International Journal of Automotive Technology, Vol. 22, No. 1, pp. 119–129 (2021)
DOI 10.1007/s12239–021–0013–7

pISSN 1229–9138/ eISSN 1976–3832

ACCURATE PATH TRACKING BY ADJUSTING LOOK-AHEAD POINT IN PURE PURSUIT METHOD

Joonwoo Ahn[1)], Seho Shin[2)], Minsung Kim[1)] and Jaeheung Park[1, 3)*]

[1)]Graduate School of Convergence Science and Technology, Seoul National University, Seoul 08826, Korea
[2)]System LSI, Samsung Electronics Co., Ltd., 1 Samseongjeonja-ro, Hweaseong 18448, Korea
[3)]Advanced Institute of Convergence Technology, 145 Gwanggyo-ro, Suwon 16229, Korea

(Received 8 January 2020; Revised 21 April 2020; Accepted 6 May 2020)

ABSTRACT–Path tracking is an essential aspect of the navigational process of self-driving cars. Especially, accurate path tracking is important for not only normal urban roads but also narrow and complex roads such as parking lot and alleyway. The pure pursuit method is one of the geometric path-tracking methods. Using this method, the look-ahead point can be selected far away and the control input is computed in real-time, which is advantageous when the given path is not smooth or when the path is specified using waypoints. Moreover, this method is more robust to localization errors than the model-based path-tracking method. However, the original pure pursuit method and its variants have limited tracking performance. Therefore, this paper proposes a new method that heuristically selects a look-ahead point by considering the relationship between a vehicle and a path. Using this new look-ahead point, the vehicle can stably converge to the desired path and track the path without encountering the cutting-corner problem. The proposed method was tested using simulation and our self-driving car platform. Our results show that the vehicle tracks the desired path more accurately using our proposed algorithm than using the previous pure pursuit methods.

KEY WORDS : Path tracking, Geometric path-tracking method, Accurate path tracking, Pure pursuit method, Look-ahead point selection

1. INTRODUCTION

Research on autonomous vehicles has progressed substantially in recent years (Thrun *et al.,* 2006; Urmson *et al.,* 2008). An important factor for realizing driving is path tracking capability. The path-tracking method computes the control command that enables a vehicle to follow a planned path. The model-based method and the geometric method are widely used for tracking the paths of vehicles (Paden *et al.,* 2016).

Model-based path-tracking methods use the kinematic or dynamic characteristics of vehicles. These methods consider the kinematic or dynamic effects by which it is possible to predict and compensate for errors that may occur between the vehicle movement and the path (Rajamani, 2011). The output feedback linearization method uses the internal structure of the vehicle's kinematic model for high vehicle speeds (Kanayama, 1990). The linear–quadratic regulator (LQR) method is applied using the lateral dynamics of a vehicle (d'Andrea-Novel *et al.*, 1995). Unknown disturbances (the nonlinear uncertainties of rolling friction and vehicle occupants changing) are additionally considered using a parameter self-tuning LQR method (Piao *et al*., 2019).

A feed-forward term can be added to the LQR to handle a steady-state error (Rajamani, 2011). For accurate, and smooth path tracking, optimal preview control takes into account not only tracking errors but also road curvatures ahead of a vehicle (Peng *et al.*, 2020). The performance of the model predictive control (MPC) method is similar to that of the methods mentioned above, and it also satisfies the constraints of the vehicle (Kim *et al.,* 2014). Additionally, the nonlinearities, parametric uncertainties, time-varying, external disturbance are considered to improve the performance (Peng *et al.,* 2020). However, model-based path-tracking methods have the following problems (Bu *et al.,* 2013); i) They tend to be unstable because they are dependent on the localization error and non-smooth paths. ii) They require a long computation time to find a solution.

This paper considers urban driving environments such as passing through a narrow cluttered space, turning at a high curvature corner, parking the vehicle, avoiding an obstacle, and changing lanes. In these environments, the driving speed is relatively low, and the path curvature is relatively high. Also, the path can be non-smooth in the form of waypoints and the localization error can also be high. In

*Corresponding author. e-mail: park73@snu.ac.kr

these cases, a geometric path-tracking method is generally preferred over the model-based path-tracking methods (Bu *et al.,* 2013). Therefore, this paper proposes an accurate path-tracking method based on one of the geometric methods called the pure pursuit method.

Geometric path-tracking methods use a geometric relationship between a vehicle and a path; there are vector pursuit, pure pursuit, and Stanley methods. These methods are less susceptible to path smoothness, scarcity of waypoints, and localization errors. The Stanley method calculates the steering angle to reduce the lateral distance and the angular difference in the direction between the front wheels and the nearest point on the path. This method is suitable for high-speed driving and tracks a corner well. However, the Stanley method was developed only for forward driving; it tends to become unstable for discontinuous curvatures (Thrun *et al.*, 2006). The vector pursuit method uses the screw theory that moves the vehicle in the desired direction (Wit, 2000). This method is sensitive to the selection of the look-ahead point and the curvature change of the path.

