IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Perpendicular Parking of Car-like Robots Allowing a Cusp on the Path

JONGHOEK KIM[1]

[1]J. Kim is with Electrical and Electronic Convergence Department, Hongik University, Sejong 30016, South Korea (e-mail: jonghoek@hongik.ac.kr).

Corresponding author: Jonghoek Kim (e-mail: jonghoek@hongik.ac.kr).

This work was supported by the National Research Foundation (NRF) of Korea grant funded by the Korea government (MSIT) (No. 2019057282)

ABSTRACT This paper introduces the perpendicular parking algorithm of car-like robots, such that the generated path consists of a cusp. This path planning is based on the car's turning radii, which can be determined by the car's geometry and its maximum steering angle. As far as we know, this paper is novel in developing autonomous perpendicular parking based on circular arc and straight line segments, such that a cusp on the generated path is allowed. Since a cusp is allowed, the proposed parking approach is suitable for parking in a small space. The simulation results show the validity of the proposed approach.

INDEX TERMS Autonomous car; Autonomous parking; Autonomous perpendicular parking; Path planning; Small space parking;

I. INTRODUCTION

Even for experienced drivers, parking can be a difficult task, especially in the case where parking spots are very narrow. The research effort in autonomous parking has result in an extensive literature [1]–[7]. The automobile industry has already started producing several cars with parking assistants which can actively control acceleration, breaking and steering, the research interest in the automated parking is still strong.

Autonomous vehicles must perform several tasks simultaneously, such as lane following while avoiding collision [8]–[12] or autonomous parking [5]–[7], [13]–[17]. Since an autonomous car must perform various tasks simultaneously, it is highly desirable that a car performs a task using a minimal computational load. This requirement for low computational load inspired us to develop a path planning method which is computationally efficient.

Many papers [5]–[7], [13]–[17] considered autonomous parallel parking. Perpendicular parking was handled in several papers, such as [2]–[4], [18], [19]. The path planning method in [2] does not work for an arbitrary initial position. In [3], [4], a constrained optimization is utilized to generate a path without collision. Here, constraints include collision-free condition and mechanical constraints. In [3], weighting parameters must be selected carefully to achieve a safe parking obeying the constraints.

When a human tries to park a car in a small parking spot, he or she drives the car back and forth to generate a path with a cusp. However, planning approaches in [3], [18], [20] considered feedback steering controls for perpendicular reverse parking in a single maneuver. [3], [18], [20] cannot generate a path with a cusp, which may restrict the car's motion in a small parking space.

[21] solved the path planning problem when the car cannot reverse and cusps are not allowed. [21] gives a sufficient set of paths, i.e., a set which always contains an optimal path. [21] showed that any optimal path can be described by one of 6 *words*: $LRL, LSR, RLR, RSR, RSL, LSL$, where L , R, and S stand for "go left", "go right", and "go straight", respectively. Here, "left" and "right" mean counter-clockwise or clockwise around a tightest possible circle, and of course a car always goes less than 2π around any circle. L or R is an arc of a circle, and S is a straight line segment. A word, such as LSL, stands for the corresponding class of paths consist of an arc or a straight line segment. Each character in a word is called the *motion*. [21] showed that any optimal path can be described by a word with three motions.

We acknowledge that the steering wheel can be turned to adjust the turning radius of the car. This implies that the turning radius of the car does not have to be fixed. However, this adjustable turning radius makes the path planning much more complicated, since the search space for path planning increases considerably. By fixing the turning radius, the path of the car is composed of circular arc and straight line seg-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2971250, IEEE Access

ments. Thus, planning problem becomes simple and intuitive.

Similarly to ours, the authors of [13] fixed the turning radius to enable the path planning for autonomous parallel parking. However, [13] is tailored for parallel parking and cannot be extended to perpendicular parking. The method presented in this paper is a geometric path planning method specifically tailored for perpendicular parking.

In [19], rapid-exploring random tree (RRT) algorithms were used to generate a safe path to the parking space. However, RRT algorithms generate random samples to build a path and does not generate the optimal path as long as the number of samples does not go to infinity. In [19], a start tree is generated having the car's initial position as its root, and a goal tree is generated having the goal position as its root. The start tree is related to the forward maneuver of the car, and the goal tree is related to the backward maneuver of the car. Then, a bidirectional RRT tree is generated as a union of the two trees. A single cusp appears at the point where the two trees meet, since the start tree and the goal tree are related to forward and backward maneuvers respectively.

This paper introduces the perpendicular parking algorithm of car-like robots, such that the generated path consists of a cusp. Inspired by [21], our approach uses a word to generate a path consist of arcs and straight line segments. We generate a path with multiple motion segments. This path planning is based on the car's minimum turning radii, which can be determined by the car's geometry and its maximum steering angle.

As far as we know, this paper is novel in developing autonomous perpendicular parking based on circular arc and straight line segments, such that a cusp on the generated path is allowed. Since a cusp is allowed, the proposed parking approach is suitable for parking in a small space.[1] We compare the proposed approach with [19] to verify the effectiveness of our approach.

The paper is organized as follows: Section II discusses definitions and assumptions related to our paper. Section III introduces our path planning for autonomous perpendicular parking. Section IV is devoted to steering and speed controls to follow the generated path. Section V introduces MATLAB simulations to verify our path plan method. Section VI provides conclusions.

