Boundary Layer Heuristic for Search-Based Nonholonomic Path Planning in Maze-Like Environments

Changliu Liu, Yizhou Wang and Masayoshi Tomizuka

***Abstract*— Automatic valet parking is widely viewed as a milestone towards fully autonomous driving. One of the key problems is nonholonomic path planning in maze-like environments (e.g. parking lots). To balance efficiency and passenger comfort, the planner needs to minimize the length of the path as well as the number of gear shifts. Lattice A* search is widely adopted for optimal path planning. However, existing heuristics do not evaluate the nonholonomic dynamic constraint and the collision avoidance constraint simultaneously, which may mislead the search. To efficiently search the environment, the boundary layer heuristic is proposed which puts large cost in the area that the vehicle must shift gear to escape. Such area is called the boundary layer. A simple and efficient geometric method to compute the boundary layer is proposed. The admissibility and consistency of the additive combination of the boundary layer heuristic and existing heuristics are proved in the paper. The simulation results verify that the introduction of the boundary layer heuristic improves the search performance by reducing the computation time by 56.1%.**

I. Introduction

Autonomous driving is a promising technology to revolutionize today's transportation system. Among many applications of this technology, automatic valet parking [1] is widely regarded as a milestone towards fully autonomous driving. However, path planning in parking lots is challenging due to (i) the complication of the nonholonomic vehicle dynamics, and (ii) the constraint imposed by the tight, cluttered and maze-like environment. To balance efficiency and passenger comfort, the objective of path planning in such environment is to find an optimal, safe and feasible path that minimizes the travel distance as well as the number of gear shifts.

Existing path planning methods include search-based methods such as A* or D* search [2], sampling-based methods such as rapidly-exploring random tree (RRT) [3], and dynamic programming [4]. A detailed survey can be found in [5]. This paper focuses on search-based methods, especially the lattice A* method [6] which directly encodes the nonholonomic vehicle dynamics and the geometry of the environment in the search graph.

A search graph contains nodes and links. Each node is a discrete vehicle state including vehicle position, heading and moving direction. Two nodes are directly connected by a link only if all constraints are satisfied. Starting from the initial node, the graph will be traversed until the goal node is reached. The order to traverse the graph depends on the objective of the path planning problem. To efficiently guide the search, two types of heuristics have been developed as discussed in [2] and [7]: *non-holonomic-without-obstacles* and *holonomic-with-obstacles*. As their names imply, the *non-holonomic-without-obstacles* heuristic evaluates the objective considering only the nonholonomic dynamic constraint, while the *holonomic-with-obstacles* heuristic evaluates the objective considering only the collision avoidance constraint. However, these heuristics ignore the cost on gear shifts induced by the interaction between the nonholonomic dynamic constraints and the collision avoidance constraint. They may be bad guides in certain cases. Recently, methods to simultaneously evaluate the nonholonomic constraint and the collision avoidance constraint have been proposed in [8] and [9]. However, these heuristics do not directly approximate the cost on gear shifts, which may limit their capabilities to address diverse planning objectives.

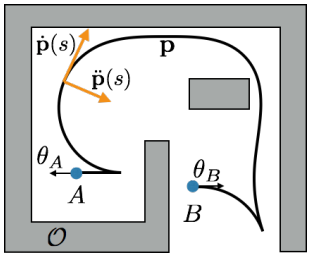


Fig. 1: The path planning problem in a maze-like environment.

In this paper, the boundary layer heuristic is proposed to estimate the cost on gear shifts induced by the interaction between the nonholonomic dynamic constraints and the collision avoidance constraint. Borrowed from fluid dynamics, the concept of boundary layer is defined as the area in the proximity of obstacles such that once a nonholonomic vehicle enters, it must shift gear to escape. Informed with the position of the boundary layers, the vehicle can better evaluate the consequences of getting into those areas, and hence reach the goal node faster. It will be shown that (i) the computation of the boundary layers is purely geometric, and (ii) the additive combination of the boundary layer heuristic and existing heuristics are admissible and consistent.

The remainder of the paper is organized as follows: the optimal path planning problem is formulated in Section II; Section III describes the lattice A* method and existing heuristics; Section IV introduces the boundary layer heuristic and proves the admissibility and consistency of this heuristic; Section V illustrates the performance of different heuristics; Section VI concludes the paper.

C. Liu and M. Tomizuka are with the Department of Mechanical Engineering, University of California, Berkeley, CA 94720 USA (e-mail: changliuliu, tomizuka@berkeley.edu).

Y. Wang is with Faraday & Future Inc, Gardena, CA 90248 USA yizhou.wang@faradayfuture.com

II. Problem Formulation

A. The Problem

Denote the pose of a vehicle as (x, y, θ) where x and y denote the planar position of the vehicle and θ is the vehicle heading. The path planning problem in a maze-like environment illustrated in Fig.1 is formulated below.

