOTION PLANNING

Experiment	Success rate			
	PRM*	I-RRT*	FMT*	IAFMT*
Simulation	8/10	6/10	8/10	8/10
Pick-and-place	6/10	2/10	8/10	9/10

2) Results and Discussion

The algorithms only need to plan one trajectory in the randomized simulated clutter scene, while they try to plan four different trajectories in succession in the real world, which means that the motion planning is failed if one of the four trajectories fails to plan. Experiment results in Table III show that the planning success rate of Informed RRT* is reduced by 40% with the increase of problem difficulty, and IAFMT* performs well in all experiments due to its self-adaptive ability. Besides, the low computational load of IAFMT* leads to high convergence rate even on the low-end computer. It is noted that FMT* shows good performance similar to IAFMT*, but FMT* is not a self-adaptive algorithm as the parameter, i.e., the sample count, is tuned manually according to experience.

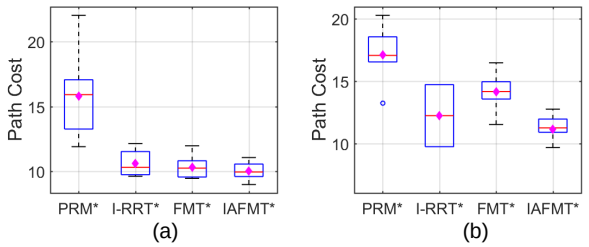


Fig. 8. The experiment results. (a) Path cost in the simulation. (b) Path cost in the pick-and-place experiment.

The ability of algorithms to explore the high-quality trajectory is presented in Fig. 8(a) and (b). The visual statistical information for path cost shows that the average and median values given by IAFMT* are the smallest than the other algorithms and its data points are denser, which indicates that IAFMT* can stably plan the high-quality trajectories. Fig. 9 presents the performance of every algorithm in each phase of the pick-and-place experiment. All algorithms plan a similar trajectory in the hold and place phases, while IAFMT* performs well in the pick and reset phases due to its outstanding motion planning ability.

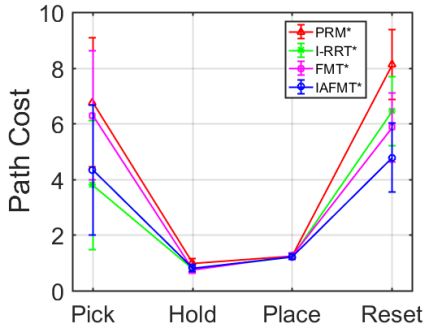


Fig. 9. Path cost in each phase.

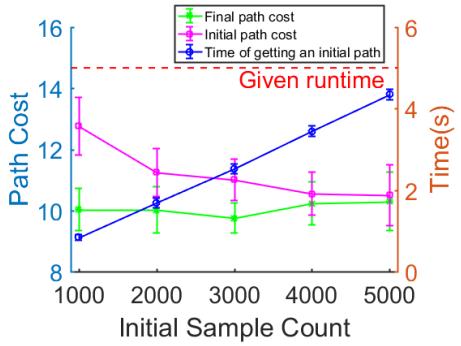


Fig. 10. The performance of IAFMT* tested by varying initial sample counts in the simulation.

Furthermore, we change the parameter of IAFMT* in the simulation to investigate the influence of variable initial sample counts on the performance. The success rate of the motion planning, solved by IAFMT* with initial sample count $n = 1000$, is 80% and the planning success rate is 100% as n is from 2000 to 5000. As shown in Fig. 10, when n is set to 1000, IAFMT* can fast search for an initial path with the high cost and it iteratively improves the solution to yield a final path with the low cost until the computational time reaches the given runtime. With the increase of the initial sample counts, there are denser initial samples in the C-space, which leads to the increase in time of getting an initial path and the low cost of the initial path approaching the final-path cost. The experiment results show that IAFMT* with different initial sample counts can stably converge to a high-quality solution, which demonstrates that IAFMT* is highly self-adaptive.

