IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received November 25, 2018, accepted December 5, 2018, date of publication December 11, 2018,
date of current version January 4, 2019.

*Digital Object Identifier 10.1109/ACCESS.2018.2886233*

# Tree-Search-Based Any-Time Time-Optimal Path-Constrained Trajectory Planning With Inadmissible Island Constraints

**PEIYAO SHEN[1,2], XUEBO ZHANG[1,2,3], (Senior Member, IEEE), AND YONGCHUN FANG[1,2], (Senior Member, IEEE)**
[1]Institute of Robotics and Automatic Information System, Nankai University, Tianjin 300071, China
[2]Tianjin Key Laboratory of Intelligent Robotics, Nankai University, Tianjin 300071, China
[3]Key Laboratory of Industrial IoT and Networked Control, Ministry of Education, Chongqing University of Posts and Telecommunications, Chongqing 400065, China

Corresponding author: Xuebo Zhang (zhangxuebo@nankai.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61573195 and Grant U1613210, and in part by the Open Project of the Key Laboratory of Industrial IoT and Networked Control Ministry of Education under Grant 2018FF07.

**ABSTRACT** Time-optimal path-constrained trajectory planning is of significant importance for enhancing the work efficiency of robotic systems. In particular, when inadmissible island constraints are considered, existing approaches are typically offline. In order to solve the problem online, this paper proposes a heuristic tree-search-based any-time time-optimal trajectory planning algorithm to achieve incremental computation of feasible trajectories. In a limited planning period, the proposed algorithm iteratively generates feasible trajectories, as optimal as possible, until the period terminates. This any-time performance of the proposed algorithm ensures that feasible trajectories can be obtained in real time and even time-optimal trajectories can be obtained if the planning period is long enough for searching. Experimental results on active-casters-based omnidirectional wheeled mobile robots demonstrate the validity of the proposed algorithm.

**INDEX TERMS** Any-time performance, time optimality, path-constrained trajectory planning, tree search, trap detection.

## I. INTRODUCTION

Trajectory planning aims to generate desired state evolution profiles for specific tasks of various robotic systems, such as manipulators [1], industrial biaxial gantries [2], aerial robots [3], fixed-wheeled mobile robots [4]–[6], omnidirectional wheeled mobile robots (OWMR) [7]–[9], while guaranteeing physical constraints and optimizing some performance indexes. The work in [10] summarizes four classical planners including reactive [1], [11], hierarchical [12], coupled [13], [14] and decoupled [15], [16] strategies.

As a trade-off between optimality and computational complexity, the decoupled strategies [17], [18] are widely used to divide the planning problem into two stages. In the first stage, *path planning* searches a geometric path which guarantees obstacle avoidance, curvature constraints [19]–[21], and so on. In the second stage, *path-constrained trajectory planning* transforms a given path in the first stage into an optimal trajectory satisfying kinematic and dynamic constraints. The common optimization objectives include jerk minimization [22], energy consumption minimization [23], traveling time minimization [24]–[27], and so on.

In order to enhance work efficiency of robotic systems, the time-optimal path-constrained trajectory planning (TOPCTP) has received significant attention. Existing literatures give special attention to two essential properties of TOPCTP: time optimality and real-time performance. On one hand, some approaches achieve the real-time performance at the expense of time optimality [28]. The work in [29] proposes an online method considering torque constraints. Under jerk constraints, the proposed method in [30] generates a near time-optimal trajectory in bounded computation time. The work in [31] generates feasible trajectories for mobile manipulators by solving a constrained sequential linear quadratic optimal control problem in real time. The work in [32] proposes a real-time approach to obtain a near time-optimal trajectory of OWMR vehicles with orthogonal wheels. Recently, the work in [33] presents a general online planning framework to generate feasible trajectories whose mathematical function is unknown in advance. Meanwhile, the work in [34] proposes an online trajectory planner, which removes the restrictions that upper and lower bounds of acceleration are positive and negative.

On the other hand, some approaches ensure the time optimality, while they cannot achieve the real-time performance [35], [36]. In an offline manner, a time-optimal trajectory is generated by using bi-directional scan methods under acceleration and jerk constraints [16]. With the aid of dynamic programming techniques [37], [38], time-optimal trajectories are generated on the phase plane $(s, \dot{s})$ with $s$ being path coordinate and $\dot{s}$ being path velocity. The work in [18] adopts the projection operator Newton method [39] to obtain time-optimal trajectories for quadrotors. Based on Pontryagin Maximum Principle, the work in [40], [41] presents a numerical integration (NI) method to generate time-optimal trajectories which possess a bang-bang structure of torque inputs. In order to follow the reference trajectory, the works in [42] and [43] design orbitally stabilizing controllers and model predictive controllers, respectively. Recently, a fast and open source implementation of this NI method is provided in [44], yet it ignores inadmissible island constraints in admissible regions. The work [40] first describes these inadmissible island constraints, which are caused by friction and copper losses in drive motors and planning task constraints. They address these inadmissible island constraints in an offline manner by constructing a dense directed graph. Accordingly, in the presence of both bounded velocity and torque, the work in [45] provides failure conditions of this NI method with detailed proofs. The works in [46] and [47] introduce a concept of 'trap region' to address the failure issue, however, they still cannot ensure online performance while satisfying inadmissible island constraints. The existence of inadmissible islands brings nonconvex admissible region on the plane $(s, \dot{s})$, and it increases computational complexity of TOPCTP.

To achieve online planning with inadmissible island constraints, we present a new tree-search-based any-time time-optimal path-constrained trajectory planning algorithm. It can generate a feasible solution in real time and it can even compute the time-optimal trajectories if the planning time is sufficiently long. Different from existing literatures, the presented approach first introduces heuristic tree search and combines it with trap region detection techniques to achieve any-time performance, which indicates that a planning algorithm does incremental computation to quickly generate feasible trajectories and optimize these trajectories as much as possible before the end of one planning period. Specially, one heuristic search tree is built and extended from the starting point on the $(s, \dot{s})$ plane. This extension prefers to select leaf nodes with the highest heuristic value to generate new tree edges which avoid inadmissible islands. In the extension, one feasible trajectory consisting of tree edges is obtained until the terminal point is reached. Then, the heuristic value of all existing leaf nodes are recalculated, and the search tree continues to extend until another feasible trajectory is generated. These iteratively generated trajectories tend to be closer to the time-optimal one before the end of one planning period. In order to shorten the search time of each iteration, the trap region attached to inadmissible islands is detected to abandon those unavailable tree edges and nodes. Although the works in [46] and [47] discover the trap region, we first propose a detection method of trap regions in the presence of inadmissible island constraints. Experimental results on active-casters-based OWMR vehicles verify the validity of the presented approach.

Compared with existing methods which cannot solve inadmissible island constraints in real time, *the major contributions of this paper are as follows*:

- The proposed approach first introduces heuristic tree search technique into TOPCTP with inadmissible island constraints to achieve iterative generation of better and better trajectories within one planning period, while existing methods just obtain one solution in offline mode.
- The proposed approach first detects the trap region attached to inadmissible islands to enhance the computation efficiency of tree search, while existing methods only compute trap regions without considering the presence of inadmissible island constraints.
- The proposed approach possesses any-time performance. Those iteratively generated trajectories tend to be time-optimal before the end of a planning period. Thus, one feasible trajectory as optimal as possible is obtained for a planning task, even in a limited planning period.

The remainder of this paper is divided into four sections. Section II gives a detailed description of the TOPCTP problem with inadmissible island constraints. Section III proposes an any-time time-optimal path-constrained trajectory planning algorithm for this problem. Section IV gives several experimental results on active-casters-based OWMR vehicles to validate the presented algorithm. Finally, Section V gives some conclusions.

## II. PROBLEM FORMULATION

### A. PROBLEM STATEMENT

In general, a $n$-dimensional robot pose $\boldsymbol{q}$ in a path-constrained trajectory planning task is represented as $\boldsymbol{q}(s)$ instead of $\boldsymbol{q}(t)$, wherein the scalars $t$, $s$ are time and path coordinate and obey a nonlinear scaling relation.

In order to guarantee velocity constraints of the robotic system, an inequality constraint of $s, \dot{s}$ is given as [45]:

$$\boldsymbol{A}(s)\dot{s} + \boldsymbol{D}(s) \leq \boldsymbol{0}, \tag{1}$$

wherein the scalar $\dot{s}$ is path velocity. The vectors $\boldsymbol{A}(s), \boldsymbol{D}(s)$ are obtained by the mathematical expression of joint velocities of the robotic system.

In order to satisfy acceleration constraints of the robotic system, an inequality constraint of $s, \dot{s}, \ddot{s}$ should hold as [45]:

$$\boldsymbol{A}(s)\ddot{s} + \boldsymbol{B}(s)\dot{s}^2 + \boldsymbol{C}(s) \leq \boldsymbol{0}, \tag{2}$$

wherein the scalar $\ddot{s}$ is path acceleration. The vectors $\boldsymbol{B}(s), \boldsymbol{C}(s)$ are obtained by the mathematical expression of joint accelerations of the robotic system.