Problem: Given the start pose $A = (x_A, y_A, \theta_A)$, the end pose $B = (x_B, y_B, \theta_B)$, we need to find a shortest vehicle path connecting A and B which avoids all obstacles and satisfies the nonholonomic constraint with minimum length of backward driving and minimum number of gear shifts.

B. The Path and the Constraints

Let $\mathbf{p} : [0, s^*] \to \mathbb{R}^2$ denotes a path which is parameterized by length s. The total length of the path is s^*. $\mathbf{p}(s) \in \mathbb{R}^2$ is a point on the path $\mathbf{p}$ which has distance s to the starting point $\mathbf{p}(0)$. $\dot{\mathbf{p}} := \partial\mathbf{p}/\partial s$ and $\ddot{\mathbf{p}} := \partial^2\mathbf{p}/\partial s^2$ denote the tangent vector and the normal vector of the path respectively. Since the path $\mathbf{p}$ is parameterized by length, the length of its normal vector represents the curvature of the path. The nonholonomic dynamic constraint requires that the curvature of the path be bounded by maximum curvature κ_{max}.

Let $\mathbf{d} : [0, s^*] \to \{-1, 1\}$ be the indicator of driving direction along the path $\mathbf{p}$. At $\mathbf{p}(s)$, $\mathbf{d}(s) = 1$ means forward driving and $\mathbf{d}(s) = -1$ means backward driving. The discontinuous points on $\mathbf{d}$ represent gear shifts. Define $\delta(\mathbf{d})$ be the function that counts the number of gear shifts.

Let $\mathbf{T}_{a\to b}$ be a trajectory $[\mathbf{p}, \mathbf{d}]$ connecting a and b. The states a and b specify boundary conditions for the trajectory.

The area occupied by obstacles in the environment is denoted by $\mathcal{O} \subset \mathbb{R}^2$. Define $\mathcal{O}^\theta \subset \mathbb{R}^2$ to be the obstacle in the configuration space [10] given vehicle heading θ. $\mathcal{O}^\theta$ is an enlarged set of $\mathcal{O}$. Since the vehicle is not a point mass but occupies a finite area, $(x, y) \notin \mathcal{O}^\theta$ ensures that any part of the vehicle is outside of $\mathcal{O}$. For simplicity, we identify $\mathcal{O}^\theta$ with $\mathcal{O}$ in this paper. Refer to [11] for shape-aware planning.

C. Mathematical Formulation of the Path Planning Problem

The cost related to a trajectory is defined as

$$J(\mathbf{T}_{A\to B}) := \int_0^{s^*} c_s(\mathbf{d}(s))ds + c_g\delta(\mathbf{d}). \tag{1}$$

The first term $J_1(\mathbf{T}_{A\to B}) := \int_0^{s^*} c_s(\mathbf{d}(s))ds$ penalizes the weighted traveling distance where $c_s : \{1, -1\} \to \mathbb{R}^+$ is the unit cost for forward or backward driving. Typically $c_s(-1) > c_s(1)$. The second term $J_2(\mathbf{T}_{A\to B}) := c_g\delta(\mathbf{d})$ penalizes the number gear shifts with unit cost $c_g \in \mathbb{R}^+$. Based on the previous discussion, the following optimization is formulated for the path planning problem,

$$\begin{aligned} \min_{\mathbf{T}_{A\to B}} \quad & J(\mathbf{T}_{A\to B}) && (2)\\ & \|\ddot{\mathbf{p}}(s)\| \le \kappa_{max} && (3)\\ & \mathbf{p}(s) \notin \mathcal{O}^{\theta(s)}. && (4) \end{aligned}$$

Equation (2) is the cost function. Equation (3) is the constraint on the curvature of the path regarding the nonholonomic vehicle dynamics. Equation (4) is the constraint for obstacle avoidance where $\theta(s) = \arctan(\dot{\mathbf{p}}(s))$.

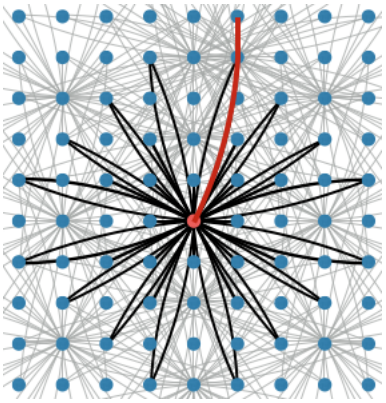


Fig. 2: Illustration of the lattice in 2D. The links from the red node are shown in black and other links are shown in gray (not all links are shown). The red curve represent a feasible trajectory starting from the red node.

As implied by the cost function $J(\mathbf{T}_{A\to B})$, in a tight environment, the forward driving distance, the backward driving distance, and the number of gear shifts should be balanced. However, the problem (2-4) is in general hard to solve as it is nonlinear and non-convex which contains both continuous and discrete decision variables. Search-based methods such as lattice A* search are usually adopted to solves (2-4) approximately, which will be briefly reviewed in the following section.