VI. Conclusion

In this paper, an anytime asymptotically-optimal sampling-based algorithm, namely Informed Anytime Fast Marching Tree, is presented to solve motion planning problems, especially high-dimensional complex problems. This paper also gives the theoretical analysis of probabilistic completeness, asymptotic optimality, and computational complexity on the proposed algorithm. The challenging simulation and experimental results verify that the proposed algorithm can converge to a high-quality solution with an efficient, stable, and self-adaptive performance.

In the future, we will introduce the GPU-based parallel computing technique into IAFMT* to achieve real-time motion planning and it is interesting to evaluate the performance of IAFMT* without knowledge of the geometry of obstacles and environments.

References

[1] T. Faulwasser, T. Weber, P. Zometa and R. Findeisen, “Implementation of Nonlinear Model Predictive Path-Following Control for an Industrial Robot,” *IEEE Trans. Contr. Syst. Technol.*, vol. 25, DOI 10.1109/Tcst.2016.2601624, no. 4, pp. 1505-1511, Jul. 2017.

[2] T. F. Shu, S. Gharaaty, W. F. Xie, A. Joubair and I. A. Bonev, “Dynamic Path Tracking of Industrial Robots With High Accuracy Using Photogrammetry Sensor,” *IEEE/Asme Trans. Mech.*, vol. 23, DOI 10.1109/Tmech.2018.2821600, no. 3, pp. 1159-1170, Jun. 2018.

[3] B. Lu, H. K. Chu, K. C. Huang and L. Cheng, “Vision-Based Surgical Suture Looping Through Trajectory Planning for Wound Suturing,” *IEEE Trans. Autom. Sci. Eng.*, vol. 16, DOI 10.1109/Tase.2018.2840532, no. 2, pp. 542-556, Apr. 2019.

[4] N. G. Yu, Y. J. Zhai, Y. H. Yuan and Z. X. Wang, “A Bionic Robot Navigation Algorithm Based on Cognitive Mechanism of Hippocampus,” *IEEE Trans. Autom. Sci. Eng.*, vol. 16, DOI 10.1109/Tase.2019.2909638, no. 4, pp. 1640-1652, Oct. 2019.

[5] H. Yang, J. Qi, Y. C. Miao, H. X. Sun and J. H. Li, “A New Robot Navigation Algorithm Based on a Double-Layer Ant Algorithm and Trajectory Optimization,” *IEEE Trans. Ind. Electron*, vol. 66, DOI 10.1109/Tie.2018.2886798, no. 11, pp. 8557-8566, Nov. 2019.

[6] J. Yuan, S. K. Yang and J. X. Cai, “Consistent Path Planning for On-Axle-Hitching Multisteering Trailer Systems,” *IEEE Trans. Ind. Electron*, vol. 65, DOI 10.1109/Tie.2018.2823691, no. 12, pp. 9625-9634, Dec. 2018.

[7] H. Cheon and B. K. Kim, “Online Bidirectional Trajectory Planning for Mobile Robots in State-Time Space,” *IEEE Trans. Ind. Electron.*, vol. 66, DOI 10.1109/tie.2018.2866039, no. 6, pp. 4555-4565, Jun. 2019.

[8] P. P. Cai, I. Chandrasekaran, J. M. Zheng and Y. Y. Cai, “Automatic Path Planning for Dual-Crane Lifting in Complex Environments Using a Prioritized Multiobjective PGA,” *IEEE Trans. Ind. Inform.*, vol. 14, DOI 10.1109/Tii.2017.2715835, no. 3, pp. 829-845, Mar. 2018.

[9] S. M. LaValle and J. J. Kuffner, “Randomized kinodynamic planning,” *Int. J. Robot. Res.*, vol, 20. DOI 10.1177/02783640122067453, no. 5, pp. 378-400, May 2001.

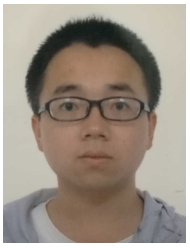
[10] L. E. Kavraki, P. Svestka, J. C. Latombe and M. H. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” *IEEE Trans. Robot. Autom.*, vol. 12, DOI 10.1109/70.508439, no. 4, pp. 566-580, Aug. 1996.