With the aid of (2), the inequality of $\ddot{s}$ is obtained as

$$\alpha(s, \dot{s}) \leq \ddot{s} \leq \beta(s, \dot{s}), \tag{3}$$

where minimum path acceleration $\alpha(s, \dot{s})$ and maximum path acceleration $\beta(s, \dot{s})$ are computed as

$$\alpha(s, \dot{s}) = \max\{\frac{-B_i(s)\dot{s}^2 - C_i(s)}{A_i(s)}|A_i(s) < 0\},$$
$$\beta(s, \dot{s}) = \min\{\frac{-B_i(s)\dot{s}^2 - C_i(s)}{A_i(s)}|A_i(s) > 0\},$$

wherein the integer $i \in [1, m]$, with $m$ being the dimension of the vector $\boldsymbol{A}(s)$. The scalars $A_i(s), B_i(s), C_i(s)$ are elements of vectors $\boldsymbol{A}(s), \boldsymbol{B}(s)$ and $\boldsymbol{C}(s)$, respectively. The detailed description of (3) can be found in [40] and [45].

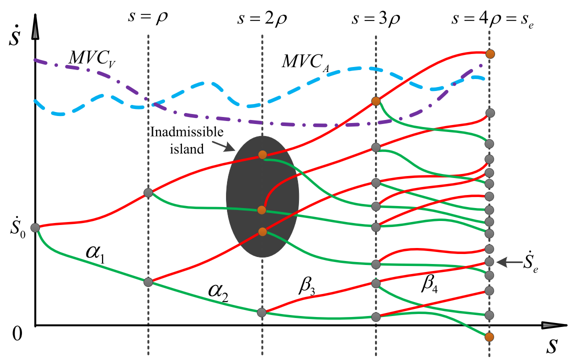


**FIGURE 1. Tree search: the parameters are set as $\delta = 2$, $\rho = s_e/4$, the circle points are tree nodes, and the solid curves are tree edges. Tree nodes: the gray points are in the admissible region and orange points are in the inadmissible region. The scalars $\dot{s}_0, \dot{s}_e$ are the starting and terminal velocities, respectively. Tree edges: the red curves are accelerating trajectories with $\beta(s, \dot{s})$ and green curves are decelerating trajectories with $\alpha(s, \dot{s})$. The black ellipse is the inadmissible island decided by planning tasks. The purple dash-dot and cyan dash curves represent the velocity and acceleration constraints of the robotic system, respectively.**

With the aid of (3), the maximum velocity curve (the cyan dash curve in Fig. 1), is obtained as

$$MVC_A(s) = \min\{\dot{s} \geq 0|\alpha(s, \dot{s}) = \beta(s, \dot{s})\}, \quad s \in [0, s_e], \tag{4}$$

with the scalar $s_e$ being the total length of a specified path. When path velocities are higher than $MVC_A(s)$, $\alpha(s, \dot{s})$ is greater than $\beta(s, \dot{s})$, and it indicates that the acceleration constraint (3) is violated.

With the aid of (1), the maximum velocity curve which guarantees velocity constraints (the purple dash-dot curve in Fig. 1), is obtained as

$$MVC_V(s) = \min\{-D_i(s)/A_i(s)|A_i(s) > 0\}, \tag{5}$$

with the integer $i \in [1, m]$, the scalar $D_i(s)$ being the element of the vector $\boldsymbol{D}(s)$.

The maximum velocity curve satifying velocity and acceleration constraints is computed as

$$MVC(s) = \min(MVC_A(s), MVC_V(s)), s \in [0, s_e]. \tag{6}$$

The admissible region on the plane $(s, \dot{s})$ is enclosed by the curve $MVC(s)$ and the lines $\dot{s} = 0$, $s = 0$, $s = s_e$. The work in [40] has reported that there exists the 'inadmissible island' in the admissible region, such as the black ellipse in Fig. 1.

#### 1) INADMISSIBLE ISLANDS

They are surrounded by admissible regions satisfying velocity and acceleration constraints of the robot system. In inadmissible islands, trajectories violate extra inequality constraints of path coordinate, path velocity and path acceleration, caused by friction and copper losses in joints of the robotic system or the requirement of planning tasks given by operating personnel. For example, a vehicle collision avoidance task at crossroad requires that a vehicle moving towards north drives with higher or lower speed than another vehicle moving towards west to pass through the crossroad safely, and the requirement is transformed into the plane $(s, \dot{s})$ as a rectangle inadmissible island. The more detailed description can be found in [40].

In the admissible region, the continuous curve from $(s = 0, \dot{s} = \dot{s}_0)$ to $(s = s_e, \dot{s} = \dot{s}_e)$, which satisfies the constraints (3) and avoids inadmissible islands, is called the feasible trajectory

$$f(s) : s \in [0, s_e] \rightarrow f \in [0, MVC(s)], \tag{7}$$

where the scalars $\dot{s}_0$ and $\dot{s}_e$ represent starting and terminal path velocities, respectively.

The traveling time of (7) is computed as

$$T = \int_0^{s_e} \frac{1}{f(s)} ds. \tag{8}$$

According to (7) and (8), the time-optimal path-constrained trajectory planning problem with inadmissible island constraints is defined as

$$\begin{aligned} &\min T \\ &\text{s.t.} f \in F, \end{aligned} \tag{9}$$

where $F$ is the set of feasible trajectories.

### B. EXAMPLE ON OWMR

It is noted that the proposed approach could apply to full-actuated robots, overactuated robots, or nonholonomic robots. This subsection takes an active-casters-based OWMR vehicle as an example to show the detailed procedure of obtaining the problem (9).

As shown in Fig. 2, the PC and DSP-FPGA board are used to compute the pose and trajectory, while the laser and encoders are responsible for the robot location and collecting motor speed. The geometric diagram of active-casters-based OWMR vehicle is shown in Fig. 3. $\mathscr{F}(X_rO_rY_r)$ and $\mathscr{F}(X_wO_wY_w)$ are the robot coordinate system and reference coordinate system, respectively. The pose of the vehicle is represented as $\boldsymbol{q} = [x\ y\ \theta]^{\mathrm{T}}$, with $x \in \mathbb{R}$ and $y \in \mathbb{R}$ being the position of $O_r$ in $\mathscr{F}(X_wO_wY_w)$ and $\theta \in \mathbb{S}$ being the vehicle orientation. The scalars $r, R, d$ are respectively the radius of

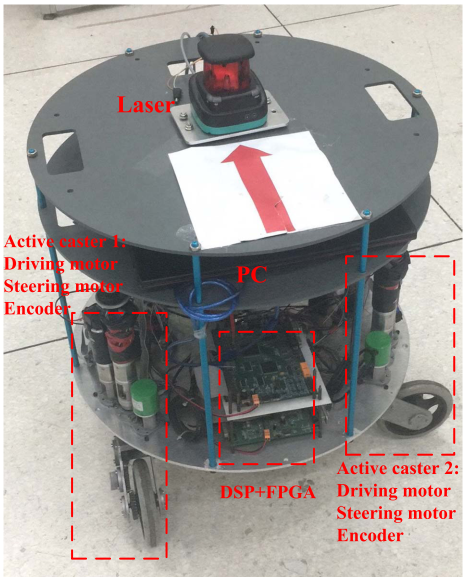


**FIGURE 2.** NK-OMNI I: An omnidirectional wheeled mobile robot with two active casters.

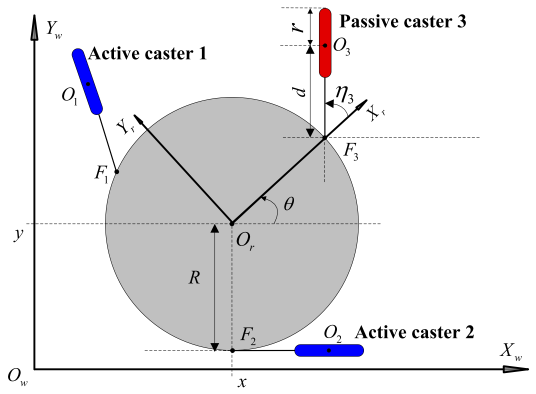


**FIGURE 3.** The OWMR using one red passive caster and two blue active casters.

casters, the radius of the vehicle frame and the length of the steering link $F_iO_i$, $i \in [1, 3]$, wherein $O_i$ is the center point of the casters. The angles of three casters $\eta_i \in \mathbb{S}$ are the angles from $O_rF_i$ to $F_iO_i$, $i \in [1, 3]$, with anticlockwise being positive. Each blue active caster is equipped with two motors to driving and steering velocities independently, while the red passive caster without motors is used to keep balance.