III. Lattice A* Search

A. The Lattice and The Search Algorithm

The state space for problem (2-4) has four dimensions (x, y, θ, d). A lattice [6] contains a set of discretized states (e.g. nodes) and a set of links connecting the nodes where every link represents a feasible path regarding constraints (3) and (4). The planning problem (2-4) can be solved approximately using graph search on the lattice as shown in Fig.2. The task is to traverse the graph from the starting nodes $(x_A, y_A, \theta_A, \pm 1)$ using A* algorithm until either of the goal nodes $(x_B, y_B, \theta_B, \pm 1)$ is found. The notations used in the graph search are defined below:

- Node $\mathcal{N}_i$ is a discretized state $(x_i, y_i, \theta_i, d_i)$ for index i.
- Link $\mathcal{L}_{i,j}$ is a spiral curve connecting node $\mathcal{N}_i$ and $\mathcal{N}_j$ which does not intersect with any obstacles and satisfies the maximum curvature constraint as shown in Fig.2. The cost associated with a link is $c_s(d_j)\|\mathcal{L}_{i,j}\| + c_g\frac{|d_i - d_j|}{2}$ where $\|\mathcal{L}_{i,j}\|$ computes the length of the link.
- Successor $S(\mathcal{N}_i)$ of node $\mathcal{N}_i$ is a set of nodes such that there is a link between $\mathcal{N}_i$ and any $\mathcal{N}_j \in S(\mathcal{N}_i)$.
- Forward cost $F(\mathcal{N}_i)$ is a summation of all link costs from the start node to $\mathcal{N}_i$.
- Heuristic cost $H(\mathcal{N}_i)$ is the estimation of the minimum cost from $\mathcal{N}_i$ to the goal nodes.
- Total cost $T(\mathcal{N}_i) = F(\mathcal{N}_i) + H(\mathcal{N}_i)$.
- Fringe of the search problem contains a list of nodes that has not been expanded.
- Closed set $V(\mathcal{N}_i)$ of the search problem indicates whether node $\mathcal{N}_i$ has been expanded before.

initialize $V(\mathcal{N})$ as zero for all nodes $\mathcal{N}$;
insert $(x_A, y_A, \theta_A, 1)$ and $(x_A, y_A, \theta_A, -1)$ to fringe;
while *True* **do**
  if *fringe is empty* **then**
    return failure;
  end
  Pop node $\mathcal{N}^*$ from fringe with smallest $T(\mathcal{N}^*)$;
  if $\mathcal{N}^*$ *is the goal* **then**
    return $\mathcal{N}^*$;
  end
  if $V(\mathcal{N}^*) = 0$ **then**
    $V(\mathcal{N}^*) := 1$;
    for *child-node in* $S(\mathcal{N}^*)$ **do**
      insert child-node to fringe;
    end
  end
end

Algorithm 1: Lattice A* Search.

The pseudo code is shown in Algorithm 1.

B. The Heuristics

Heuristics are approximations of the cost-to-go function, which are needed to guide the search as the search space is very large. The cost-to-go function with respect to (2-4) is

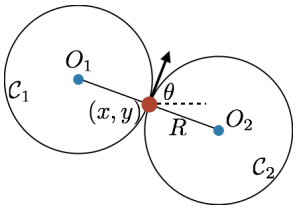
$$J^*(\mathcal{N}_i, \mathcal{N}_j) := \min_{\mathbf{T}_{\mathcal{N}_i \to \mathcal{N}_j}, \|\ddot{\mathbf{p}}\| \le \kappa_{max}, \mathbf{p} \notin \mathcal{O}^\theta} J(\mathbf{T}_{\mathcal{N}_i \to \mathcal{N}_j}). \quad (5)$$

In order for the A* search to find the optimal path, the heuristic needs to be admissible, e.g. $H(\mathcal{N}) \le J^*(\mathcal{N}, B)$, and consistent, e.g. $H(\mathcal{N}_i) - H(\mathcal{N}_j) \le J^*(\mathcal{N}_i, \mathcal{N}_j)$ [12].

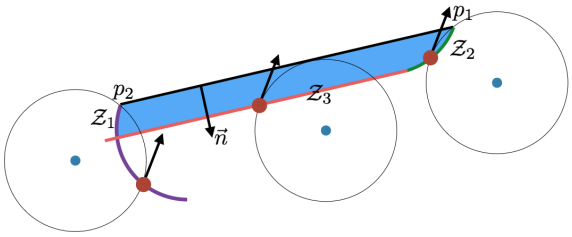
In literature, there are two types of heuristics as discussed in Section I: *non-holonomic-without-obstacles* heuristic h_f and *holonomic-with-obstacles* heuristic h_D. The *non-holonomic-without-obstacles* heuristic $h_f(\mathcal{N})$ equals to the minimized $J(\mathbf{T}_{\mathcal{N} \to B})$ subject to constraint (3), which is also called the free-space heuristic. The *holonomic-with-obstacles* heuristic $h_D(\mathcal{N})$ equals to the minimized $J(\mathbf{T}_{\mathcal{N} \to B})$ subject to constraint (4). As the vehicle is holonomic, there is no backward driving and gear shifts. $J(\mathbf{T}_{\mathcal{N} \to B})$ reduces to the length of path $\mathbf{p}$, i.e. the dijkstra distance from (x, y) to (x_B, y_B) given obstacles $\mathcal{O}$. Hence h_D is also called the D* heuristic.