II. DEFINITIONS AND ASSUMPTIONS

This paper assumes that obstacle environments are known a priori. However, in a real situation, it is not easy to sense the correct position of the obstacles. [3] proposed the sensing algorithm to generate obstacle information using multiple sensors (GPS, lidar, or radar). The empty parking place can be extracted using the approach in [22]. Thus, the goal of this paper is to generate a safe path to the parking space using the obstacle information, which is accessible using multiple sensors. Once a safe path is generated, then steering and speed are controlled to make the car follow the generated path.

We briefly introduce the sensing algorithm in [3] to generate obstacle information using multiple sensors (GPS, lidar, or radar). Suppose that the car is equipped laser sensors. The complete point cloud on close obstacles is obtained from the sensors. Then, an Euclidean Cluster Extraction algorithm is used to have each obstacle represented as a cluster. The orientation of each cluster is extracted by fitting a line model to the points belonging to the contour of the cluster using a RANSAC algorithm. The orientation of the bounding box will be equal to the orientation of the fitter line.

Without loss of generality, this paper considers a front wheel drive vehicle. The kinematic model of a car with front-wheel steering (non-holonomic system) is described as follows.

$$\dot{X} = v * F * (\cos(\theta), \sin(\theta), \tan(\phi)/L)^T. \quad (1)$$

Here, $X = (x, y, \theta)^T$, x and y are the Cartesian coordinates of the midpoint of the rear wheel axle. θ is the orientation angle of the car. v is the speed of the car. ϕ is the steering angle, and L is the wheel base (distance between the front and rear wheel axles). The steering ϕ and the speed v are control commands which drive the vehicle. F indicates whether the car moves forwards or backwards. If the car moves forwards, then we set $F = 1$. Otherwise, we set $F = -1$.

Inspired by [13], this paper presents how to plan the car's path which is followed by the midpoint of the rear axle. [23] tackled a path planning problem when the car can reverse and cusps are allowed. Inspired by [23], the movement of the car-like car is composed of the following six motions $L^+, L^-, R^+, R^-, S^+, S^-$. Here, L^+ indicates turning to the left while going forwards, and L^- indicates turning to the left while going backwards. R^+ indicates turning to the right while going forwards, and R^- indicates turning to the right while going backwards. S^+ indicates going straight forwards, and S^- indicates going straight backwards. A motion in $\{L^+, L^-, R^+, R^-\}$ is an arc of a tightest possible circle, and a motion in $\{S^+, S^-\}$ is a line segment.

Let r denote the radius of a tightest possible circle which is associated to a motion in $\{L^+, L^-, R^+, R^-\}$. r is set by the geometry of the car [13]. $r = \frac{L}{tan(\beta)}$ where L is the wheelbase, and β is the maximum steering angle [13]. As we increase the steering angle β, the turning radii decreases.

We handle the case where the parking space is to the right of the car. The case where the parking space is to the left of the car can also be handled similarly to the case presented in this paper. Our constraint is that the car is perpendicular to the parking space initially. This is what a driver usually does to perform perpendicular parking.

We make the car move backwards while entering the parking space. This backward parking is what a driver usually does to perform perpendicular parking in a narrow parking

[1] Since our parking approach does not need a driver sitting inside the car, the car can be parked in an extremely small space between two obstacles, such that a human driver cannot get out of the car blocked by the obstacles. Therefore, comfort of the driver is not important in this paper.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2971250, IEEE Access

space. It is worth to note that parking maneuvers with forward motions are seldom considered in the literature [3]. This backward parking maneuver is desirable considering the fact that once the parking is done, the driver can easily pull the car out of the parking space by just moving forwards.

Our goal is to search for a word, such that the word makes the perpendicular parking possible. We derive a word which starts from S^+ and ends at S^-. This implies that the car moves straight forwards initially, and it moves straight backwards lastly.

$InitP$ denotes the initial position of the car. Also, $GoalP$ denotes the goal position of the car. The line segment associated to S^+ intersects $InitP$, and the line segment associated to S^- intersects $GoalP$. These two line segments intersect at one point, say $InterP$. See Fig. 1. In this figure, the arrow direction indicates the movement direction of the car. Also, obstacles on both sides of the parking space are illustrated with two rectangles.

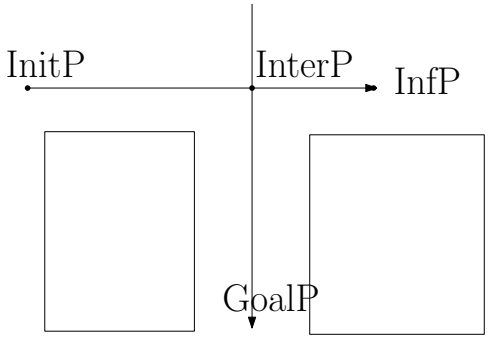


FIGURE 1. Illustration. In this figure, the arrow direction indicates the movement direction of the car. Also, obstacles on both sides of the parking space are illustrated with two rectangles.

Consider an infinite ray which is associated to S^+, such that the ray starts from $InterP$. Let $InfP$ denote a point on the infinite ray, such that the point is infinitely far from $InterP$. Let $Ray(InterP, InfP)$ denote the ray. $A(InitP, InterP, InfP)$ is an angle of $Ln(InterP, InitP)$ measured counter-clockwise from $Ray(InterP, InfP)$. Here, $Ln(InterP, InitP)$ denotes the line segment whose two end points are $InterP$ and $InitP$ respectively. In Fig. 1, $A(InitP, InterP, InfP)$ is 180 degrees.