[11] J. Suh, J. Gong and S. Oh, “Fast Sampling-Based Cost-Aware Path Planning With Nonmyopic Extensions Using Cross Entropy,” *IEEE Trans. Robot.*, vol. 33, DOI 10.1109/Tro.2017.2738664, no. 6, pp. 1313-1326, Dec. 2017.

[12] B. F. Ye, Q. Tang, J. Yao and W. X. Gao, "Collision-Free Path Planning and Delivery Sequence Optimization in Noncoplanar Radiation Therapy," *IEEE Trans. Cyber.*, vol. 49, DOI 10.1109/Tcyb.2017.2763682, no. 1, pp. 42-55, Jan. 2019.
[13] I. B. Jeong, S. J. Lee and J. H. Kim, "Quick-RRT*: Triangular inequality-based implementation of RRT* with improved initial solution and convergence rate," *Expert Syst. Appl.*, vol. 123, DOI 10.1016/j.eswa.2019.01.032, pp. 82-90, Jun. 2019.
[14] M. Elbanhawi and M. Simic, "Sampling-Based Robot Motion Planning: A Review," *IEEE Access*, vol. 2, DOI 10.1109/Access.2014.2302442, pp. 56-77, Jan. 2014.
[15] Y. Li, R. X. Cui, Z. J. Li and D. M. Xu, "Neural Network Approximation Based Near-Optimal Motion Planning With Kinodynamic Constraints Using RRT," *IEEE Trans. Ind. Electron.*, vol. 65, DOI 10.1109/Tie.2018.2816000, no. 11, pp. 8718-8729, Nov. 2018.
[16] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, DOI 10.1177/0278364911406761, no. 7, pp. 846-894, Jun. 2011.
[17] M. Zucker *et al.*, "CHOMP: Covariant Hamiltonian optimization for motion planning," *Int. J. Robot. Res.*, vol. 32, DOI 10.1177/0278364913488805, no. 9-10, pp. 1164-1193, Aug. 2013.
[18] S. Choi, K. Lee and S. Oh, "Gaussian Random Paths for Real-Time Motion Planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst*, DOI 10.1109/IROS.2016.7759237, pp. 1456-1461, Oct. 2016.
[19] J. D. Gammell, T. D. Barfoot and S. S. Srinivasa, "Informed Sampling for Asymptotically Optimal Path Planning," *IEEE Trans. Robot.*, vol. 34, DOI 10.1109/Tro.2018.2830331, no. 4, pp. 966-984, Aug. 2018.
[20] J. D. Gammell, S. S. Srinivasa and T. D. Barfoot, "Batch Informed Trees (BIT*): Sampling-based Optimal Planning via the Heuristically Guided Search of Implicit Random Geometric Graphs," in *Proc. IEEE Int. Conf. Robot. Autom.*, DOI 10.1109/ICRA.2015.7139620, pp. 3067-3074, Jul. 2015.
[21] L. Janson, E. Schmerling, A. Clark and M. Pavone, "Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions," *Int. J. Robot. Res.*, vol. 34, DOI 10.1177/0278364915577958, no. 7, pp. 883-921, Jun. 2015.
[22] J. A. Starek, J. V. Gomez, E. Schmerling, L. Janson, L. Moreno and M. Pavone, "An Asymptotically-Optimal Sampling-Based Algorithm for Bi-directional Motion Planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, DOI 10.1109/IROS.2015.7353652, pp. 2072-2078, Dec. 2015.
[23] E. Schmerling, L. Janson and M. Pavone, "Optimal Sampling-Based Motion Planning under Differential Constraints: the Driftless Case," in *Proc. IEEE Int. Conf. Robot. Autom.*, DOI 10.1109/ICRA.2015.7139514, pp. 2368-2375, Jul. 2015.
[24] Z. Tahir, A. H. Qureshi, Y. Ayaz and R. Nawaz, "Potentially guided bidirectionalized RRT* for fast optimal path planning in cluttered environments," *Robot. Auton. Syst.*, vol. 108, DOI 10.1016/j.robot.2018.06.013, pp. 13-27, Oct. 2018.
[25] B. Sakcak, L. Bascetta, G. Ferretti and M. Prandini, "Sampling-based optimal kinodynamic planning with motion primitives," *Auton. Robot.*, vol. 43, DOI 10.1007/s10514-019-09830-x, no. 7, pp. 1715-1732, Oct. 2019.
[26] J. D. Marble and K. E. Bekris, "Asymptotically Near-Optimal Planning With Probabilistic Roadmap Spanners," *IEEE Trans. Robot.*, vol. 29, DOI 10.1109/Tro.2012.2234312, no. 2, pp. 432-444, Apr. 2013.
[27] O. Salzman and D. Halperin, "Asymptotically Near-Optimal RRT for Fast, High-Quality Motion Planning," *IEEE Trans. Robot.*, vol. 32, DOI 10.1109/Tro.2016.2539377, no. 3, pp. 473-483, Jun. 2016.
[28] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli and S. Teller, "Anytime Motion Planning using the RRT*," in *Proc. IEEE Int. Conf. Robot. Autom.*, DOI 10.1109/ICRA.2011.5980479, pp. 1478-1483, May 2011.
[29] O. Salzman and D. Halperin, "Asymptotically-optimal motion planning using lower bounds on cost," in *Proc. IEEE Int. Conf. Robot. Autom.*, DOI 10.1109/ICRA.2015.7139773, pp. 4167-4172, Jul. 2015.
[30] I. A. Sucan, M. Moll and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robot. Autom. Mag.*, vol. 19, DOI 10.1109/Mra.2012.2205651, no. 4, pp. 72-82, Dec. 2012.