The dynamically extended model of the vehicle is described as

$$\boldsymbol{\omega} = J\dot{\boldsymbol{q}}, \tag{10}$$
$$\boldsymbol{a} = \dot{\boldsymbol{\omega}}, \tag{11}$$

where $\boldsymbol{\omega} \in \mathbb{R}^4$ represents the driving and steering velocity of active casters, and $\boldsymbol{a} \in \mathbb{R}^4$ represents the driving and steering acceleration of active casters. The matrix $J \in \mathbb{R}^{4\times3}$ is represented as

$$J = [\boldsymbol{J}_1 \ \boldsymbol{J}_2 \ \boldsymbol{J}_3 \ \boldsymbol{J}_4]^{\mathrm{T}},$$

$$\boldsymbol{J}_1 = \left[\frac{-\cos(\theta+\eta_1+2\pi/3)}{r} \ \ \frac{-\sin(\theta+\eta_1+2\pi/3)}{r} \ \ \frac{-R\sin(\eta_1)}{r}\right]^{\mathrm{T}},$$
$$\boldsymbol{J}_2 = \left[\frac{\sin(\theta+\eta_1+2\pi/3)}{d} \ \ \frac{-\cos(\theta+\eta_1+2\pi/3)}{d} \ \ \frac{-d-R\cos(\eta_1)}{d}\right]^{\mathrm{T}},$$
$$\boldsymbol{J}_3 = \left[\frac{-\cos(\theta+\eta_2-2\pi/3)}{r} \ \ \frac{-\sin(\theta+\eta_2-2\pi/3)}{r} \ \ \frac{-R\sin(\eta_2)}{r}\right]^{\mathrm{T}},$$
$$\boldsymbol{J}_4 = \left[\frac{\sin(\theta+\eta_2-2\pi/3)}{d} \ \ \frac{-\cos(\theta+\eta_2-2\pi/3)}{d} \ \ \frac{-d-R\cos(\eta_2)}{d}\right]^{\mathrm{T}}.$$

Note that the steering velocities in $\boldsymbol{\omega}$ are described as

$$\dot{\eta}_1 = \boldsymbol{J}_2^{\mathrm{T}}\dot{\boldsymbol{q}}, \tag{12}$$
$$\dot{\eta}_2 = \boldsymbol{J}_4^{\mathrm{T}}\dot{\boldsymbol{q}}. \tag{13}$$

Along specified paths by the first stage of the decoupled planning, the OWMR vehicle pose is represented as

$$\boldsymbol{q}(s) : s \in [0, s_e] \rightarrow \boldsymbol{q} \in \mathbb{R}^3. \tag{14}$$

The first and second time derivatives of $\boldsymbol{q}$ are described as

$$\dot{\boldsymbol{q}} = \boldsymbol{q}_s\dot{s}, \tag{15}$$
$$\ddot{\boldsymbol{q}} = \boldsymbol{q}_{ss}\dot{s}^2 + \boldsymbol{q}_s\ddot{s}, \tag{16}$$

where $\boldsymbol{q}_s = \partial\boldsymbol{q}/\partial s$, $\boldsymbol{q}_{ss} = \partial\boldsymbol{q}_s/\partial s$. In addition, substituting (15) into (12) and (13) yields that

$$d\eta_1/ds = \boldsymbol{J}_2^{\mathrm{T}}\boldsymbol{q}_s, \tag{17}$$
$$d\eta_2/ds = \boldsymbol{J}_4^{\mathrm{T}}\boldsymbol{q}_s. \tag{18}$$

Then, along the specified path, the angles $\eta_1(s)$ and $\eta_2(s)$ are obtained by integrating numerically (17) and (18) from $s = 0$ to $s = s_e$.

Physical constraints on velocity and acceleration of active casters are given as

$$-\boldsymbol{\omega}_{max} \le \boldsymbol{\omega} \le \boldsymbol{\omega}_{max}, \tag{19}$$
$$-\boldsymbol{a}_{max} \le \boldsymbol{a} \le \boldsymbol{a}_{max}, \tag{20}$$

where $\boldsymbol{\omega}_{max}, \boldsymbol{a}_{max} \in \mathbb{R}^4$ are constant vectors.

In order to guarantee velocity constraints (19), substituting (10) and (15) into (19) yields the inequality (1), wherein the vectors $\boldsymbol{A}(s)$, $\boldsymbol{D}(s)$ are described as

$$\boldsymbol{A}(s) = [(J\boldsymbol{q}_s)^{\mathrm{T}} - (J\boldsymbol{q}_s)^{\mathrm{T}}]^{\mathrm{T}}, \tag{21}$$
$$\boldsymbol{D}(s) = [-\boldsymbol{\omega}_{max}^{\mathrm{T}} - \boldsymbol{\omega}_{max}^{\mathrm{T}}]^{\mathrm{T}}. \tag{22}$$

In order to guarantee acceleration constraints (20), substituting (11), (15) and (16) into (20) yields the inequality (2), wherein the vectors $\boldsymbol{B}(s)$, $\boldsymbol{C}(s)$ are given as

$$\boldsymbol{B}(s) = [(J\boldsymbol{q}_{ss} + J_s\boldsymbol{q}_s)^{\mathrm{T}} - (J\boldsymbol{q}_{ss} + J_s\boldsymbol{q}_s)^{\mathrm{T}}]^{\mathrm{T}}, \tag{23}$$
$$\boldsymbol{C}(s) = [-\boldsymbol{a}_{max}^{\mathrm{T}} - \boldsymbol{a}_{max}^{\mathrm{T}}]^{\mathrm{T}}. \tag{24}$$

Note that the matrix $J_s \in \mathbb{R}^{4\times 3}$ equals to $\partial J/\partial s$. Finally, according to (1) and (2), the trajectory planning problem (9) of OWMRs is obtained as Section II-A.

## III. SOLUTION

For this TOPCTP problem (9), with the aid of heuristic tree search techniques, an any-time time-optimal trajectory planning algorithm is presented to achieve incremental optimization of feasible trajectories during one planning period. Trap regions considering inadmissible islands are detected to shorten the search time. The schematic diagram of the solution is shown in Fig. 4.

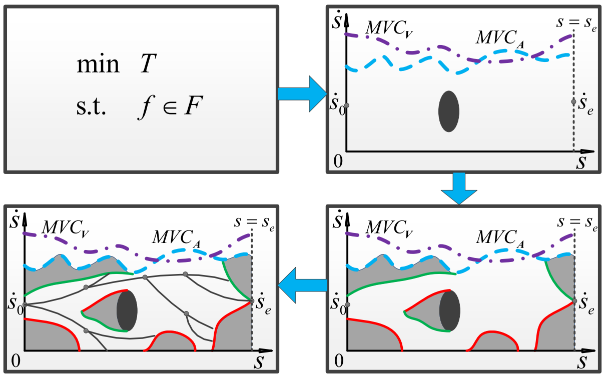


**FIGURE 4. The top-left diagram describes the mathematical expression of the TOPCTP problem with inadmissible island constraints. It is redescribed as the top-right diagram: Cyan dash and purple dash-dot curves are resulted from acceleration and velocity constraints respectively, while the black ellipse stands for inadmissible island constraints. The proposed algorithm first detects all trap regions (gray regions in the bottom-right diagram). With the aid of trap regions, the proposed algorithm costs little time to construct a heuristic search tree between the starting and terminal points as show in the bottom-left diagram. Feasible trajectories obtained from the search tree are optimized incrementally during one planning period. Time-optimal trajectories can be obtained when the period is long enough for searching.**

### A. HEURISTIC TREE SEARCH

The core of the tree search is to build a tree from the starting point ($s = 0, \dot{s} = \dot{s}_0$) to the terminal point ($s = s_e, \dot{s} = \dot{s}_e$) on the plane $(s, \dot{s})$. The tree nodes are the points on the plane $(s, \dot{s})$ as shown in Fig. 1. The tree edges are the curves which are obtained by doing forward direction integral with path acceleration (3) from the tree nodes. The feasible trajectory consists of tree edges which satisfy constraints (3) and avoid inadmissible islands.

The node degree $\delta$ represents the number of subtrees of a tree node [48]. Thus, starting from one tree node, $\delta$ tree edges are integrated forward with the following acceleration

$$\ddot{s}_i = \alpha(s,\dot{s}) + \frac{(i-1)(\beta(s,\dot{s}) - \alpha(s,\dot{s}))}{\delta - 1}, \quad i \in [1,\delta], \tag{25}$$

wherein $\delta$ is greater than or equal to 2. The integral length of tree edges is described as $\rho \in (0, s_e)$. As shown in Fig. 1, the tree search using $\delta = 2$ and $\rho = s_e/4$ generates one feasible trajectory consisting of decelerating trajectories ($\alpha_1, \alpha_2$) and accelerating trajectories ($\beta_3, \beta_4$).

In the iteration of the tree search, starting from the point $(0, \dot{s}_0)$, the tree gradually extends to the point $(s_e, \dot{s}_e)$ and one feasible trajectory is generated in the admissible region. In this subsection, the heuristic function is incorporated into the tree-search procedure to estimate each leaf node.

According to the problem (9), the heuristic function is defined as

$$H(s,\dot{s}) = \dot{s} + \frac{\varepsilon}{1 + \sqrt{(s - s_e)^2 + (\dot{s} - \dot{s}_e)^2}}, \tag{26}$$

wherein the parameter $\varepsilon$ is the weight of the second term in (26), and the range of $\varepsilon$ is from zero to a user-specified positive real number. The tree search selects tree nodes in descending order of $H(s, \dot{s})$. The leaf node with the maximum heuristic value has the highest priority to extend forward the search tree.