III. PERPENDICULAR PARKING

A. $A(INITP, INTERP, INFP)$ IS 180 DEGREES

Consider the case where $A(InitP, InterP, InfP)$ is 180 degrees. We present a short word to perform perpendicular parking as follows :$S^+R^-S^-$. The car is perpendicular to the parking space initially. Thus, utilizing S^+, the car moves perpendicular to the parking space. Since the parking space is to the right of the car, R^- is required for parking while going backwards.

We present how to derive an arc path associated to R^-. We draw a circle with radius r, such that the circle is tangential to both $Ray(InterP, InfP)$ and $Ln(InterP, GoalP)$. Here, $Ln(InterP, GoalP)$ is the line segment whose two end points are $InterP$ and $GoalP$ respectively. See Fig. 2. In this figure, the dotted circle indicates the circle which is tangential to both $Ray(InterP, InfP)$ and $Ln(InterP, GoalP)$. An arc of this circle is utilized as an arc path associated to R^-. $Ray(InterP, InfP)$ is tangential to an arc path associated to R^- at a point, say $TangentP$.

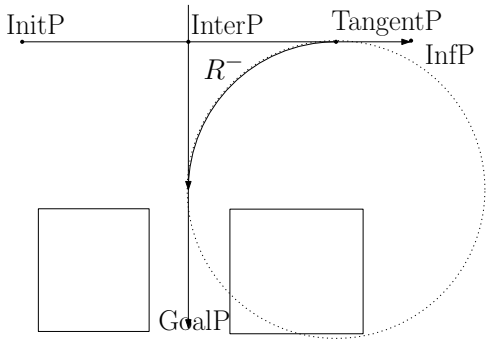


FIGURE 2. Illustration. The dotted circle indicates the circle which is tangential to both $Ray(InterP, InfP)$ and $Ln(InterP, GoalP)$. An arc of this circle is utilized as an arc path associated to R^-.

There may be a case where R^- maneuver in $S^+R^-S^-$ makes the car collide with an obstacle next to the parking space. See Fig. 3. In this figure, an arc of the dotted circle is utilized as an arc path associated to R^-. See that this arc path leads to collision.

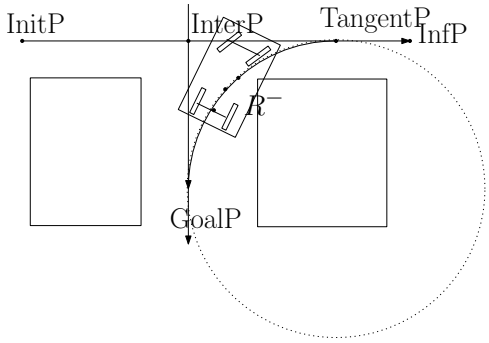


FIGURE 3. Illustration. In this figure, an arc of the dotted circle is utilized as an arc path associated to R^-. See that this arc path leads to collision.

B. $A(INITP, INTERP, INFP)$ IS LESS THAN 180 DEGREES.

If R^- maneuver in $S^+R^-S^-$ makes the car collide, then we gradually decrease $A(InitP, InterP, InfP)$ by changing the position of $InfP$. Fig. 3 shows the initial position of $InfP$. While fixing both $Ln(InterP, InitP)$ and $InterP$, we gradually change the position of $InfP$ so that $A(InitP, InterP, InfP)$ gradually decreases to 90 degrees. In our MATLAB simulations, we change the position of $InfP$ so that $A(InitP, InterP, InfP)$ decreases by 10 degrees at each iteration. Recall that an arc path associated to R^- is generated by drawing a circle with radius r, such that the circle is tangential to both $Ray(InterP, InfP)$ and $Ln(InterP, GoalP)$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2971250, IEEE Access

We present how to determine whether the path planning leads to collision at each iteration. We draw an arc path associated to the car's edge and check whether the arc meets an obstacle boundary. In the previous paragraph, we considered a circle with radius r, such that the circle is tangential to both $Ray(InterP, InfP)$ and $Ln(InterP, GoalP)$. Let $Center$ denote the center of the circle. Also, w is the width of the car. We draw a circle with radius $r - w/2$, such that the circle's center is $Center$. The arc of this circle is utilized as an arc path associated to the car's edge.

See Fig. 4 for an illustration. In this figure, B is utilized as an arc associated to the car's edge. In the case where B meets an obstacle boundary, the path planning at this iteration leads to collision. Thus, we need to change the position of $InfP$ so that $A(InitP, InterP, InfP)$ decreases at the next iteration.

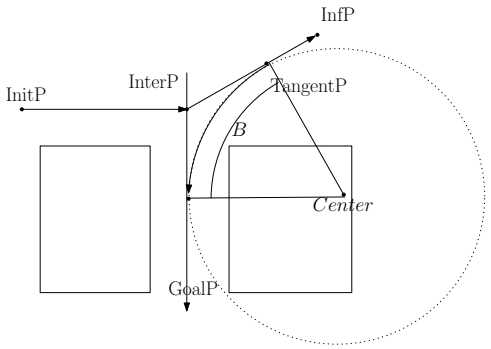


FIGURE 4. B is utilized as an arc path associated to the car's edge. In the case where B meets an obstacle boundary, the path planning at this iteration leads to collision.

While $A(InitP, InterP, InfP)$ decreases, we search for a moment when an arc path associated to R^- does not result in collision. In other words, we search for the case where B does not meet an obstacle. See Fig. 5 for an illustration of the case where an arc path associated to R^- does not result in collision. In this figure, $A(InitP, InterP, InfP)$ is 150 degrees.