Among the geometric path-tracking methods, the pure pursuit method is used. This method calculates the steering angle using a look-ahead point (Coulter, 1992). The look-ahead point is located at a certain distance from the current position of the vehicle on the path. If this point is well chosen, it is more robust to the non-smooth path and the large cross-track error as compared with the Stanley or vector pursuit methods. Also, this method is simple to implement; therefore, it is the most popular method.

The pure pursuit method has two problems depending on the location of the look-ahead point. First, there is a trade-off relationship between the stability and tracking performance. If the look-ahead point is set close to the vehicle, the tracking stability is not good because there would be a large heading error. Conversely, if this point is set far from the vehicle, the tracking becomes slow. The trade-off problem is addressed by using an adaptive strategy to adjust the look-ahead distance (Qinpeng *et al.,* 2019). The vehicle can converge on a path without the heading oscillation, but not as quickly as possible. Second, the cutting-corner problem; a cross-track error occurs where the curvature varies because the curvature of the look-ahead point is different from the curvature of the path near the vehicle. To address the cutting-corner problem, the look-ahead point is selected around the path with a constant offset distance (Andersen *et al.,* 2016). Depending on the offset distance setting, this problem can be reduced at a certain curvature, but there are errors at other curvatures.

This study proposes a heuristic method to solve the above problems. The look-ahead point is selected by considering the geometric relationship between a vehicle and a path. This consists of two steps. In the first step, a look-ahead point is selected on a path by taking into account the position and the direction of the vehicle and the path. In the second step, the look-ahead point, which is obtained from the first step, is moved and located outside the path to address the cutting-corner problem. This step is particularly effective when the vehicle is near the path and when the curvature abruptly increases, for example, when entering a corner. the effectiveness of the proposed method is verified in the simulation and by using a vehicle test. Our experimental results prove that the proposed method can adjust the position and the orientation of the vehicle with less oscillation and less cutting-corner phenomena than the conventional pure pursuit methods.

This paper is organized as follows. In Section 2, the pure pursuit method is introduced, and its limitations are described. Section 3 introduces a method used to select the look-ahead point. Section 4 describes the effectiveness of the proposed method from the results of the simulations and the vehicle-driving experiments. Conclusion is presented in Section 5.

2. PURE PURSUIT PATH-TRACKING METHOD

In this section, the pure pursuit path-tracking method is described in detail. Then, other methods of selecting the look-ahead point and their limitations are introduced.

2.1. Pure Pursuit Method

The bicycle model (Coulter, 1992) simplified form of the four-wheel Ackerman steering kinematics is used to derive a steering angle for the pure pursuit method:

$$\begin{aligned} \dot{x}_{p_v} &= v\cos(\theta_{car}) \\ \dot{y}_{p_v} &= v\sin(\theta_{car}) \\ \dot{\theta}_{car} &= \frac{v\tan(\delta)}{L} \end{aligned} \tag{1}$$

where p_v is the center of the rear wheel of the vehicle position and x_{p_v} and y_{p_v} are the Cartesian coordinates of the vehicle in the global frame; θ_{car} is the vehicle's orientation in the global frame; δ is the steering angle in the vehicle frame; v is the longitudinal velocity; and L is the wheelbase.

The pure pursuit method tracks the path p_{ref} by calculating the look-ahead point p_l on the p_{ref}. There are two ways to calculate p_l. First, a point where p_{ref} meets a circle having a radius of a look-ahead distance around p_v becomes p_l. As shown in Figure 1(a), L_f can be become the look-ahead distance. However, if the path is highly curved with very high curvatures between the look-ahead point and the vehicle, there may be several look-ahead points to select. Or, if the distance between the vehicle and the path is large and the look-ahead distance is small, there is no look-ahead point to select.

The second way (Campbell, 2007) to deal with these problems is as follows; It is assumed that p_{ref} consists of a set of waypoints. First, a waypoint with the closest distance between p_v and p_{ref} is selected, which is represented by p_w in Figure 1 (a). From p_w, waypoints are accumulated in the direction of the path and a distance is calculated. A waypoint where the accumulated distance equals to the look-ahead distance becomes p_l. Thus, only one p_l can be selected regardless of the path curvature or the look-ahead distance.