Typically, the heuristic for A* search is chosen as $H(\mathcal{N}) = \max(h_f(\mathcal{N}), h_D(\mathcal{N}))$. As both h_f and h_D are admissible and consistent, H is admissible and consistent. Hence the search is optimal. However, as the free-space heuristic h_f is inefficient to compute online and it requires large memory space to store if computed offline, we only consider the D* heuristic h_D in this paper. h_D can be computed both online and offline very efficiently and do not require much memory to store the result.

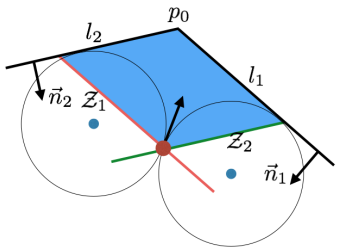
Nonetheless, gear shifts are sometimes inevitable for non-holonomic vehicles in a tight environment, which is not captured by the existing heuristics. A *non-holonomic-with-obstacles* heuristic is needed.



(a) The minimum tangent circles.



(b) The boundary layer for a line segment.



(c) The boundary layer for a concave corner.

Fig. 3: Computing the boundary layer for a fixed heading θ.

IV. Boundary Layer Heuristic

A. Defining the Boundary Layer

The definition of the boundary layer concerns with the moving direction of the vehicle instead of the vehicle heading. Given state (x, y, θ, d), without loss of generality, we assume $d = 1$. If $d = -1$, the following analysis is equivalent to $(x, y, \theta + \pi, 1)$. Define the boundary layer $\mathcal{B} \subset \mathbb{R}^2 \times [0, 2\pi)$ to be a collection of poses (x, y, θ) such that there does not exist a path from the pose to the goal pose B that satisfies constraints (3) and (4) and does not contain any gear shift, e.g.

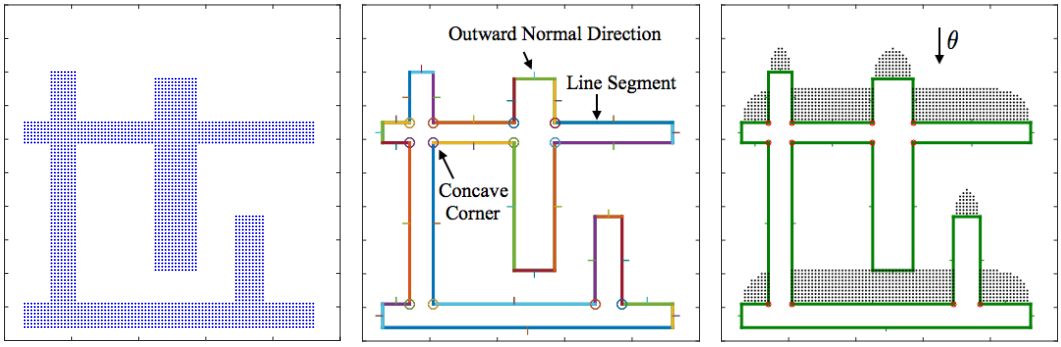
$$\mathcal{B}(\mathcal{O}) := \{(x, y, \theta) \in \mathbb{R}^2 \times [0, 2\pi) | \nexists \mathbf{T}_{(x,y,\theta) \to B}, \\ \text{s.t.} \|\ddot{\mathbf{p}}(s)\| \le \kappa_{max}, \mathbf{p}(s) \notin \mathcal{O}^{\theta(s)}, \delta(d) = 0\}. \quad (6)$$

B. Computing the Boundary Layer

For an arbitrary environment, it is in general hard to fully determine the boundary layer without solving the path planning problem. In this paper, it is assumed that $\mathcal{O}$ is a collection of polygons. With this assumption, we only explore the expression of the boundary layer for line segments l and concave corners c as these two geometric objects can fully represent $\partial\mathcal{O}$. In the following analysis, we fix θ and computes

$$\mathcal{B}_\theta(\mathcal{O}) := \{(x, y) \in \mathbb{R}^2 | (x, y, \theta) \in \mathcal{B}(\mathcal{O})\}, \quad (7)$$

as $\mathcal{B}(\mathcal{O}) = \cup_\theta(\mathcal{B}_\theta(\mathcal{O}), \theta)$.



(a) The occupancy grid map. (b) The extracted geometric objects. (c) The computed boundary layer $\mathcal{B}_\theta$.