Consider the case where $A(InitP, InterP, InfP)$ is less than 180 degrees. In this case, the car cannot move along the path composed of $Ln(InitP, InterP)$ and $Ln(InterP, TangentP)$, due to the maximum turn rate of the car. We thus need to smooth the path so that the car can traverse the path smoothly.

We next present how to smooth the path composed of $Ln(InitP, InterP)$ and $Ln(InterP, TangentP)$. We draw a circle with radius r, such that the circle is tangential to both $Ln(InitP, InterP)$ and $Ln(InterP, TangentP)$. Let t_1 denote a point on $Ln(InitP, InterP)$, at which point the tangential circle intersects $Ln(InitP, InterP)$. Let t_2 denote a point on $Ln(InterP, TangentP)$, at which point the tangential circle intersects $Ln(InterP, TangentP)$.

In Fig. 5, the tangential circle is illustrated with a dashed circle. In this figure, an arc of the dashed circle is utilized as an arc path associated to L^+.

The car starts from $InitP$ and moves along $Ln(InitP, t_1)$, which is associated to S^+. Then, it moves along an arc path associated to L^+. Thereafter, it moves along $Ln(t_2, TangentP)$, which is associated to S^+. Next, it traverses an arc path associated to R^-. Finally, it moves backwards to reach $GoalP$ (S^-). The word for these movements is $S^+L^+S^+R^-S^-$.

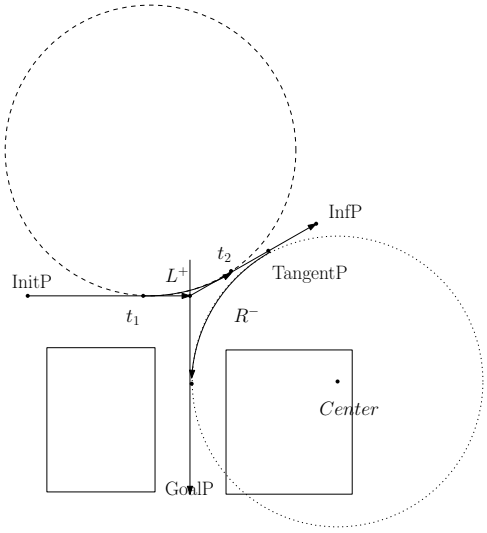


FIGURE 5. Illustration. In this figure, an arc of the dashed circle is utilized as an arc path associated to L^+.

1) $A(InitP, InterP, InfP)$ is 90 degrees

While fixing both $Ln(InterP, InitP)$ and $InterP$, we gradually change the position of $InfP$ so that $A(InitP, InterP, InfP)$ gradually decreases to 90 degrees.

Suppose that $A(InitP, InterP, InfP)$ decreases to 90 degrees. This case, we search for a safe path to avoid collision utilizing the following word :$S^+L^+S^-$. This word implies that we make the car turn left so that it does not collide with an obstacle as it moves backwards later.

See Fig. 6 for an illustration of the case where we use $S^+L^+S^-$ for parking maneuver. In this case, $Ray(InterP, InfP)$ and $Ln(InterP, GoalP)$ are parallel to each other. See that $A(InitP, InterP, InfP)$ is 90 degrees.

We draw a circle with radius r, such that the circle is tangential to $Ln(InitP, InterP)$ and $Ray(InterP, InfP)$. Let t_1 denote a point on $Ln(InitP, InterP)$, at which point the tangential circle intersects $Ln(InitP, InterP)$. Let t_2 denote a point on $Ray(InterP, InfP)$, at which point the tangential circle intersects $Ray(InterP, InfP)$.

In Fig. 6, the tangential circle is illustrated with a dashed circle. In this figure, an arc of the dashed circle is utilized as an arc path associated to L^+.

The car starts from $InitP$ and moves along $Ln(InitP, t_1)$, which is associated to S^+. Then, it moves along an arc path associated to L^+. Thereafter, it moves backwards along $Ln(t_2, GoalP)$, which is associated to S^-. See Fig. 6 for an illustration.

There may be a situation where the tangential point t_1

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2971250, IEEE Access

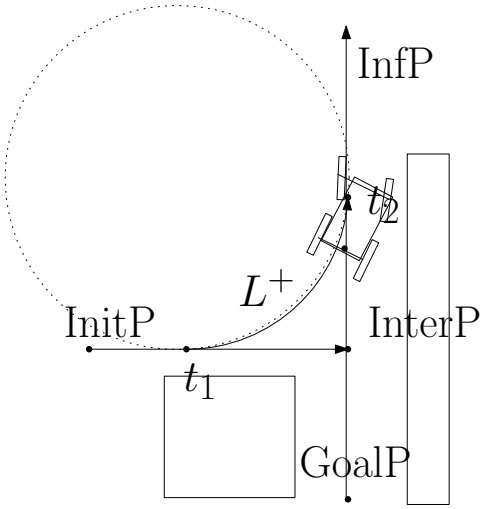


FIGURE 6. Illustration. In this figure, an arc of the dashed circle is utilized as an arc path associated to L^+. The word for this parking maneuver is $S^+L^+S^-$.

appears behind the car initially. This case, the car which is at $InitP$ needs to move backwards to reach t_1. This backward maneuver is associated to S^- not to S^+. Once the car reaches t_1, then it moves along the arc of the tangential circle until it reaches t_2. This maneuver is associated to l^+. Then, it moves backwards to reach the goal point. This maneuver is associated to S^-. In summary, the car moves according to $S^-l^+S^-$ maneuver. Thus, we generate the path ($S^+l^+S^-$ or $S^-l^+S^-$) connecting $InitP, t_1, t_2, GoalP$ in this order.