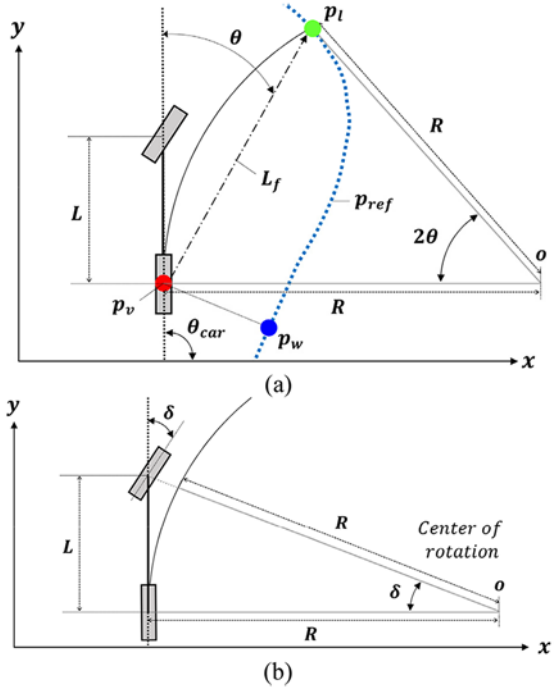

Figure 1. (a) Variables of the path-tracking method and (b) Ackermann steering geometry.

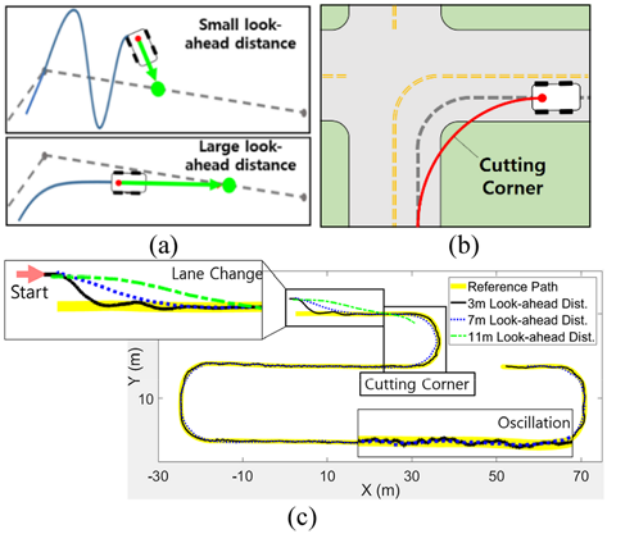

Figure 2. Limitations of the pure pursuit method: (a) Instability of path tracking with a short look-ahead distance and poor tracking performance when a large look-ahead distance is selected. (b) The cutting corner problem is a phenomenon in which the vehicle moves inside the corner instead of around it. (c) Analysis of the pure pursuit method according to fixed speeds in real vehicle experiments.

Therefore, this paper selects p_l similarly to this approach. The specific method is described in Section 3.1.

To reach look-ahead point p_l, the steering angle (δ) is calculated using the curvature of a circular arc R connecting the vehicle (p_v) and the position of along the path p_{ref} (Kuwata *et al.,* 2008). L_f is the distance between the position of the vehicle (p_v) and the look-ahead point (p_l) for forward driving. These are shown in Figure 1 (a). R is obtained geometrically using the sine law and the curvature k of the arc is defined as follows:

$$\frac{L_f}{\sin(2\theta)} = \frac{R}{\sin\left(\frac{\pi}{2}-\theta\right)}$$

$$\frac{L_f}{\sin(\theta)} = 2R \tag{2}$$

$$k = \frac{2\sin(\theta)}{L_f}$$

where θ is a look-ahead heading which is the difference between the heading of the vehicle and that of the vector from p_v to p_l.

When the vehicle turns, the front and the rear wheels move along the circular arc R with the center of rotation o, as proposed by the Ackermann steering geometry. As shown in Figure 1 (b), there exists a geometric relationship between the vehicle steering angle (δ) and R; this relationship can be approximated as $\tan(\delta) = L/R$. R is equal to $1/k$, where k is the curvature. δ can be computed using the kinematic bicycle model derived in equation (1) and k:

$$\delta = tan^{-1}(kL)$$

$$\delta = tan^{-1}\left(\frac{2Lsin\theta}{L_f}\right) \tag{3}$$

2.2. Limitations of Pure Pursuit Method

The pure pursuit method has two major limitations when calculating the steering angle by selecting the look-ahead point. The tracking performance depends on the position where this point is selected on the path (see Figure 2).

2.2.1. Trade-off relationship between tracking performance and stability

When the path is re-planned and changed, such as when there is a lane change or an obstacle avoidance motion, the tracking performance may be poor or unstable depending on the location of the look-ahead point. If the distance difference between the following path and the re-planned path is large, the path planning part needs to deal with this problem. If the distance is not too large, the path tracking part also needs to deal with this problem briefly.