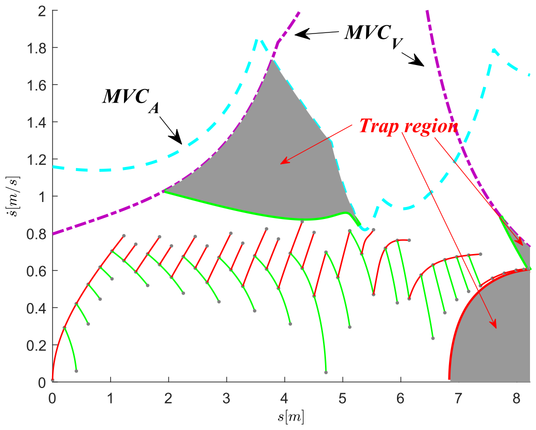


**FIGURE 5. The tree search with $\varepsilon = 1$ generates one feasible trajectory. The green solid curves represent decelerating trajectories with $\alpha(s, \dot{s})$, and the red solid curves represent accelerating trajectories with $\beta(s, \dot{s})$. The gray regions are the trap regions. There are totally 71 gray tree nodes in tree search.**

#### 1) PHYSICAL INTERPRETATION OF $\varepsilon$

- $\varepsilon > 0$: The expansion prefers the tree node which is closer to the terminal point $(s_e, \dot{s}_e)$. Therefore, the explored region area decreases as shown in Fig. 5, and the computation time is reduced to guarantee that one feasible trajectory is obtained in real time.
- $\varepsilon = 0$: The expansion prefers the tree node with the maximum path velocity. Thus, the explored region area increases as shown in Fig. 6, and the generated trajectory is time-optimal.

From the above analysis, the parameter $\varepsilon$ possesses an important property which affects the traveling time and computation time of the feasible trajectory. With the aid of this property, the optimal solution of (9) can be obtained before the end of one planning period. First, the search tree with an initial $\varepsilon > 0$ extends to generate a feasible trajectory rapidly. Accordingly, if the planning period ends, then this feasible trajectory is returned, otherwise the search tree continues the extension with a smaller $\varepsilon$ to generate a feasible solution as optimal as possible.

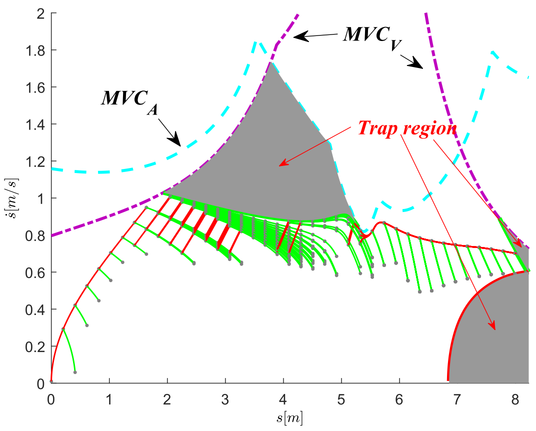


**FIGURE 6.** The tree search with $\varepsilon = 0$ generates the time-optimal trajectory. The green solid curves represent decelerating trajectories with $\alpha(s, \dot{s})$, and the red solid curves represent accelerating trajectories with $\beta(s, \dot{s})$. The gray regions are the trap regions. There are totally 742 gray tree nodes in tree search.

*Remark 1:* When the parameter $\varepsilon$ is set as a large positive value, the nearest tree node to the terminal point $(s_e, \dot{s}_e)$ is chosen to extend the search tree along the given path, and it indicates that the expansion of the search tree always assigns appropriate path velocity to untraversed path coordinate of the given path. Thus, the proposed algorithm can cost very little computation time to obtain a feasible trajectory along the given path. □

## B. TRAP REGION CONSIDERING INADMISSIBLE ISLANDS

In this subsection, in order to increase efficiency of the tree search, trap regions considering inadmissible islands are defined and detected.

*Definition 1:* A trap region is contained in the admissible region. Starting from the points in the trap region, the trajectory cannot reach the terminal point $(s_e, \dot{s}_e)$ while satisfying constraints (3) and avoiding inadmissible islands.

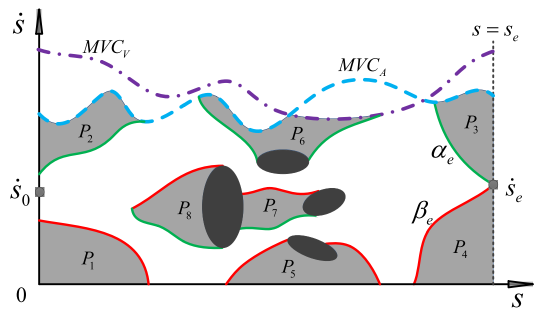


**FIGURE 7.** The gray regions $P_1 - P_8$ are trap regions considering inadmissible islands. The black ellipses are inadmissible islands. The red curves are accelerating trajectories with $\beta(s, \dot{s})$ and green curves are decelerating trajectories with $\alpha(s, \dot{s})$. Note that the accelerating $\beta_e$ and decelerating $\alpha_e$ are not contained in $P_3, P_4$.

In Fig. 7, the gray regions represent trap regions. According to *Definition 1*, the corresponding detecting method of trap regions is given as **Algorithm 1**. The input argument $Q$ represents the set of inadmissible islands, such as four black ellipses in Fig. 7. The output argument $P$ represents the set of trap regions considering inadmissible islands, such as eight gray regions in Fig. 7. The called subfunctions are described as follows:

- *RightMostPoints*$(Q)$ : This subfunction returns the set of rightmost points of each inadmissible island boundary.
- *DecTraj*$(g, Q, Dec)$ : Starting from the point $g$, one decelerating trajectory is computed by doing backward direction integral with $\alpha(s, \dot{s})$ until the boundary of the admissible region or islands in $Q$ is hit. This decelerating trajectory is added into $Dec$, and the hitting point is returned.
- *AccTraj*$(g, Q, Acc)$ : Starting from the point $g$, one accelerating trajectory is computed by doing backward direction integral with $\beta(s, \dot{s})$ until the boundary of the admissible region or islands in $Q$ is hit. This accelerating trajectory is added into $Acc$, and the hitting point is returned.
- *SearchDecPoint*$(h)$ : If the point $h$ is in $MVC$ or the line $\dot{s} = 0$, then starting from $h$, this subfunction searches backward along $MVC$ or the line $\dot{s} = 0$ the first point $\Omega$. If the point $h$ is in the boundary of $Q$, then starting from $h$, this subfunction searches clockwise along the boundary the first point $\Omega$. At the point $\Omega$, the decelerating trajectory with $\alpha(s, \dot{s})$ backward dives into the admissible region immediately. Finally, $\Omega$ is returned.
- *SearchAccPoint*$(h)$ : If the point $h$ is in $MVC$ or the line $\dot{s} = 0$, then starting from $h$, this subfunction searches backward along $MVC$ or the line $\dot{s} = 0$ the first point $\Omega$. If $h$ is the boundary of $Q$, then starting from $h$, this subfunction searches anticlockwise along the boundary the first point $\Omega$. At the point $\Omega$, the accelerating trajectory with $\beta(s, \dot{s})$ backward dives into the admissible region immediately. Finally, $\Omega$ is returned.
- *CombineCurves*$(Q, Dec, Acc)$ : This subfunction combines decelerating trajectories $Dec$, accelerating trajectories $Acc$, the boundaries of $Q$ and the admissible region to enclose and return different trap regions.

*Remark 2:* In order to understand the proposed TRDA more clearly, this remark offers an analogy as an explanation. The admissible region is likened to the ground, and the inadmissible islands are likened to high mountains, then the trap regions are likened to the shadows of these mountains. The proposed TRDA first searches the corners of the shadows along high mountains. Starting from these corners, TRDA computes the boundaries of the shadows with numerical integration. Finally, these boundaries are combined into the shadows, namely the trap regions. □

With the aid of **Algorithm 1**, all trap regions considering inadmissible islands are detected, and the tree search in Section III-A avoids these trap regions to decrease computation time and memory of solving the problem (9).

## C. TRAJECTORY PLANNING

In this subsection, an any-time time-optimal path-constrained trajectory planning algorithm is given in **Algorithm 2**.