IV. SPEED AND STEERING CONTROLS TO FOLLOW THE GENERATED PATH

This section presents how to set speed and steering controls to follow the generated path. The generated path is composed of straight segments and circular arcs of minimal radius.

A circular arc can be followed by making the car turn with the maximum steering angle. This implies that whenever the car is on a circular arc, we set the steering of the car as the maximum steering angle.

A simple approach to make a car move along a straight segment is to set the steering of the car as zero. However, this open loop approach cannot assure that the car tracks the segment. In this paper, a straight segment is followed by making the car move towards the end point of the segment.

Thus, we use the following feedback control approach to make the car track a straight segment. Let (x_s, y_s) denote the end point of a straight segment. The heading of the car must head towards (x_s, y_s). Thus, the desired heading of the car is

$$\theta_d = tan^{-1}(\frac{y_s - y}{x_s - x}). \tag{2}$$

The error of the heading is

$$e_\theta = \theta_d - \theta. \tag{3}$$

We then change e_θ so that it exists between $-\pi$ and π. We use

$$e_{\theta_2} = tan^{-1}(\frac{s(e_\theta)}{c(e_\theta)}). \tag{4}$$

Then, we use the steering command as

$$\phi = \tanh(\frac{e_{\theta_2}}{C_1}) * C_2. \tag{5}$$

Here, C_1 and C_2 are positive constants. This control implies that in the case where $e_{\theta_2} > 0$, we set the steering control as $\phi > 0$. Also, in the case where $e_{\theta_2} < 0$, we set the steering control as $\phi < 0$. In simulations, we set C_2 as $\pi/180$. This implies that $\|\phi\|$ in (5) is bounded above by $\pi/180$, which is associated to 1 degree.

The speed of the car is controlled so that whenever the car finishes moving along a segment (a straight segment or a circular arc), it stops at the end of the segment.

Recall that v denotes the speed of the car. Also, a_m denotes the maximum acceleration of the car. The required time interval to decrease the car's speed from v to 0 is

$$T_r = v/a_m. \tag{6}$$

Using (6), the traversal distance of the car as it decreases its speed from v to 0 is

$$D = \frac{v^2}{2a_m}. \tag{7}$$

Suppose that the car is at one end point of a segment. Also, suppose that the length of the segment is L_S. Let L_t denote the traversal distance of the car along the segment. As the car travels the segment, L_t gradually increases from 0 to L_S.

In the case where $L_S - L_t > D$, we set the *speed command* of the car as v_{max}. In the case where $L_S - L_t \leq D$, we set the *speed command* of the car as 0. In this way, the speed of the car is controlled so that whenever the car finishes moving along a segment (a straight segment or a circular arc), it stops at the end of the segment.

We consider discrete-time systems. Let $V_k \in \{0, v_{max}\}$ denote the speed command at time step k. Also, let dt denote the sampling interval. Let v_k denote the speed of the car at time step k. We update v_k using the following rule:

$$v_{k+1} = v_k + sign(V_k - v_k) * a_m * dt. \tag{8}$$

Here, $sign(V_k - v_k)$ indicates the sign of $V_k - v_k$. Also, v_{max} is set as the upper bound for v_{k+1} so that v_{k+1} cannot be bigger than v_{max}. Moreover, 0 is set as the lower bound for v_{k+1} so that v_{k+1} cannot be smaller than 0.

V. MATLAB SIMULATIONS

In this section, we use MATLAB simulations to verify our autonomous parking method. The proposed approach is to decrease $A(InitP, InterP, InfP)$ gradually until it becomes 90 degrees. At each iteration, $A(InitP, InterP, InfP)$ decreases by 10 degrees.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2971250, IEEE Access

The simulation settings are as follows. $GoalP$ is (2,-1) in meters. The car length is 3 meters. $\beta = \pi/6$. The wheelbase (distance between the front and rear wheel axles) is 2.2 meters. $r = 2.2/tan(\pi/6)$ meters, and the car width is $w = 2.5$ meters. $C_1 = 0.001$, and $C_2 = \pi/180$.

We compare the proposed approach with [19] to verify the effectiveness of our approach. Under the RRT planner in [19], one cusp appears at the point where the start tree and the goal tree meet, since the start tree and the goal tree are related to forward and backward maneuvers respectively.

RRT planner [19] results in a fragmental path, which needs to be smoothed to make the car follow the path. Smoothing of a path is presented in the literature [24], [25]. In this paper, we do not present the smoothed path, since smoothing of the RRT path is not within the scope of the paper. We only present the fragmental path which is generated by the RRT planner in [19].

The RRT planner in [19] stops when the number of randomly generated nodes reaches 50. If a path is not generated until 50 nodes are deployed, then we reset the path planner. The neighbor of a node, say n, is a node whose distance from n is less than 3 meters.

In the RRT planner, nodes are randomly deployed in the workspace to generate a path to the goal. For fair comparison with our approach, we performed 20 Monte-Carlo simulations and calculate the average path length and its variance. Also, we derive average computation time for path planning methods.

In simulation figures, obstacles are illustrated with rectangles. The red curve indicates the path generated utilizing our algorithms. Also, the trajectory of the car is depicted with green circles. The boundary of the car is illustrated with a blue box in simulation figures.

We first simulate the case where $InitP$ is (-5,5), as depicted in Fig. 7. In this case, perpendicular parking utilizing $S^+R^-S^-$ does not result in collision. See that our approach can handle the case where a car parks in a small parking space.