If the vehicle is outside the path and the look-ahead point is selected near the vehicle, the difference between the vehicle

heading and the path direction changes significantly. When the vehicle reaches the path, this difference would still exist, and the vehicle would go out of the path (see Figure 2 (a) upper graph). As shown in Figure 2 (c) graph for Lane Change with 3 m Look-ahead distance, the starting position of the vehicle is 3 m from the positive Y-axis of the reference path, and its heading is the same as the reference path. The target velocity is 3.36 m/s. At 3 m, oscillation appears for the look-ahead distance case. This is because the pure pursuit method considers only the positional relationship between the vehicle and the path; it does not consider the orientation between the vehicle and the path. As shown in Figure 2 (c) Oscillation, when the look-ahead distance is long with high speeds, the heading of the vehicle still changes.

However, if the look-ahead point is selected too far, the vehicle does not quickly converge to the path. This is due to the lack of consideration of the distance from the vehicle to the path. This is shown in the bottom graph of Figures 2 (a) and (c) in the lane changes when the look-ahead distances are 7 and 11 m. When the starting position is set to 7 m, the cutting-corner problem occurs. If it is set to 11 m, the vehicle slowly converges to the path, and a large cutting-corner problem is observed.

2.2.2. Cutting-corner problem

Figure 2 (b) is a schematization of the cutting-corner problem. The cutting-corner problem occurs when the vehicle moves inward into the corner, such as when entering an intersection. This is because there is a curvature difference between the look-ahead point and the current location of the vehicle, especially when the curvature of the look-ahead point is larger than the curvature near the vehicle. Also, the difference between the radius of the arc generated from the vehicle to the look-ahead point and the radius of the arc of the path of the vehicle becomes large. This causes the cross-track error. Even if a look-ahead point is selected near the vehicle, that is the look-ahead distance is short, the cross-track error occurs slightly. This is shown in Figure 2 (c) for the cutting corner with 7 and 11 m look-ahead distances. In the case of the 11 m Look-ahead distance, the cutting corner problem is large, and the vehicle leaves the path. This happens even when the look-ahead distance is 3 m.

Conversely, when the vehicle exits the corner, the cutting-corner problem rarely occurs. At this time, a difference appears between the curvature of the look-ahead point and the curvature of the corner. However, the radius of the arc generated from the vehicle to the look-ahead point is very close to the radius of the arc of the corner where the vehicle is driving. Therefore, there is a very little cross-track error.

2.3. Methods Used to Select Look-ahead Point in Pure Pursuit Method

Studies have been conducted on how to determine a look-ahead point to address the mentioned problems (Hingwe and Tomizuka, 1998; Chen and Tan, 1999; Campbell, 2007). Typically, there are two methods.

2.3.1. Look-ahead point proportional to velocity

The look-ahead distance is determined to be proportional to the velocity of the vehicle (Kuwata *et al.,* 2008). The vehicle tracks the path stably at various velocities. However, it is difficult to converge to the path smoothly when there is a large cross-track error. This is because this way does not take into account the alignment of the direction of the path and the heading of the vehicle at the same time.

2.3.2. Look-ahead point with offset distance

To reduce the cutting-corner problem, a method for assigning an offset distance to the look-ahead point has been proposed (Andersen *et al.*, 2016). This method considers the heading and the location of the look-ahead point. However, it is not easy to determine the optimal offset distance for various situations, and the cutting-corner problem still exists.

Even with the above methods, there are limitations depending on the location of the look-ahead point. To overcome the limitations and track the path more accurately, a method to select the look-ahead point by considering the geometric relationship between the vehicle and the path is introduced in Section 3.

3. METHOD USED TO SELECT POSITION OF LOOK-AHEAD POINT USING GEOMETRIC RELATIONSHIP BETWEEN VEHICLE AND PATH

In this section, a method used to select the look-ahead point by considering the geometric relationship between a vehicle and a path is proposed. This method has two steps. *Step 1* selects a look-ahead point p_d within the path. In this step, the vehicle can converge on the path quickly without any oscillations. *Step 2* calculates a look-ahead point p_l to reduce the cutting-corner problem. When the vehicle is close to the path and is entering the corner while driving on a straight path, the look-ahead point (p_d) is selected outside the path and is represented by p_l. Each step will be described in detail in the following sections.

3.1. *Step 1*: Selection of Look-ahead Point (p_d) in Path

This section introduces a method used to select a look-ahead point p_d within a path. If the distance between the vehicle and the reference path is large, the path planning method is needed to reach the reference path by generating a local path. If this distance is not too large (less than about 5.8 m 2 lanes wide), this situation can be handled simply by the proposed method. The previous pure pursuit had convergence problems to the path if the selected look-ahead point was too close or too far. A detailed analysis is explained in Section 2.2.1.