**Algorithm 1** Trap Region Detecting Algorithm (TRDA)

```
Input: $s_e$, $\dot{s}_e$, $MVC$, $Q$
Output: $P$
1: $P \leftarrow$ NULL
2: $G \leftarrow RightMostPoints(Q)$
3: Add $(s_e, \dot{s}_e)$ into $G$
4: $Dec \leftarrow$ NULL
5: for all $g$ in $G$ do
6:     $h \leftarrow DecTraj(g, Q, Dec)$
7:     $q \leftarrow SearchDecPoint(h)$ and Add $q$ into $G$
8: end for
9: $Acc \leftarrow$ NULL
10: for all $g$ in $G$ do
11:     $h \leftarrow AccTraj(g, Q, Acc)$
12:     $q \leftarrow SearchAccPoint(h)$ and Add $q$ into $G$
13: end for
14: $P \leftarrow CombineCurves(Q, Dec, Acc)$
15: return $P$
```

**Algorithm 2** An Any-Time Time-Optimal Path-Constrained Trajectory Planning Algorithm

```
Input: $s_e$, $\dot{s}_0$, $\dot{s}_e$, $MVC$, $\Pi$, $Q$
Output: $f$
1: $P \leftarrow TRDA(s_e, \dot{s}_e, MVC, Q)$
2: if $IsObstructed(\dot{s}_0, \dot{s}_e, P, Q)$ then
3:     return Failure
4: end if
5: $\Upsilon, L, S, G, \delta, \rho, \epsilon \leftarrow InitTreeSearch(s_e, \dot{s}_0, \dot{s}_e)$
6: while $GetRunTime() < \Pi$ do
7:     repeat
8:         $H \leftarrow HeuristicFun(L)$
9:         $L^* \leftarrow OutMaxLeaf(L, H)$
10:        $V, E \leftarrow ComputeEdgeNode(L^*, \delta, \rho)$
11:        $UpdateTree(\Upsilon, L, V, E, P)$
12:    until $IsSameNode(V, G)$ or $CrossLastSolution(E, f)$
13:    $f \leftarrow ObtainTrajectory(\Upsilon, S, G)$
14:    $ReduceEpsilon(\epsilon)$
15: end while
16: return $f$
```

The input arguments $\Pi$ and $Q$ are a planning period and inadmissible islands, respectively. First, the 1st and 2nd lines achieve the initialization of the tree search in Section III-A and computation of trap regions in Section III-B, respectively. Then, toward the target $(s_e, \dot{s}_e)$, the 3rd to 12th lines extend the search tree using $(0, \dot{s}_0)$ as a root node until the end of $\Pi$. In this extension, feasible trajectories are obtained and optimized by reducing the parameter $\epsilon$ in (26). The corresponding subfunctions are described as follows:

- *IsObstructed*($\dot{s}_0$, $\dot{s}_e$, $P$, $Q$): If trap regions $P$ and inadmissible islands $Q$ connect to form a barrier between $\dot{s}_0$ and $\dot{s}_e$, then *TRUE* is returned, or *FALSE*.
- *GetRunTime()*: This subfunction returns the elapsed time of Algorithm 2. The while-loop structure terminates when the returned value is greater than or equal to a planning period $\Pi$.
- *InitTreeSearch*$(s_e, \dot{s}_0, \dot{s}_e)$: This subfunction initializes one tree search from the starting node $S = (0, \dot{s}_0)$ to the terminal node $G = (s_e, \dot{s}_e)$. Except for $S, G$, it also initializes and returns the parameters $\delta$, $\rho$ in (25), $\epsilon$ in (26), and one new search tree $\Upsilon$ with $S$ as the root node, and its leaf node set $L = \{S\}$.
- *HeuristicFun*$(L)$: With the aid of (26), this subfunction computes and returns the heuristic value of each leaf node in the set $L$.
- *OutMaxLeaf*$(L, H)$: The leaf node with the maximum heuristic value is returned and removed from the leaf nodes $L$.
- *ComputeEdgeNode*$(L^*, \delta, \rho)$: Starting from the leaf node $L^*$, new tree edges are computed by doing forward direction integral with $\ddot{s}_i$, $i \in [1, \delta]$ in (25), respectively. The integral length equals to $\rho$. These new generated tree edges and nodes are returned.
- *UpdateTree*$(\Upsilon, L, V, E, P)$: Tree edges $E$ and nodes $V$ which are in the admissible region but not in trap regions $P$, are added into the search tree $\Upsilon$ and leaf nodes $L$.
- *IsSameNode*$(V, G)$: This subfunction decides whether one leaf node in $V$ is same with the terminal node $G = (s_e, \dot{s}_e)$. If the Euclidean distance between two nodes is close on the plane $(s, \dot{s})$, then *TRUE* is returned, or *FALSE*.
- *CrossLastSolution*$(E, f)$: This subfunction decides whether one tree edge in $E$ intersects the last solution $f$. If they intersect, then *TRUE* is returned, or *FALSE*.
- *ObtainTrajectory* $(\Upsilon, S, G)$: This subfunction traverses tree edges in $\Upsilon$ from the terminal node $G$ to the starting node $S$. Those traversed edges constitute one optimal trajectory in $\Upsilon$, and it is returned.
- *ReduceEpsilon*$(\epsilon)$: This subfunction reduces the parameter $\epsilon$ in (26) until zero.

With the aid of the heuristic tree search and trap regions, this algorithm iteratively generates feasible trajectories as optimal as possible during one planning period $\Pi$. When the heuristic parameter $\epsilon$ is reduced to zero before the end of the planning period, a time-optimal trajectory is obtained for (9).

*Remark 3:* Note that the proposed algorithm constructs tree edges and tree nodes to explore the admissible region under the maximum velocity curve in (6). When the tree connects the starting and terminal velocities, one feasible trajectory is obtained. If the parameter $\varepsilon$ in (26) is set as a large value, the generation of the feasible trajectory costs very little computation time as described in Remark 1. Then, the proposed algorithm reduces $\varepsilon$ and constructs more tree edges and tree nodes to explore the admissible region near the maximum velocity curve. It indicates that the proposed algorithm needs more computation time. When new tree edges intersect the previously generated trajectory, the proposed algorithm generates another faster trajectory. If the parameter $\varepsilon$ is reduced to zero, the proposed algorithm explores the highest path velocity in the admissible region to output

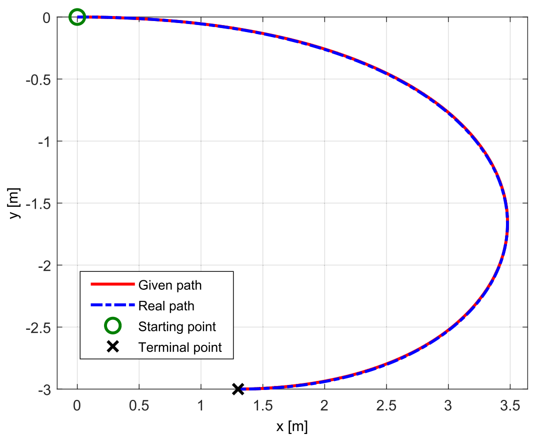


**FIGURE 8.** Experimental paths: The red solid curve is the given path, and the blue dash-dot curve is the real path obtained from the laser equipment.

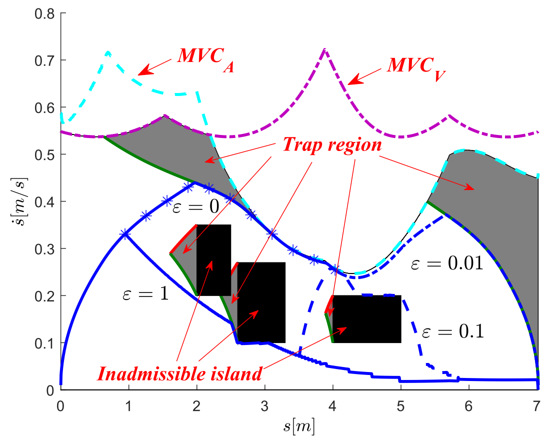


**FIGURE 9.** The proposed algorithm detects trap regions considering inadmissible islands, and iteratively generates feasible trajectories, which tend to be time-optimal with the smaller parameter $\varepsilon$ in (26).

time-optimal trajectories. As the parameter $\varepsilon$ gradually decreases to zero, the number of nodes and edges will increase larger and larger, and thus the proposed algorithm needs a relatively longer planning period to obtain time-optimal trajectories by incremental tree searching.

## IV. EXPERIMENTAL RESULTS

In order to verify the proposed algorithm, this section provides two experimental cases on OWMRs. The first case is used to show the any-time time-optimal planning performance, that is, along a specified path, the proposed algorithm iteratively generates feasible trajectories as optimal as possible during one planning period, even a time-optimal solution. The second case is used to show that the proposed algorithm may be applied in transportation systems [50], [51] or warehouse logistics for collision avoidance at crossroad.

### A. FIRST CASE

In this subsection, the any-time performance and time optimality of the proposed algorithm are demonstrated on an OWMR as shown in Fig. 2. Along the given cubic Bézier path in Fig. 8, the position of the OWMR is described as:

$$x = (1-\lambda)^3 x_0 + 3(1-\lambda)^2\lambda x_1 + 3(\lambda^2 - \lambda^3)x_2 + \lambda^3 x_3,$$
$$y = (1-\lambda)^3 y_0 + 3(1-\lambda)^2\lambda y_1 + 3(\lambda^2 - \lambda^3)y_2 + \lambda^3 y_3,$$

wherein the scalar $\lambda \in [0, 1]$ is the path parameter, and the points $(x_i, y_i)$ are path control points. The scalars $\lambda$ and $s$ obey a nonlinear scaling relation. The OWMR orientation is represented as $\theta(s) = \pi s/s_e, s \in [0, s_e]$. Each component of velocity constraints $\boldsymbol{\omega}_{max}$ in (19) is set as 8.8 $[rad/s]$, and each component of acceleration constraints $\boldsymbol{a}_{max}$ in (20) is set as 4 $[rad/s^2]$. The starting and terminal path velocities are set as $\dot{s}_0 = \dot{s}_e = 0$.