Fig. 4: Computing the boundary layer in mixed environment.

Define a minimum tangent circle as a circle of radius $R := 1/\kappa_{max}$ that passes (x, y) with tangent direction $v(\theta) := [\cos\theta, \sin\theta]$ as shown in Fig.3a. Considering the direction $v(\theta)$, the left circle centered at $O_1 := (x - R\sin\theta, y + R\cos\theta)$ is denoted as $\mathcal{C}_1$ and the right circle centered at $O_2 := (x + R\sin\theta, y - R\cos\theta)$ is denoted as $\mathcal{C}_2$. In the following discussion, the boundary layer is computed assuming the goal point is far from the identified geometric objects.

1) Boundary Layer for Line Segments: Suppose the two end points of a line segment l is $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$, with outward normal direction $\vec{n}$. p_1 is on the left of p_2 regarding the normal direction $\vec{n}$. If $\vec{n} \cdot v(\theta) < 0$, the boundary layer is nonempty, e.g. $\mathcal{B}_\theta(l) \neq \emptyset$ where $\emptyset$ denotes an empty set, and is enclosed by the line segment l and the following three curves: (i) the contour $\mathcal{Z}_1$ of (x, y) such that $\mathcal{C}_1$ passes p_2; (ii) the contour $\mathcal{Z}_2$ of (x, y) such that $\mathcal{C}_2$ passes p_1; (iii) the contour $\mathcal{Z}_3$ of (x, y) such that $\mathcal{C}_1$ or $\mathcal{C}_2$ is tangent to l, whichever is closer to l. The three curves are illustrated in Fig.3b. Note that it is possible that $\mathcal{Z}_1$ and $\mathcal{Z}_2$ intersects.

As $\mathcal{C}_2$ or $\mathcal{C}_1$ is tangent to l, the distance from $\mathcal{Z}_3$ to l is $d = R - R|\vec{v}(\theta - \frac{\pi}{2}) \cdot \vec{n}|$. Hence the area enclosed by l and $\mathcal{Z}_3$ satisfies

$$0 \leq (x - x_1, y - y_1) \cdot \vec{n} \leq d. \tag{8}$$

For $\mathcal{Z}_1$, $\|O_1 p_2\| = R$. The area enclosed by $\mathcal{Z}_1$ satisfies that

$$\overrightarrow{p_2 p_1} \cdot \overrightarrow{p_2 O_1} \geq 0 \text{ or } \|\overrightarrow{p_2 O_1}\| \leq R. \tag{9}$$

For $\mathcal{Z}_2$, $\|O_2 p_2\| = R$. The area enclosed by $\mathcal{Z}_2$ satisfies that

$$\overrightarrow{p_1 p_2} \cdot \overrightarrow{p_1 O_2} \geq 0 \text{ or } \|\overrightarrow{p_1 O_2}\| \leq R. \tag{10}$$

Then $\mathcal{B}_\theta(l)$ is defined by (8), (9) and (10).

2) Boundary Layer for Concave Corners: Consider a corner point is $p_0 = (x_0, y_0)$, the line on the left is denoted as l_1 with outward normal vector $\vec{n}_1$, and the line on the right is denoted as l_2 with outward normal vector $\vec{n}_2$. As illustrated in Fig.3c, the boundary layer for the corner, e.g. $\mathcal{B}_\theta(c)$, is enclosed by the lines l_1 and l_2 and the following two curves: (i) the contour $\mathcal{Z}_1$ of (x, y) such that $\mathcal{C}_1$ is tangent to l_2; (ii) the contour $\mathcal{Z}_2$ of (x, y) such that $\mathcal{C}_2$ is tangent to l_1. As $\mathcal{C}_2$ is tangent to l_1, the distance from $\mathcal{Z}_1$ to l_1 is

$$d_1 = R - R\vec{v}(\theta - \frac{\pi}{2}) \cdot \vec{n}_1. \tag{11}$$

As $\mathcal{C}_1$ is tangent to l_2, the distance from $\mathcal{Z}_2$ to l_2 is

$$d_2 = R - R\vec{v}(\theta + \frac{\pi}{2}) \cdot \vec{n}_2. \tag{12}$$

Then $\mathcal{B}_\theta(c) = \{(x, y) | (x - x_0, y - y_0) \cdot \vec{n}_1 \in [0, d_1], (x - x_0, y - y_0) \cdot \vec{n}_2 \in [0, d_2]\}$.

3) Boundary Layer for Mixed Environments: In a mixed environment, suppose the information of the obstacles is stored in an occupancy grid map. The first step is to find all line segments from the occupancy grid map together with their outward normal vectors. The second step is to find all concave corners from the set of line segments. Then we compute the boundary layer for each line segment and each concave corner. The process is illustrated in Fig.4. This method gives a good approximation of the true boundary layer defined in (6) if the goal point B is not in the computed $\mathcal{B}(\mathcal{O})$, which is assumed in this paper. Moreover, the computation is purely geometric.