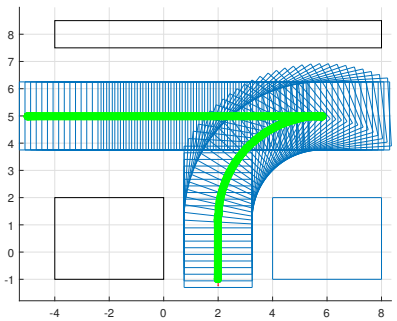


FIGURE 7. We simulate the case where $InitP$ is (-5,5). In this case, perpendicular parking utilizing $S^+R^-S^-$ does not result in collision. The average computational time to simulate one Monte-Carlo in Fig. 7 is 0.08 seconds using MATLAB. The average path length is 18 meters and its variance is zero.

The average computational time to simulate one Monte-Carlo in Fig. 7 is 0.08 seconds using MATLAB. Moreover, it is safe to assume that the use of a compiled language (such as C++) would lead to faster computations than the interpreted MATLAB language. The average path length is 18 meters and its variance is zero, since there is no random generation in our method.

Speed and steering controls associated to Fig. 7 are presented in Fig. 8. Whenever the car switches from one segment to another, it decreases its speed to zero. While the car's speed is zero, it changes its steering angle.

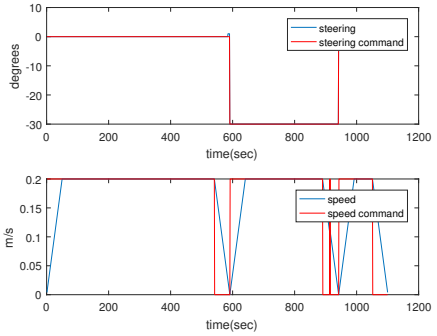


FIGURE 8. Speed and steering controls associated to Fig. 7.

We next test the RRT planner in [19] using the environment in Fig. 7. Fig. 9 shows the generated path using one Monte-Carlo simulation. The average computational time to simulate one Monte-Carlo is 87 seconds using MATLAB. The average path length is 22 meters and its variance is 22, since nodes are randomly generated in RRT methods. See that the proposed method outperforms the RRT planner in [19]. The proposed method runs much faster than the RRT planner in [19].

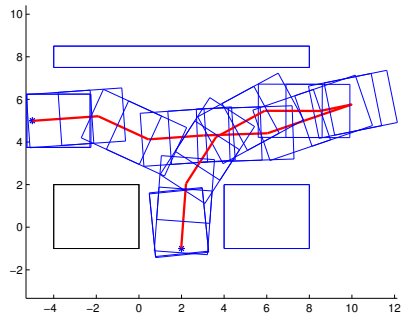


FIGURE 9. We test the RRT planner in [19] using the environment in Fig. 7. The average path length is 22 meters and its variance is 22, since nodes are randomly generated in RRT methods. The average computational time to simulate one Monte-Carlo is 87 seconds using MATLAB.

We next simulate the case where $InitP$ is (-5,3.5), as depicted in Fig. 10. In this case, perpendicular parking utilizing $S^+R^-S^-$ leads to collision. Thus, a safe path is found by decreasing $A(InitP, InterP, InfP)$ gradually. A safe path is found at the first iteration, i.e., when

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2971250, IEEE Access

$A(InitP, InterP, InfP) = 170$ degrees. The average computational time to simulate one Monte-Carlo in Fig. 10 is 0.1 seconds using MATLAB. The average path length is 16 meters and its variance is zero, since there is no random generation in our method.

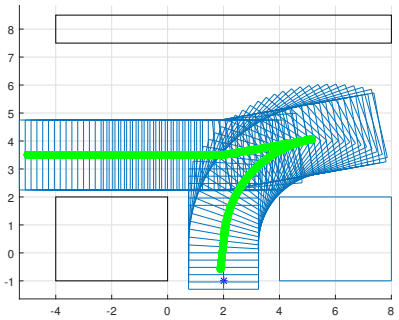


FIGURE 10. We simulate the case where $InitP$ is (-5,3.5). A safe path is found at the first iteration, i.e., when $A(InitP, InterP, InfP) = 170$ degrees. The average computational time to simulate one Monte-Carlo in Fig. 10 is 0.1 seconds using MATLAB. The average path length is 16 meters and its variance is zero.

Speed and steering controls associated to Fig. 10 are presented in Fig. 11. Whenever the car switches from one segment to another, it decreases its speed to zero. While the car's speed is zero, it changes its steering angle.

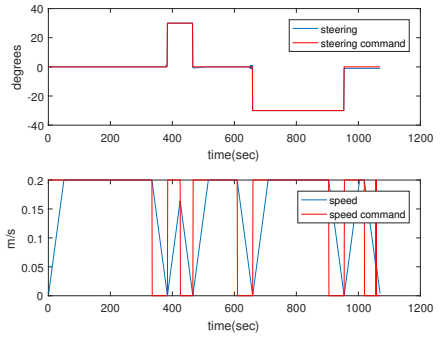


FIGURE 11. Speed and steering controls associated to Fig. 10.

We next test the RRT planner in [19] using the environment in Fig. 10. Fig. 12 shows the generated path using one Monte-Carlo simulation. The average computational time to simulate one Monte-Carlo is 60 seconds using MATLAB. The average path length is 19 meters and its variance is 14, since nodes are deployed randomly in RRT methods. The proposed method runs much faster than the RRT planner in [19]. See that the proposed method outperforms the RRT planner in [19].