Jing Xu was born in Yingkou, China. He received his B.S., M.S. degrees in School of Mechanical Engineering, Liaoning Shihua University, Fushun, China, in 2013 and 2016, respectively. He is currently pursuing the PhD degree with School of Mechanical Engineering and Automation, Northeastern University, Shenyang, China. His research interests include robot kinematics, robot motion planning, robot control, service robotics and parallel robot system.

Kechen Song received the B.S., M.S. and Ph.D. degrees in School of Mechanical Engineering and Automation, Northeastern University, Shenyang, China, in 2009, 2011 and 2014, respectively. During 2018-2019, he was an Academic Visitor in the Department of Computer Science, Loughborough University, UK. He is currently an Associate Professor in the School of Mechanical Engineering and Automation, Northeastern University. His research interest covers vision-based inspection system for steel surface defects, surface topography, image processing, pattern recognition and robotics.



Defu Zhang received the B.S. degree in School of Mechanical Engineering and Automation, Hebei University, Baoding, China, in 2015, and the M.S. degree in School of Mechanical Engineering and Automation, Northeastern University, Shenyang, China, in 2017. He is currently pursuing the Ph.D. degree with School of Mechanical Engineering and Automation, Northeastern University, China. His research interests include deep learning with sample lack and semantic segmentation.

Hongwen Dong received the B.S. degree in School of Mechanical Engineering and Automation, Liaoning University of Technology, Jinzhou, China, in 2016, and the M.S. degree in School of Mechanical Engineering and Automation, Northeastern University, Shenyang, China, in 2018. He is currently pursuing the Ph.D. degree with School of Mechanical Engineering and Automation, Northeastern University, China. His research interests include deep learning, pattern recognition and semantic segmentation.

Yunhui Yan received the B.S., M.S. and Ph.D. degrees in School of Mechanical Engineering and Automation, Northeastern University, Shenyang, China, in 1981, 1985 and 1997, respectively. He has been a teacher in Northeastern University of China since 1982, and became as professor in 1997. During 1993-1994, he stayed in the Tohoku National Industrial Research Institute as a visiting scholar. His research interest covers intelligent inspection, image processing, pattern recognition and robotics.

Qinggang Meng (M'06–SM'18)received the B.S. and M.S. degrees from the School of Electronic Information Engineering, Tianjin University, China, and the Ph.D. degree in computer science from Aberystwyth University, U.K. He is a Professor with the Department of Computer Science, Loughborough University, U.K. His research interests include biologically and psychologically inspired learning algorithms and developmental robotics, service robotics, robot learning and adaptation, multi-UAV cooperation, drivers distraction detection, human motion analysis and activity recognition, activity pattern detection, pattern recognition, artificial intelligence, and computer vision. He is a fellow of the Higher Education Academy, U.K.