C. The Boundary Layer Heuristic

Given the boundary layer $\mathcal{B}(\mathcal{O})$, the boundary layer heuristic is defined as

$$h_{BL}(\mathcal{N}) = \begin{cases} c_g & (x, y, \theta) \in \mathcal{B}(\mathcal{O}), d = 1 \\ c_g & (x, y, \theta + \pi) \in \mathcal{B}(\mathcal{O}), d = -1 \\ 0 & else \end{cases}. \tag{13}$$

In the following discussion, we show the admissibility and consistency of h_{BL} and $h_{BL} + h_D$. Since consistency implies admissibility, we will be focused on consistency.

1) Consistency of h_{BL}: For any two nodes $\mathcal{N}_i$ and $\mathcal{N}_j$, we need to show $h_{BL}(\mathcal{N}_i) - h_{BL}(\mathcal{N}_j) \leq J^*(\mathcal{N}_i, \mathcal{N}_j)$. By definition (13), $h_{BL}(\mathcal{N}_i) - h_{BL}(\mathcal{N}_j)$ can only be $\pm c_g$ or 0. Since $J^*(\mathcal{N}_i, \mathcal{N}_j) \geq 0$, we only need to consider the case $h_{BL}(\mathcal{N}_i) = c_g$ and $h_{BL}(\mathcal{N}_j) = 0$, i.e. $\mathcal{N}_i$ is in the boundary layer but $\mathcal{N}_j$ is not. By definition of $\mathcal{B}(\mathcal{O})$ in (6), the vehicle needs to shift gear to get out of boundary layer. Then $J^*(\mathcal{N}_i, \mathcal{N}_j) \geq c_g = h_{BL}(\mathcal{N}_i) - h_{BL}(\mathcal{N}_j)$. Hence the heuristic h_{BL} is consistent.

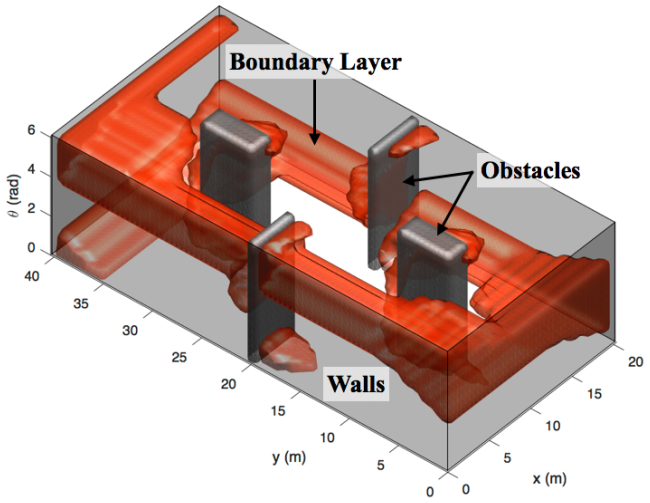


Fig. 5: Illustration of the boundary layer in 3D.

2) Consistency of $H^* := h_{BL} + h_D$: Consider any $\mathbf{T}_{\mathcal{N}_i \to \mathcal{N}_j}$ that satisfies constraints (3) and (4). By previous discussion, $h_{BL}(\mathcal{N}_i) - h_{BL}(\mathcal{N}_j) \leq c_g \delta(\mathbf{d}) = J_2(\mathbf{T}_{\mathcal{N}_i \to \mathcal{N}_j})$. Moreover, by triangular inequality, $h_D(\mathcal{N}_i) - h_D(\mathcal{N}_j)$ is smaller than or equal to $c_s(1)$ times the dijkstra distance between $\mathcal{N}_i$ and $\mathcal{N}_j$, hence is smaller than $J_1(\mathbf{T}_{\mathcal{N}_i \to \mathcal{N}_j})$. As $J = J_1 + J_2$, then $H^*(\mathcal{N}_i) - H^*(\mathcal{N}_j) \leq J(\mathbf{T}_{\mathcal{N}_i \to \mathcal{N}_j})$. Since $\mathbf{T}_{\mathcal{N}_i \to \mathcal{N}_j}$ is arbitrary, then $H^*(\mathcal{N}_i) - H^*(\mathcal{N}_j) \leq J^*(\mathcal{N}_i, \mathcal{N}_j)$. So H^* is consistent.

Consistency of H^* implies that lattice A* search algorithm with heuristic $h_{BL} + h_D$ is optimal.

V. Performance

In this section, we illustrate the performance of the A* search using different heuristics. The test scenario is a closed $20m \times 40m$ ground with several obstacles, which is the test ground for the unmanned vehicle contest in World Robotics Conference (WRC) 2016 in Beijing, China. The computed boundary layer for the test environment is shown in 3D in Fig.5. The resolution of the lattice is 0.5m for the xy plane. 16 discrete headings are chosen. The primitive set is shown in Fig.2.