We set up another obstacle environment. In simulation figures (Fig. 13), two obstacles on both sides of the parking space are illustrated with two polygons. We simulate the case where $InitP$ is (-5,3.5). In this scenario, perpendicular parking utilizing $S^+R^-S^-$ leads to collision. Thus, we decrease $A(InitP, InterP, InfP)$ gradually until it becomes

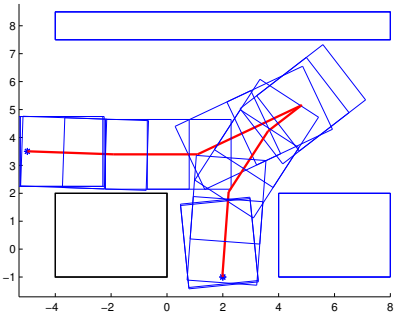


FIGURE 12. We test the RRT planner in [19] using the environment in Fig. 10. The average computational time to simulate one Monte-Carlo is 60 seconds using MATLAB. The average path length is 19 meters and its variance is 14, since nodes are deployed randomly in RRT methods.

100 degrees. This implies that a safe path is found after 8 iterations. See that the car moves through a very narrow passage between two obstacles. The average computational time to simulate one Monte-Carlo in Fig. 13 is 0.2 seconds using MATLAB. The average path length is 16 meters and its variance is zero, since there is no random generation in our method.

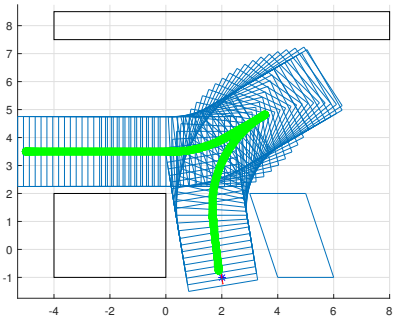


FIGURE 13. A safe path is found after 8 iterations. See that the car moves through a very narrow passage between two obstacles. The average computational time to simulate one Monte-Carlo in Fig. 13 is 0.2 seconds using MATLAB. The average path length is 16 meters and its variance is zero.

Speed and steering controls associated to Fig. 13 are presented in Fig. 14. Whenever the car switches from one segment to another, it decreases its speed to zero. While the car's speed is zero, it changes its steering angle.

We next test the RRT planner in [19] using the environment in Fig. 13. The RRT planner fails to find a safe path to the parking space due to irregular obstacles in the environment.

VI. CONCLUSIONS

This paper introduces the perpendicular parking algorithm of car-like robots, such that the generated path consists of a cusp. This path planning is based on the car's turning radii, which can be determined by the car's geometry and its maximum steering angle.

We compare the proposed approach with [19] to verify the effectiveness of our approach. We acknowledge that RRT

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2971250, IEEE Access

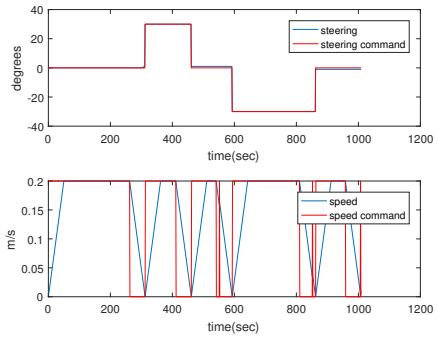


FIGURE 14. Speed and steering controls associated to Fig. 13.

methods in [19] have the advantage of providing general solutions, while the proposed method only works in perpendicular parking scenarios. As our future works, we will verify the effectiveness of our approach using a real car-like robot.

As far as we know, this paper is novel in developing autonomous perpendicular parking based on circular arc and straight line segments, such that a cusp on the generated path is allowed. Since a cusp is allowed, the proposed parking approach is suitable for parking in a small space.

In the case where there are many obstacles close to a parking space, it may be necessary to generate a path consisting of multiple cusps. As our future works, we will develop an algorithm to generate a path with multiple cusps.

ACKNOWLEDGMENT

This work was supported by the National Research Foundation (NRF) of Korea grant funded by the Korea government (MSIT) (No. 2019R1F1A1057282)

REFERENCES

[1] K. Min, J.Choi, H. Kim, and H. Myung, "Design and implementation of path generation algorithm for controlling autonomous driving and parking," in 2012 12th International Conference on Control, Automation and Systems, Korea, Oct 2012, pp. 956–959.

[2] M. H. Li and P. K. Tseng, "Implementation of an autonomous driving system for parallel and perpendicular parking," in 2016 IEEE/SICE International Symposium on System Integration (SII), Dec 2016, pp. 198–203.

[3] D. Pérez-Morales, O. Kermorgant, S. Domínguez-Quijada, and P. Martinet, "Autonomous Perpendicular And Parallel Parking Using Multi-Sensor Based Control," in IEEE/RSJ International Conference on Intelligent Robots and Systems, Vancouver, Canada, Sep. 2017. [Online]. Available: https://hal.inria.fr/hal-01689850

[4] L. Han, Q. H. Do, and S. Mita, "Unified path planner for parking an autonomous vehicle based on rrt," in 2011 IEEE International Conference on Robotics and Automation, May 2011, pp. 5622–5627.

[5] W. Wang, Y. Song, J. Zhang, and H. Deng, "Automatic parking of vehicles: A review of literatures," International Journal of Automotive Technology, vol. 15(6), pp. 967 – 978, 2014.