To address these problems, the Dubins path algorithm is considered. This algorithm is used to determine p_d for fast and monotonic convergence to the reference path. The point where the Dubins path (red dash line in Figure 3) meets the reference path (blue dash line in Figure 3) is designated as p_d. The Dubins path is not used as a path to track but used as a measure to compute the look-ahead point p_d on the path. It is

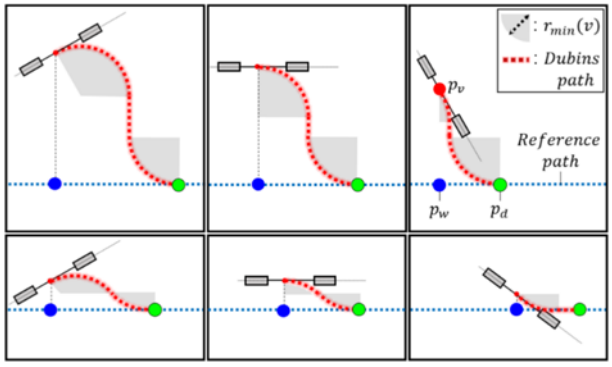

Figure 3. The look-ahead point p_d is determined within the path according to the position and orientation relationship between the vehicle and the path. The vehicle can approach the path in the shortest distance without a heading error; p_v is the vehicle position, and p_w is the nearest point between the vehicle and the path.

assumed to be a good option for the pure pursuit method because the Dubins path not only satisfies the vehicle's non-holonomic constraint but also generates the shortest distance path (Dubins, 1957). Many experimental trials supported our assumptions, and more interestingly, the trajectory generated by tracking p_d was similar to the Dubins path.

The Dubins path is generated by combining the circular arcs having the maximum curvature with the straight lines obtained by joining the start and the goal pose. This path is of six types: RSR, RSL, LSR, LSL, RLR and LRL (*turn right* (R), *turn left* (L) or *go straight* (S)). For example, for the vehicle position (p_v) and the goal position (p_d), as shown in Figure 3, the optimal path is shown as an RSL type. Then, p_v and p_d are connected to the desired direction at each end, and the shortest curve that does not exceed the minimum turning radius of the vehicle is formed.

The minimum turning radius of the circular arc is required to create a Dubins path. This arc is represented as the shaded area in Figure 3. This radius changes according to a velocity command v_{cmd}, and the distance from p_v to p_l, becomes proportional to the velocity:

$$r_{min}(v_{cmd}) = \frac{v_{cmd}^2}{9.79(e_{max}+f_{max})} \quad (4)$$

This is from the relationship of the velocity radius curve defined by the American Association of State Highway and Transportation Officials (AASHTO) (Kilinc and Baybura, 2012). the unit of $r_{min}(v_{cmd})$ is meter, and v_{cmd} is in m/s; e_{max} is the cross slope within the horizontal curve, and f_{max} is a frictional coefficient of the road surface. This paper assumes that there is no slope on the road (e_{max}: 0), and f_{max} is set to 0.05. The value of $r_{min}(v_{cmd})$ is specified to range from a minimum of the minimum turning radius to a maximum of 12 m.

The pseudo-code for selecting p_d by using the Dubins path

Algorithm 1: Selection of Look-ahead Point p_d

```
1  Input: p_ref, p_v, r_min(v)
2  Output: p_d
3  for i = 0 to p_ref(i) do
4  |   p_w ← argmin_{p_ref(i)} ||p_ref(i) − p_v||
5  |   i ← i + 1
6  end
7  i_{p_w} = index of p_w
8  for i = i_{p_w} to p_ref(i) do
9  |   if r_min(v_cmd) ≤ ||p_w − p_ref(i)|| then
10 |   |   len_dubin ← DubinsPathLength(p_v, r_min(v_cmd), p_ref(i))
11 |   |   p_d ← argmin_{p_ref(i)}(len_dubin)
12 |   end
13 |   i ← i + 1
14 end
```

is shown in Algorithm 1. This algorithm has two main phases: i) Finding the nearest way-point p_w (lines 3 to 6) and ii) Selecting the look-ahead point p_d (lines 8 to 14). In the first phase, among the waypoints of $p_{ref}(i)$, the closest distance to p_v is searched and designated as p_w. This is shown as a blue filled circle in Figure 3. The path of $p_{ref}(i)$ points is a set of way-points and i is an index of the way-point. This is represented as $[(x_{p_{ref}}(0), y_{p_{ref}}(0)), \ldots (x_{p_{ref}}(i), y_{p_{ref}}(i)), \ldots (x_{p_{ref}}(n), y_{p_{ref}}(n))]$.