In this paper, the start pose and the ending pose are chosen as $A = (22.5m, 15m, 0deg)$ and $B = (8m, 5m, 0deg)$. To illustrate the performance of h_{BL}, two sets of optimization parameters are chosen. In case 1, the parameters are chosen to be $c_s(1) = c_s(-1) = 1$ and $c_g = 15$. In case 2, the gear cost is $c_g = 50$ while other parameters remain the same. For both cases, the path planning problem is solved using (i) uniform cost search (which is equivalent to A* search with zero heuristic); (ii) A* search with h_D; and (iii) A* search with $h_D + h_{BL}$. Algorithm 1 was run in C++ on a virtual machine on Macbook of 2.3 GHz using Intel Core i7. The performance (e.g. optimal cost, maximum size of fringe, number of iterations and computation time) of the A* search using different heuristics in the two cases is shown in Table I. The optimal paths (which are not unique) and the nodes that were expanded during the search are shown in Fig.6. In those figures, the color of a 2D point represents the number of times that a node associated with that point is expanded. Regarding the results from the tables and the figures, we conclude that,

TABLE I: The performance of different heuristics.

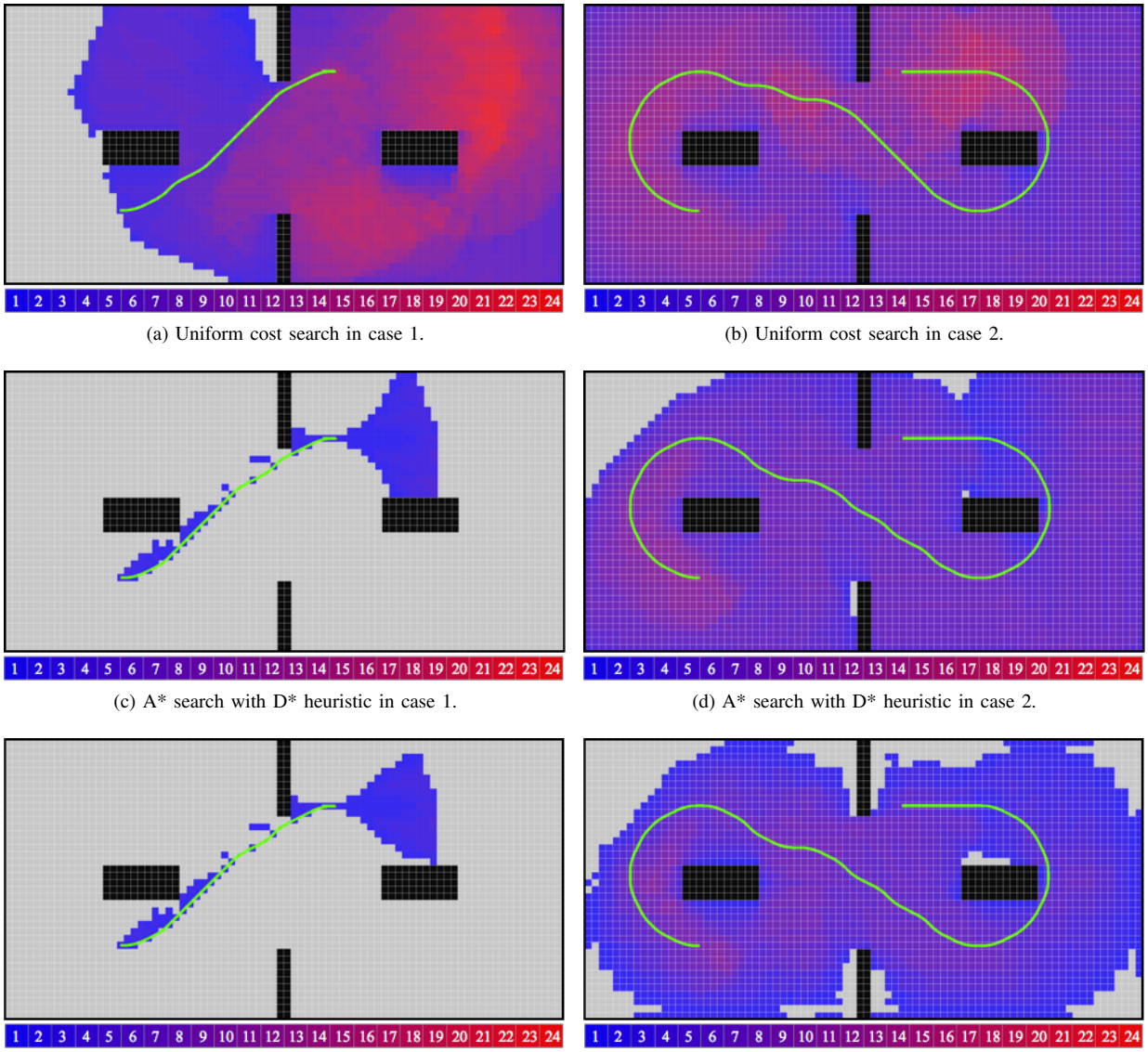
	Heuristic	Cost	Max Fringe Size	Iterations	Time (ms)
Case 1	0	35.6421	23066	25089	2418
	h_D	35.6421	892	402	10
	$h_D + h_{BL}$	35.6421	783	330	8
Case 2	0	62.6127	30725	28387	2341
	h_D	62.6127	22778	19114	1309
	$h_D + h_{BL}$	62.6127	17632	11421	574