[6] B. Li, K. Wang, and Z. Shao, "Time-optimal maneuver planning in automatic parallel parking using a simultaneous dynamic optimization approach," IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 11, pp. 3263–3274, Nov 2016.

[7] J. Moon, I. Bae, and S. Kim, "Real-time near-optimal path and maneuver planning in automatic parking using a simultaneous dynamic optimization approach," in 2017 IEEE Intelligent Vehicles Symposium (IV), USA, June 2017, pp. 193–196.

[8] H. Marzbani, H. Khayyam, C. N. TO, Ä. V. Quoc, and R. N. Jazar, "Autonomous vehicles: Autodriver algorithm and vehicle dynamics," IEEE Transactions on Vehicular Technology, vol. 68, no. 4, pp. 3201–3211, April 2019.

[9] Y. Rasekhipour, A. Khajepour, S. Chen, and B. Litkouhi, "A potential field-based model predictive path-planning controller for autonomous road vehicles," IEEE Transactions on Intelligent Transportation Systems, vol. 18, no. 5, pp. 1255–1267, May 2017.

[10] J. Kim, "Boundary tracking control for autonomous vehicles with rigidly mounted range sensors," Journal of Intelligent and Robotic Systems, vol. 95(3-4), p. 1041âĂŞ1048, 2019.

[11] ——, "Control laws to avoid collision with three dimensional obstacles using sensors," Ocean Engineering, vol. 172, pp. 342–349, 2019.

[12] G. Cabodi, P. Camurati, A. Garbo, M. Giorelli, S. Quer, and F. Savarese, "A smart many-core implementation of a motion planning framework along a reference path for autonomous cars," Electronics, vol. 8(2), no. 177, pp. 1–28, 2019.

[13] S. Choi, C. Boussard, and B. d'Andrea Novel, "Easy path planning and robust control for automatic parallel parking," IFAC Proceedings Volumes, vol. 44(1), pp. 656 – 661, 2011.

[14] H. Vorobieva, S. Glaser, N. Minoiu-Enache, and S. Mammar, "Automatic parallel parking in tiny spots: Path planning and control," IEEE Transactions on Intelligent Transportation Systems, vol. 16, no. 1, pp. 396–410, Feb 2015.

[15] H. Vorobieva, N. Minoiu-Enache, S. Glaser, and S. Mammar, "Geometric continuous-curvature path planning for automatic parallel parking," in 2013 10th IEEE INTERNATIONAL CONFERENCE ON NETWORKING, SENSING AND CONTROL (ICNSC), April 2013, pp. 418–423.

[16] H. Vorobieva, S. Glaser, N. Minoiu-Enache, and S. Mammar, "Geometric path planning for automatic parallel parking in tiny spots," IFAC Proceedings Volumes, vol. 45, no. 24, pp. 36–42, 2012.

[17] A. Gupta, R. Divekar, and M. Agrawal, "Autonomous parallel parking system for ackerman steering four wheelers," in 2010 IEEE International Conference on Computational Intelligence and Computing Research, Dec 2010, pp. 1–6.

[18] P. Petrov, F. Nashashibi, and M. Marouf, "Path planning and steering control for an automatic perpendicular parking assist system," in 7th Workshop on Planning, Perception and Navigation for Intelligent Vehicles, Germany, Oct 2015, pp. 143 – 148.

[19] Y. Wang, D. K. Jha, and Y. Akemi, "A two-stage rrt path planner for automated parking," in 2017 13th IEEE Conference on Automation Science and Engineering (CASE), China, Aug. 2017.

[20] P. Petrov, F. Nashashibi, and M. Marouf, "Path planning and steering control for an automatic perpendicular parking assist system," in 7th Workshop on Planning, Perception and Navigation for Intelligent Vehicles (PPNIV), vol. 15, Germany, 2015, pp. 1–6.

[21] L. E. Dubins, "On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents," Amer. J. Math., vol. 145(2), pp. 497 – 516, 1957.

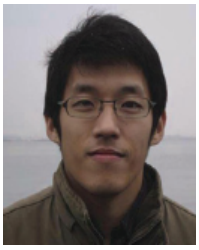
[22] D. Perez-Morales, S.Dominguez-Quijada, O.Kermorgant, and P. Martinet, "Autonomous parking using a sensor based approach," in 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), Brazil, Nov 2016, pp. 211–216.

[23] J. A. Reeds and L. A. Shepp, "Optimal paths for a car that goes both forwards and backwards," PACIFIC JOURNAL OF MATHEMATICS, vol. 145(2), pp. 367 – 393, 1990.

[24] D. P. Bertsekas, Dynamic Programming and Optimal Control. USA: Athena Scientific Belmont, 1995.

[25] L. E. Kavraki, P. Svestka, J. . Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," IEEE Transactions on Robotics and Automation, vol. 12, no. 4, pp. 566–580, Aug 1996.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2971250, IEEE Access



JONGHOEK KIM is an Assistant Professor in the Department of Electrical and Computer Engineering at Hongik University, Republic of Korea. His research is on target tracking, control theory, robotics, multi-agent systems, and optimal estimation. He worked as a senior researcher at Agency for Defense Development in Republic of Korea from 2011 to 2018. In 2011, he earned a Ph.D. degree co-advised by Dr. Fumin Zhang and Dr. Magnus Egerstedt at Georgia Institute of Technology, USA. Jonghoek Kim received his M.S. in Electrical and Computer Engineering from Georgia Institute of Technology in 2008 and his B.S. in Electrical and Computer Engineering from Yonsei University, Republic of Korea in 2006.