In the second phase, p_d is selected using the Dubins path algorithm (lines 8 to 14). To use this algorithm, the start and goal pose have to be determined. The start pose is represented as $p_v(\delta)$. Its position is set to the vehicle position, and its direction is the sum of the vehicle heading (θ_{car}) and the current steering angle (δ). The candidate goal points i are assigned in the way-point set ($p_{ref}(i)$), which is shown in line 8. The distance between the candidate goal points and p_w is larger than $r_{min}(v_{cmd})$ (line 9). This is to prevent p_d from being too close to the vehicle (sixth picture (bottom right) in Figure 3). Besides, this distance is shorter than 12 m to limit the search range. Among these candidate goal points i, the one generated with the shortest length by the Dubins path (len_{dubin}) is designated as p_d (i.e., the green filled circle in Figure 3; see lines 10 and 11). In the line 10, the *DubinsPathLength* function returns a length of the Dubins path with $r_{min}(v_{cmd})$, and it is represented by len_{dubin}. The nearest point that satisfies len_{dubin} in the path ($p_{ref}(i)$) is designated as p_d in line 11.

3.2. *Step 2*: Setting look-ahead point (p_l) out of path

This section introduces a method to used set the look-ahead point (p_d) outside the path to reduce the cutting-corner problem. As shown in Figure 2 (b) and Figure 4, if the vehicle tracks the look-ahead point p_d, the direction of the front wheel is toward the heading of $\vec{p}_{wd}$, which is a vector from p_w to p_d. Therefore, the vehicle deviates from $\vec{p}_{wl}$, which is a vector tangent to p_w. To address this problem, the vehicle tracks p_l that is located in the direction of $\vec{p}_{wl}$ rather than $\vec{p}_{wd}$. As shown in Figure 4, p_l is shifted toward $\vec{p}_{wl}$ from p_d by $\vec{p}_{dl}$. The degree that p_l is shifted along $\vec{p}_{dl}$ depends on τ. The formula for obtaining p_l is as follows:

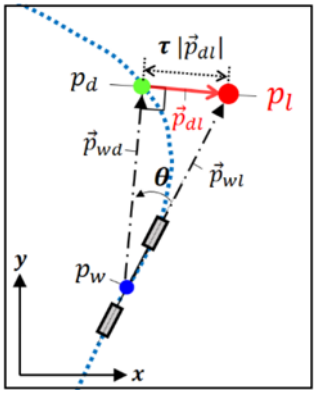

Figure 4. The point p_d is set outside of the path and represented by p_l, which is the final look-ahead point. The vehicle can enter the corner without the cutting-corner problem by p_l.

$$x_{p_l} = x_{p_d} + \tau|\vec{p}_{dl}|\cos(\theta_{\vec{p}_{dl}})$$
$$y_{p_l} = y_{p_d} + \tau|\vec{p}_{dl}|\sin(\theta_{\vec{p}_{dl}}) \quad (5)$$

where (x_{p_l}, y_{p_l}) and (x_{p_d}, y_{p_d}) are the locations of p_l and p_d, respectively and $\vec{p}_{dl}$ is a vector that connects p_d and p_l. The length and the direction of $\vec{p}_{dl}$ are denoted by $|\vec{p}_{dl}|$ and $\theta_{\vec{p}_{dl}}$, respectively; $\theta_{\vec{p}_{dl}}$, $\theta_{\vec{p}_{wl}}$ and $\theta_{\vec{p}_{wd}}$ are angles relative to the global coordinate.

$$|\vec{p}_{dl}| = |\vec{p}_{wd}|tan(|\theta|)$$
$$\theta_{\vec{p}_{dl}} = \theta_{\vec{p}_{wd}} - sign(\theta)\frac{\pi}{2} \quad (6)$$

where $|\vec{p}_{wd}| = ||p_w - p_d||$ is the length of $\vec{p}_{wd}$; θ is the directional difference between $\theta_{\vec{p}_{wl}}$ and $\theta_{\vec{p}_{wd}}$:

$$\theta = \theta_{\vec{p}_{wl}} - \theta_{\vec{p}_{wd}} \quad (7)$$

In the typical situation where the vehicle is tracking the path close, θ rarely exceeds 90 degrees. Therefore, it is assumed that the value of θ is less than 90.

τ represents the degree to which the *Step 2* method is applied. *Step 2* is applied only when the vehicle encounters the cutting-corner problem. This problem occurs when the curvature changes rapidly from a small value to a large value. One example is when the vehicle enters the corner while driving the straight path. When this value is zero, the look-ahead point is set to p_d, or when this value is 1, the look-ahead point is set to p_l. If the vehicle is close to the path and the curvature of p_d is larger than the curvature of p_w, the value of τ increases, and the look-ahead point p_l will be set outside the path.