In this case, a planning period is set as 0.050 [s], which indicates that the proposed algorithm must terminate at the end of the planning period. In addition, there exist three rectangle inadmissible islands on the plane $(s, \dot{s})$, which are specified by operating personnel as shown in Fig. 9. These inadmissible islands require that the OWMR moves with high/low speed along the corresponding parts of the given path. Starting from the begin of the planning period, the proposed Algorithm 2 first calls Algorithm 1 to detect trap regions considering inadmissible islands in 0.001 [s]. Then, the proposed Algorithm 2 iteratively generates four feasible trajectories in (7). These trajectories tend to be time-optimal with the parameter $\epsilon$ reducing to zero. The first feasible trajectory (the blue solid curve) with $\epsilon = 1$ is found out in 0.011 [s]. Then, the proposed algorithm reduces the parameter $\epsilon$ to 0.1, and it costs 0.009 [s] to obtain the second feasible trajectory by updating the first feasible trajectory with the blue dash curve. Sequentially, the parameter $\epsilon$ is reduced into 0.01, and the third feasible trajectory is obtained by updating the second feasible trajectory with the blue dash-dot curve in 0.013 [s]. Finally, the parameter $\epsilon$ is reduced into zero, and a time-optimal (the fourth) trajectory is obtained by updating the third feasible trajectory with the blue solid curve with the star marker in 0.012 [s]. In the above iteration, the proposed algorithm totally costs 0.046 [s] to generate a time-optimal trajectory for the problem (9), therefore the any-time performance of the proposed algorithm is demonstrated.

With the aid of a tracking trajectory controller with velocity saturation constraints [49], the OWMR vehicle tracks this time-optimal trajectory. The tracking errors for $x, y, \theta$ go to zero as shown in Fig. 10. The velocity and acceleration of active casters are shown in Fig. 11 and Fig. 12. It indicates that both velocity and acceleration constraints are satisfied. Moreover, there always exists one of active casters whose velocity touches the velocity boundary or acceleration boundary. Thus, Fig. 11 and Fig. 12 demonstrate the time optimality of the obtained trajectory with $\epsilon = 0$ from the proposed algorithm.

### B. SECOND CASE

This subsection aims to verify that the proposed algorithm is capable of handling inadmissible islands, which should be

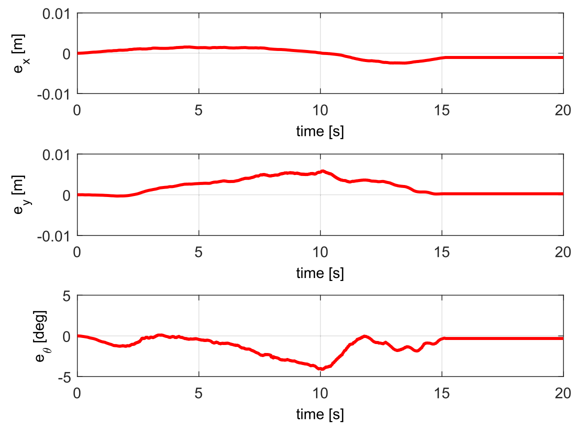


**FIGURE 10.** The tracking errors for $x, y, \theta$.

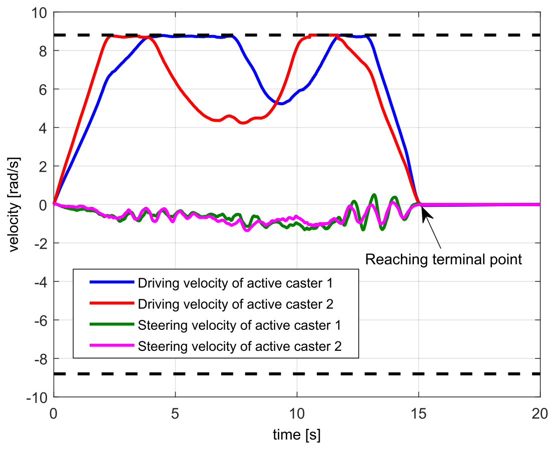


**FIGURE 11.** The velocities of active casters: The dark dash lines represent the velocity constraints $\omega_{max}$ and other solid lines represent driving and steering velocities of active casters.

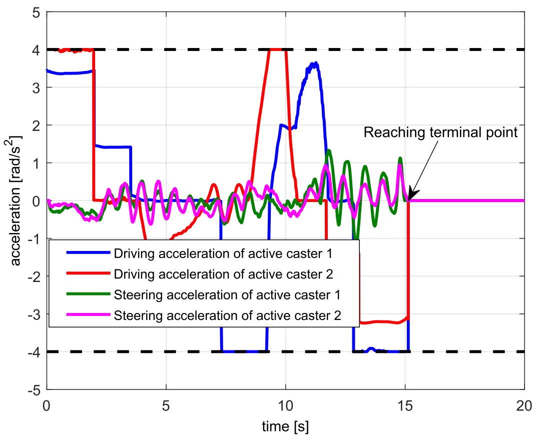


**FIGURE 12.** The accelerations of active casters: The dark dash lines represent the acceleration constraints $a_{max}$ and other solid lines represent driving and steering accelerations of active casters.

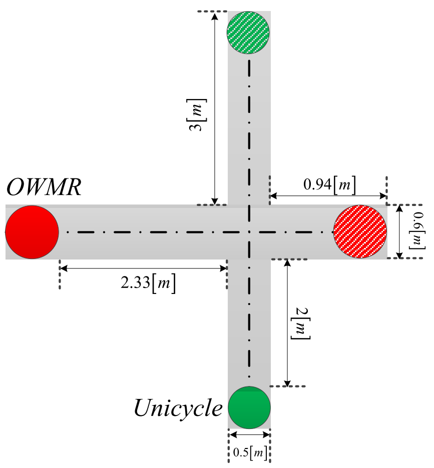


**FIGURE 13.** OWMR and unicycle vehicles move along straight roads to pass through the same crossroad. The solid circles represent the starting position of OWMR and unicycle vehicles. The shadow circles represent the terminal position of OWMR and unicycle vehicles.

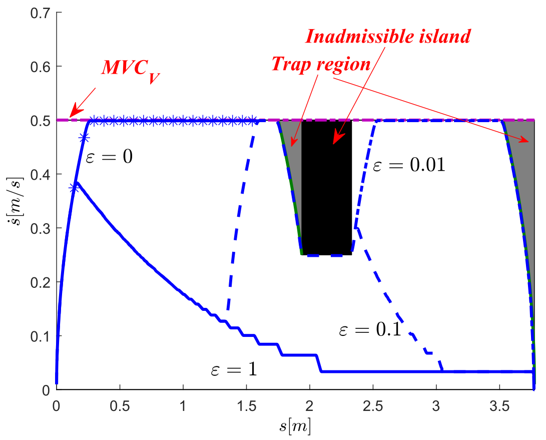


**FIGURE 14.** The feasible trajectories obtained iteratively in the proposed algorithm tend to be time-optimal with the smaller parameter $\varepsilon$ in (26).

considered in collision avoidance at crossroads. As shown in Fig. 13, OWMR and unicycle vehicles pass through the same crossroad along different straight roads. The unicycle vehicle moves at a constant speed 0.5 $[m/s]$. For the OWMR vehicle, each component of velocity constraints $\boldsymbol{\omega}_{max}$ in (19) is set as 6.84 $[rad/s]$, and each component of acceleration constraints $\boldsymbol{a}_{max}$ in (20) is set as 6.84 $[rad/s^2]$.