- The heuristic $h_D + h_{BL}$ is admissible and consistent since A* search with $h_D + h_{BL}$ finds the same optimal cost as uniform cost search does.
- The heuristic $h_D + h_{BL}$ has more advantage over the heuristic h_D when the gear cost is larger. In case 1 when the gear cost is relatively small, the maximum size of the fringe, the number of iterations and the computation time reduce 12.2%, 17.9% and 20.0% respectively by adding the BL heuristic. In case 2 when the gear cost is relatively big, the number is 22.6%, 40.2% and 56.1% respectively.
- The improved performance of the heuristic $h_D + h_{BL}$ is due to the fact that the A* solver visits the points in the boundary layer less frequently as shown in Fig.6d and Fig.6f.
- The optimal path changes a lot when the optimization parameters changes. When the cost for gear shift is small, the optimal trajectory is to move forward a little bit and then drive backward to the destination as shown in Fig.6a. When the cost for gear shift is large, the optimal trajectory is to detour and drive forward to the destination as shown in Fig.6a.

Note that due to discretization, the path has some unnecessary oscillations. Trajectory smoothing [13] is needed before sending the path to low level controller for execution.

VI. Conclusion

In this paper, the optimization problem for nonholonomic path planning in a maze-like environment is formulated. To find a safe, feasible and comfortable path to the destination, the vehicle needs to minimize the travel distance and the number of gear shifts. Lattice A* search is used to solve the problem. However, existing heuristics may provide bad guides to the search as the interaction between the nonholonomic dynamic constraint and the collision avoidance constraint is ignored. The boundary layer heuristic is proposed in this paper, which assigns a large cost to the nodes in the boundary layer, e.g. area that the vehicle must shift gear to escape. The computation of the boundary layer is purely geometric. It is proved in the paper that the additive combination of the boundary layer heuristic and the D* heuristic is consistent, hence optimal. By introducing the boundary layer heuristic, the total computation time can reduce 56.1% when the cost of gear shift is relatively large.



(a) Uniform cost search in case 1.

(b) Uniform cost search in case 2.

(c) A* search with D* heuristic in case 1.

(d) A* search with D* heuristic in case 2.

(e) A* search with D* and BL heuristics in case 1.

(f) A* search with D* and BL heuristics in case 2.

Fig. 6: Illustration of the results in the two cases.

REFERENCES

[1] C. Löper, C. Brunken, G. Thomaidis, S. Lapoehn, P. P. Fouopi, H. Mosebach, and F. Köster, "Automated valet parking as part of an integrated travel assistance," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2013, pp. 2341–2348.

[2] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *The International Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, 2010.

[3] L. Han, Q. H. Do, and S. Mita, "Unified path planner for parking an autonomous vehicle based on RRT," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 5622–5627.

[4] G. Schildbach and F. Borrelli, "A dynamic programming approach for nonholonomic vehicle maneuvering in tight environments," in *IEEE Intelligent Vehicles Symposium (IV)*, 2016, pp. 151–156.

[5] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, 2016.

[6] M. Pivtoraiko and A. Kelly, "Generating near minimal spanning control sets for constrained motion planning in discrete state spaces," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2005, pp. 3231–3237.

[7] M. Likhachev and D. Ferguson, "Planning long dynamically feasible maneuvers for autonomous vehicles," *The International Journal of Robotics Research*, vol. 28, no. 8, pp. 933–945, 2009.

[8] S. Yoon, S. E. Yoon, U. Lee, and D. H. Shim, "Recursive path planning using reduced states for car-like vehicles on grid maps," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 5, pp. 2797–2813, 2015.

[9] C. Chen, M. Rickert, and A. Knoll, "Path planning with orientation-aware space exploration guided heuristic search for autonomous parking and maneuvering," in *IEEE Intelligent Vehicles Symposium (IV)*, 2015, pp. 1148–1153.

[10] T. Lozano-Pérez, *Spatial Planning: A Configuration Space Approach*. Springer New York, 1990, pp. 259–271.

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

[12] S. Russell and P. Norvig, "Artificial intelligence: A modern approach," *Prentice Hall*, vol. 25, p. 27, 1995.

[13] C. Liu, C.-Y. Lin, Y. Wang, and M. Tomizuka, "Convex feasible set algorithm for constrained trajectory smoothing," in *American Control Conference (ACC)*, 2017, p. to appear.