Therefore, τ is calculated according to the distance (α) and the curvature difference (β):

$$\tau = (1-\alpha)\beta \quad (8)$$

For example, Figures 5 (c) and (d) show that τ is small because β is small regardless of α. In Figure 4, this value becomes large because α is small and β is large.

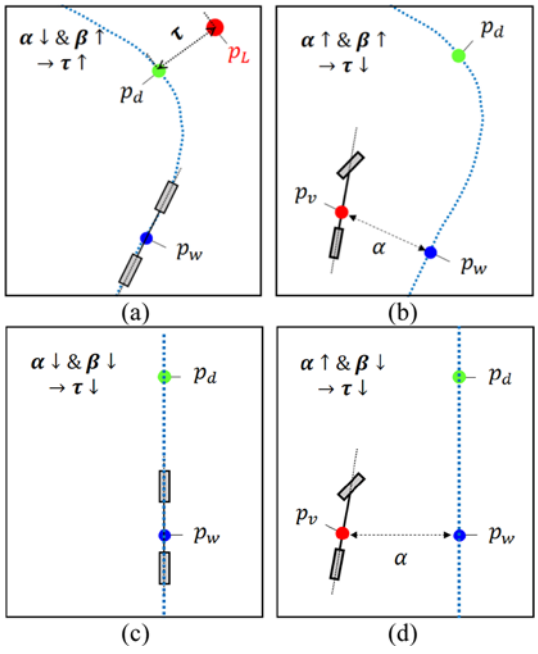

Figure 5. Examples of τ in various situations: (a) the distance between the vehicle and the path is short, and α is low. The curvature difference between p_w and p_d is large, and β is high. According to Equation (8), τ has a large value. (b) The α and β values are high; therefore, τ is low. (c) and (d) The vehicle tracks a straight path; τ is zero because β is zero.

α ($\in$ [0, 1]) is a normalized value of the distance between p_v and p_w:

$$\alpha = \begin{cases} \dfrac{\|p_v - p_w\|}{\alpha_{max}} & \|p_v - p_w\| < \alpha_{max} \\ 1 & \|p_v - p_w\| \geq \alpha_{max} \end{cases} \quad (9)$$

where α_{max} is a constant value. An α value close to 0 indicates that the vehicle is very close to the path. This is shown in Figure 5 (a) and (c). β ($\in$ [0, 1]) is the normalized value of the difference between cur_{p_w} and cur_{p_d}; cur_{p_w} is the curvature of the nearest way-point (p_w) from the vehicle, and cur_{p_d} is the curvature of the path where p_d is located. This is applied only if cur_{p_w} is lower than cur_{p_d}. The reason is that β applies only to the situation where the cutting-corner problem occurs.

$$\beta = \begin{cases} 0 & if\ (cur_{p_w} \geq cur_{p_d}) \\ \dfrac{cur_{p_d} - cur_{p_w}}{\beta_{max}} & else\ if\ (cur_{p_d} - cur_{p_w} < \beta_{max}) \\ 1 & else\ if\ (cur_{p_d} - cur_{p_w} \geq \beta_{max}) \end{cases} \quad (10)$$

where β_{max} is a constant value and lager than 0. When the β

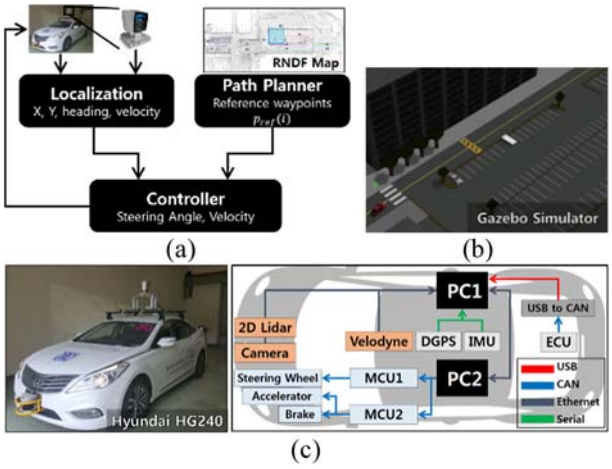

Figure 6. Experimental environments: (a) Autonomous driving system architecture, (b) Simulator environment, (c) Autonomous vehicle system.

value is close to 1, cur_{p_d} is much larger than cur_{p_w}, and the difference between them is close to β_{max}. This is shown in Figures 5 (a) and (b).

4. EXPERIMENT

In this section, an experimental setup is introduced and analyze the effectiveness of our path-tracking method, using the Gazebo simulator and autonomous vehicle-driving experiments.