In order to avoid collision with the unicycle at the crossroad, one inadmissible island (the black rectangle in Fig. 14) is set at the road of OWMR. This inadmissible island is similar to the deceleration area before the crossroad, which requires that the OWMR vehicle decelerates to make the unicycle vehicle preferentially pass through the crossroad. First, Algorithm 1 costs 0.001 [s] to detect trap regions considering this inadmissible island as shown in Fig. 14. Then, the proposed Algorithm 2 computes iteratively four feasible trajectories in (7). Moreover, those trajectories tend to be time-optimal with the smaller $\varepsilon$ in (26). With $\epsilon = 1$, the first feasible trajectory (blue solid curve) is obtained in 0.009 [s]. The parameter $\epsilon$ is reduced into 0.1, and then the second feasible trajectory is obtained by updating the first trajectory with the blue dash curve in 0.012 [s]. Then, the third feasible trajectory with $\epsilon = 0.01$ is obtained by

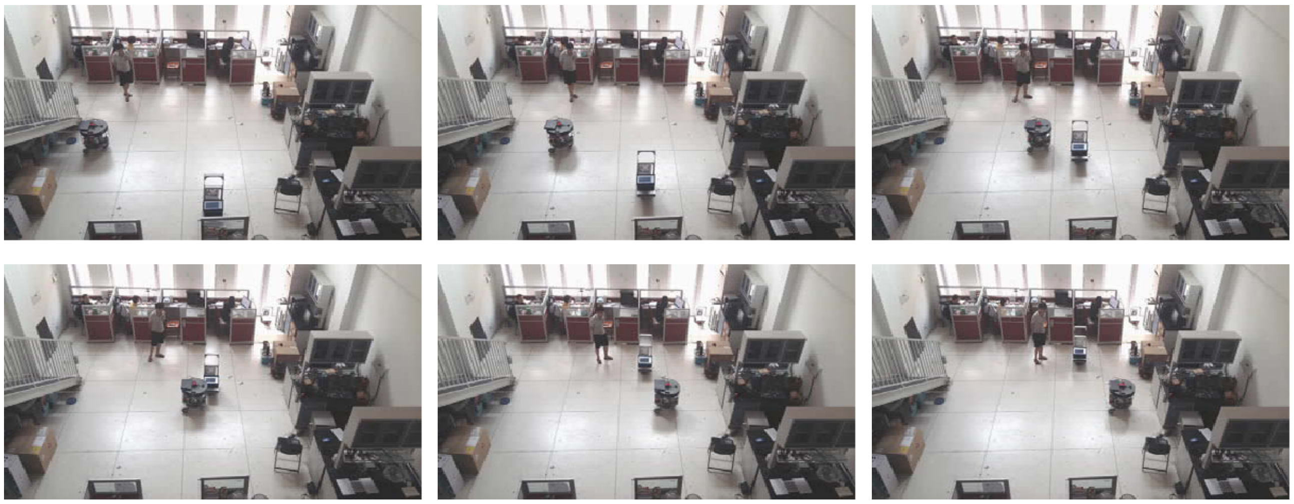


**FIGURE 15.** Snapshots of the OWMR following the time-optimal trajectory obtained by the proposed algorithm.

updating the second trajectory with the blue dash-dot curve in 0.011 [s]. Finally, a time-optimal trajectory with $\epsilon = 0$ is obtained by updating the third trajectory with the blue solid curve with the star marker in 0.012 [s]. In the above iteration, the proposed algorithm costs 0.045 [s] to compute the time-optimal trajectory with $\epsilon = 0$. The motion planning period of OWMR is also set as 0.050 [$s$], thus the proposed algorithm outputs the time-optimal trajectory before the end of the planning period. The snapshot in Fig. 15 shows that the OWMR vehicle successfully avoids collision at the crossroad with the unicycle vehicle by following the time-optimal trajectory.

## V. DISCUSSION

The main contributions and advantages of the proposed algorithm are discussed as follows:

- The introduction of heuristic tree search techniques first achieves the online processing of inadmissible island constraints. The heuristic technique constructs the search tree between starting and terminal velocities to explore the admissible region and avoid inadmissible islands. As described in Remark 1, when the parameter $\varepsilon$ in (26) is set as a large positive number, the proposed algorithm generates the first feasible trajectory quickly. The experimental results also show that the proposed algorithm only costs about 0.01s to generate the first feasible trajectory.
- The detection of trap regions greatly reduces the tree search time. The definition indicates that the trajectories falling into trap regions cannot reach the terminal velocity. Thus, it is unavailing for the tree search to explore trap regions. With searching these trap regions, the computation efficiency of tree search is largely increased.
- The incremental computation of feasible trajectories brings any-time performance which indicates that those iteratively generated trajectories tend to be closer and closer to the time-optimal one before the end of a planning period. The experimental results verify the any-time performance of the proposed algorithm. In a planning period of 0.05s, the proposed algorithm iteratively generates four feasible trajectories. In the iteration, the tree search continually explores the admissible region near the maximum velocity curve. When new tree edges intersect previously generated feasible trajectories, one faster feasible trajectory is obtained as an output candidate. If the planning task is badly need of one solution, the proposed algorithm returns the output candidate immediately to ensure that robotic systems work reliably.

## VI. CONCLUSIONS

In this paper, a novel tree-search-based any-time time-optimal path-constrained trajectory planning algorithm has been proposed in the presence of inadmissible island constraints. The online construction of heuristic search trees brings feasible trajectories satisfying inadmissible island constraints. Moreover, the detection of trap regions greatly reduces the size of the search tree. With the aid of these two techniques, the proposed algorithm costs very little computation time to return one initial feasible trajectory along the given path. Then, the proposed algorithm continues doing tree search to achieve incremental optimization of traveling time of feasible trajectories before the end of a planning period. The incremental optimization brings any-time performance which indicates that those iteratively generated trajectories tend to be time-optimal during one planning period. Experimental results on OWMRs have demonstrated that under inadmissible island constraints, the proposed algorithm could generate an initial feasible trajectory for about 0.01s, and it could even generate the time-optimal trajectories in one planning period of 0.05s.

In the future, we will analyze and prove the completeness property of the proposed algorithm. For complete planning algorithms, feasible trajectories are returned if a planning problem is solvable, otherwise a failure is returned in finite time. In addition, the proposed algorithm may be applied to multiple robot systems to avoid collision as shown in Section IV-B.

## REFERENCES

[1] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Robot. Res.*, vol. 5, no. 1, pp. 90–98, 1986.

[2] M. Yuan, Z. Chen, B. Yao, and X. Zhu, "Time optimal contouring control of industrial biaxial gantry: A high-efficient analytical solution of trajectory planning," *IEEE/ASME Trans. Mechatronics*, vol. 22, no. 1, pp. 247–257, Feb. 2017.

[3] J. Kim and J. P. Ostrowski, "Motion planning a aerial robot using rapidly-exploring random trees with dynamic constraints," in *Proc. IEEE Int. Conf. Robot. Autom.*, Sep. 2003, pp. 2200–2205.

[4] J. Liao, Z. Chen, and B. Yao, "Model-based coordinated control of four-wheel independently driven skid steer mobile robot with wheel/ground interaction and wheel dynamics," *IEEE Trans. Ind. Informat.*, vol. 14, no. 9, pp. 1–10, Sep. 2018, doi: 10.1109/TII.2018.2869573.

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

[6] L. Chen, Y. Shan, W. Tian, B. Li, and D. Cao, "A fast and efficient double-tree RRT*-like sampling-based planner applying on mobile robotic vehicles," *IEEE/ASME Trans. Mechatronics*, vol. 23, no. 6, pp. 2568–2578, Dec. 2018, doi: 10.1109/TMECH.2018.2821767.

[7] R. Holmberg and O. Khatib, "Development and control of a holonomic mobile robot for mobile manipulation tasks," *Int. J. Robot. Res.*, vol. 19, no. 11, pp. 1066–1074, 2000.

[8] M. Wada, Y. Inoue, and T. Hirama, "A new active-caster drive system with a dual-ball transmission for omnidirectional mobile robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 2525–2532.

[9] P. Shen, Y. Fang, and X. Zhang, "Trajectory planning of omnidirectional mobile robots with active casters: Multi-motor coordination and singularity avoidance," in *Proc. IEEE Int. Conf. Cyber Technol. Automat., Control Intell. Syst.*, Jun. 2015, pp. 151–156.

[10] J. J. M. Lunenburg, S. A. M. Coenen, G. J. L. Naus, M. J. G. van de Molengraft, and M. Steinbuch, "Motion planning for mobile robots: A method for the selection of a combination of motion-planning algorithms," *IEEE Robot. Autom. Mag.*, vol. 23, no. 4, pp. 107–117, Dec. 2016.

[11] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *Int. J. Robot. Res.*, vol. 17, no. 7, pp. 760–772, 1998.

[12] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige, "The office marathon: Robust navigation in an indoor office environment," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2010, pp. 300–307.

[13] B. Donald, P. Xavier, J. Canny, and J. Reif, "Kinodynamic motion planning," *J. ACM*, vol. 40, no. 5, pp. 1048–1066, 1993.

[14] T. Fraichard, "Trajectory planning in a dynamic workspace: A 'state-time space' approach," *Adv. Robot.*, vol. 13, no. 1, pp. 75–94, 1998.

[15] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 4569–4574.

[16] J. Dong, P. M. Ferreira, and J. A. Stori, "Feed-rate optimization with jerk constraints for generating minimum-time trajectories," *Int. J. Mach. Tools Manuf.*, vol. 47, nos. 12–13, pp. 1941–1955, 2007.

[17] W. Van Loock, G. Pipeleers, M. Diehl, J. De Schutter, and J. Swevers, "Optimal path following for differentially flat robotic systems through a geometric problem formulation," *IEEE Trans. Robot.*, vol. 30, no. 4, pp. 980–985, Aug. 2014.

[18] S. Spedicato and G. Notarstefano, "Minimum-time trajectory generation for quadrotors in constrained environments," *IEEE Trans. Control Syst. Technol.*, vol. 26, no. 4, pp. 1335–1344, Jul. 2018.

[19] 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. Automat.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[20] M. Likhachev and D. Ferguson, "Planning long dynamically feasible maneuvers for autonomous vehicles," *Int. J. Robot. Res.*, vol. 28, no. 8, pp. 933–945, 2009.

[21] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *IEEE Access*, vol. 2, pp. 56–77, 2014.

[22] C. G. Lo Bianco, "Minimum-jerk velocity planning for mobile robot applications," *IEEE Trans. Robot.*, vol. 29, no. 5, pp. 1317–1326, Oct. 2013.

[23] Y. Zhao and P. Tsiotras, "Speed profile optimization for optimal path tracking," in *Proc. Amer. Control Conf.*, Jun. 2013, pp. 1171–1176.

[24] F. Bourbonnais, P. Bigras, and I. A. Bonev, "Minimum-time trajectory planning and control of a pick-and-place five-bar parallel robot," *IEEE/ASME Trans. Mechatronics*, vol. 20, no. 2, pp. 740–749, Apr. 2015.

[25] Y. Zhang *et al.*, "Hybrid trajectory planning for autonomous driving in highly constrained environments," *IEEE Access*, vol. 6, pp. 32800–32819, 2018.

[26] P. Shen, X. Zhang, and Y. Fang, "Complete and time-optimal path-constrained trajectory planning with torque and velocity constraints: Theory and applications," *IEEE/ASME Trans. Mechatronics*, vol. 23, no. 2, pp. 735–746, Apr. 2018.

[27] X. Zhang, Y. Fang, and N. Sun, "Minimum-time trajectory planning for underactuated overhead crane systems with state and control constraints," *IEEE Trans. Ind. Electron.*, vol. 61, no. 12, pp. 6915–6925, Dec. 2014.

[28] X. Li, Z. Sun, D. Cao, Z. He, and Q. Zhu, "Real-time trajectory planning for autonomous urban driving: Framework, algorithms, and verifications," *IEEE/ASME Trans. Mechatronics*, vol. 21, no. 2, pp. 740–753, Apr. 2016.

[29] O. Dahl and L. Nielsen, "Stability analysis of an online algorithm for torque limited path following," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 1990, pp. 1216–1222.

[30] T. Kroger, A. Tomiczek, and F. M. Wahl, "Towards on-line trajectory computation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2006, pp. 736–741.

[31] M. Giftthaler, F. Farshidian, T. Sandy, L. Stadelmann, and J. Buchli, "Efficient kinematic planning for mobile manipulators with non-holonomic constraints using optimal control," in *Proc. IEEE Int. Conf. Robot. Autom.*, May/Jun. 2017, pp. 3411–3417.

[32] T. Kalmár-Nagy, R. D'Andrea, and P. Ganguly, "Near-optimal dynamic trajectory generation and control of an omnidirectional vehicle," *Robot. Auton. Syst.*, vol. 46, no. 1, pp. 47–64, 2004.

[33] M. Yuan, Z. Chen, B. Yao, and J. Hu, "A general online trajectory planning framework in the case of desired function unknown in advance," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 1–10, Oct. 2018, doi: 10.1109/TII.2018.2869823.

[34] M. Yuan, Z. Chen, B. Yao, and J. Hu, "An improved online trajectory planner with stability-guaranteed critical test curve algorithm for generalized parametric constraints," *IEEE/ASME Trans. Mechatron.*, vol. 23, no. 5, pp. 2459–2469, Oct. 2018.

[35] S. Kucuk, "Optimal trajectory generation algorithm for serial and parallel manipulators," *Robot. Comput.-Integr. Manuf.*, vol. 48, no. 1, pp. 219–232, Dec. 2017.

[36] H. Liu, X. Lai, and W. Wu, "Time-optimal and jerk-continuous trajectory planning for robot manipulators with kinematic constraints," *Robot. Comput.-Integr. Manuf.*, vol. 29, no. 2, pp. 309–317, 2013.

[37] F. Pfeiffer and R. Johanni, "A concept for manipulator trajectory planning," *IEEE Trans. Robot. Autom.*, vol. LRA-3, no. 2, pp. 115–123, Apr. 1987.

[38] A. Obradović, J. Vuković, N. Mladenović and Z. Mitrović, "Time optimal motions of mechanical system with a prescribed trajectory," *Meccanica*, vol. 46, no. 4, pp. 803–816, 2011.

[39] J. Hauser, "A projection operator approach to the optimization of trajectory functionals," *IFAC World Congr.*, vol. 35, no. 1, pp. 377–382, 2002.

[40] K. G. Shin and N. D. McKay, "Minimum-time control of robotic manipulators with geometric path constraints," *IEEE Trans. Autom. Control*, vol. AC-30, no. 6, pp. 531–541, Jun. 1985.

[41] J. E. Bobrow, S. Dubowsky, and J. S. Gibson, "Time-optimal control of robotic manipulators along specified paths," *Int. J. Robot. Res.*, vol. 4, no. 3, pp. 3–17, Sep. 1985.

[42] S. S. Pchelkin *et al.*, "On orbital stabilization for industrial manipulators: Case study in evaluating performances of modified PD+ and inverse dynamics controllers," *IEEE Trans. Control Syst. Technol.*, vol. 25, no. 1, pp. 101–117, Jan. 2017.

[43] T. Faulwasser, T. Weber, P. Zometa, and R. Findeisen, "Implementation of nonlinear model predictive path-following control for an industrial robot," *IEEE Trans. Control Syst. Tech.*, vol. 25, no. 4, pp. 1505–1511, Jul. 2017.

[44] Q.-C. Pham, "A general, fast, and robust implementation of the time-optimal path parameterization algorithm," *IEEE Trans. Robot.*, vol. 30, no. 6, pp. 1533–1540, Dec. 2014.

[45] P. Shen, X. Zhang, and Y. Fang, "Essential properties of numerical integration for time-optimal path-constrained trajectory planning," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 888–895, Apr. 2017.

[46] L. Žlajpah, "On time optimal path control of manipulators with bounded joint velocities and torques," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 1996, pp. 1572–1577.

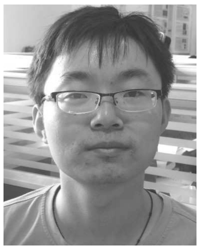
[47] F. Lamiraux and J.-P. Laumond, "From paths to trajectories for multi-body mobile robots," in *Proc. 5th Int. Symp. Exp. Robot.*, 1998, pp. 301–309.

[48] D. Knuth, *The Art of Computer Programming: Fundamental Algorithms*. Reading, MA, USA: Addison-Wesley, 1997.

[49] Y. Zhang, X. Zhang, Y. Fang, and P. Shen, "Dead reckoning and tracking control of omnidirectional mobile robots with active caster wheels," *Robot*, vol. 37, no. 3, pp. 361–368, 2015.

[50] Y. Li, C. Tang, S. Peeta, and Y. Wang, "Nonlinear consensus-based connected vehicle platoon control incorporating car-following interactions and heterogeneous time delays," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 9, pp. 1–11, Sep. 2018, doi: 10.1109/TITS.2018.2865546.

[51] Y. Li *et al.*, "Nonlane-discipline-based car-following model for electric vehicles in transportation- cyber-physical systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 1, pp. 38–47, Jan. 2018.



**PEIYAO SHEN** received the B.S. degree in intelligence science and technology from Xidian University, Xi'an, China, in 2014. He is currently pursuing the Ph.D. degree with the Institute of Robotics and Automatic Information Systems, Nankai University, Tianjin, China.

His research interest includes motion planning of omnidirectional wheeled mobile robots.

**XUEBO ZHANG** (M'12–SM'17) received the B.Eng. degree in automation from Tianjin University, Tianjin, China, in 2002, and the Ph.D. degree in control theory and control engineering from Nankai University, Nankai, China, in 2011.

He is currently an Associate Professor with the Institute of Robotics and Automatic Information System and is also with the Tianjin Key Laboratory of Intelligent Robotics, Nankai University. His research interests include mobile robotics, motion planning, and visual serving.

He served as the Organization Chair of IEEE-CYBER 2018, the General Co-Chair of IEEE-CYBER 2017, and will serve as the Program Chair of IEEE RCAR 2019. He is an Associate Editor of the *ASME Journal of Dynamic Systems, Measurement, and Control*.

**YONGCHUN FANG** (S'00–M'02–SM'08) received the B.S. and M.S. degrees in control theory and applications from Zhejiang University, Hangzhou, China, in 1996 and 1999, respectively, and the Ph.D. degree in electrical engineering from Clemson University, Clemson, SC, USA, in 2002.

From 2002 to 2003, he was a Post-Doctoral Fellow with the Sibley School of Mechanical and Aerospace Engineering, Cornell University, Ithaca, NY, USA. He is currently a Professor with the Institute of Robotics and Automatic Information Systems, Nankai University, Tianjin, China. His research interests include nonlinear control, visual serving, control of underactuated systems, and AFM-based nano-systems.

Dr. Fang is an Associate Editor of the *ASME Journal of Dynamic Systems, Measurement, and Control* and the *Journal of Control Theory and Applications*.