4.1. Experimental Setup

The architecture of the autonomous driving system is represented in Figure 6 (a). Localization obtains an ego vehicle position and heading angle data. The path planner generates the reference path using the route network definition file (RNDF) file (Challenge, 2007). This path is a set of waypoints that were splined at equidistant intervals of 0.01 m using the CatmullRom spline method. The controller that includes our path-tracking method controls the vehicle by calculating the steering angle and velocity.

Simulator and autonomous vehicle test environments are described in Figures 6 (b) and (c). The simulator Gazebo was used to test algorithms in various situations. The driving environments were created using the Google Sketchup 3D modeling software. Hyundai HG 240 was used for the vehicle experiment. Two computers are used for the robot operating system (ROS) open robot meta-os platform. One computer processed the sensor input data, and the other computer was used for running the path-tracking methods. The localization data were computed by simultaneous localization and mapping based on the 3D LiDAR (Velodyne 64 channels) and inertial measurement unit data using LOAM ROS package (Zhang and Singh, 2014). If the vehicle is driving on the curved path or the look-ahead point exists on the curved path v_{cmd} is computed by

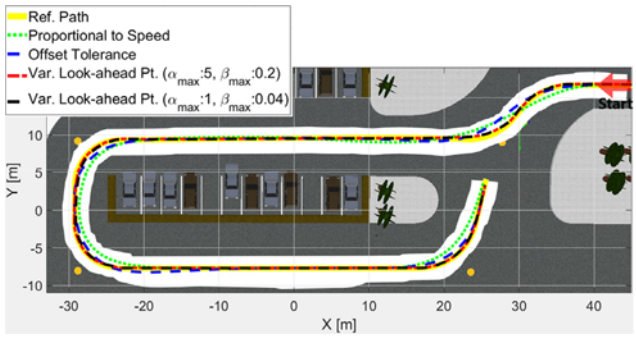

Figure 7. Simulation test results using parameter α_{max} and β_{max}. The maximum curvature of this path was 0.2, and the minimum was 0.04; v_{tar} was set to 3.05 m/s.

Table 1. Simulation Test Results Using Various Parameter (Figure 7)

Total Cross-Track Error		α_{max}		
		1	3	5
β_{max}	0.2	0.115	0.113	***0.109***
	0.06	0.116	0.114	0.111
	0.04	0.117	0.116	0.113

Note. These values are the average value of the cross-track error data. The unit of data is the meter.

the relationship of the velocity radius curve, which is defined by AASHTO (Kilinc and Baybura, 2012). Otherwise, the velocity command v_{cmd} is set to the maximum target velocity. To control v_{cmd}, PI controller is used to calculating the accelerator and brake command.

4.2. Experimental Results

The difference between the pre-planned path and the trajectory along which the vehicle actually moved is shown. Also, the cross-track error was evaluated as a quantitative indicator. Three experiments were conducted. In the first experiment, the results based on the parameters α_{max} and β_{max} are analyzed in the simulator. The second and third experiments show the effectiveness of the *Step 1* and *Step 2* methods in the simulator and the autonomous vehicle.

The following three pure pursuit methods were compared: i) The *`Proportional to Speed method.'*: In this method, the look-ahead distance was chosen proportional to the speed (Kuwata *et al.,* 2008) (Section 2.3.1)). ii) The '*Offset Tolerance method.*': In this method, the look-ahead point is selected considering the heading and the location (offset tolerance parameter *D*: 0.3) (Andersen *et al.*, 2016) (Section 2.3.2)), iii) The '*Var. Look-Ahead Pt method.*' (proposed) method (Section 3).

4.3. Analysis about parameters α_{max} and β_{max}

The effects of the parameters α_{max} and β_{max} were analyzed (see Figure 7). When α is lower than α_{max}, *Step 2* is applied in

inverse proportion to α_{max}. If α_{max} is set to less than 1 m, Step 2 is sensitive to the slight changes in the cross-track error, and the control becomes unstable. The value of α_{max} was set to less than 5 m, which is the maximum distance for which the vehicle could be separated from the path in the parking environment. The application of *Step 2* increases in proportion to β_{max}. The look-ahead point will be selected at the points that deviated from the path, even if the curvature of the curved path is slightly greater than the curvature of the straight path. As a result, the vehicle can cause a low cross-track error while tracking the straight path. However, if β_{max} was set to a very large value, the look-ahead point will be changed too much even for very small curvature changes, which would result in unstable control performance.

Nine parameter sets were tested in Table 1 (α_{max} (1, 3 and 5) and β_{max} (0.2, 0.06 and 0.04)) were tested at the parking lot environment (see Figure 7) and compared them with the results of the '*Proportional to Speed*' and '*Offset Tolerance*' methods in the simulation environment. If the curvature of the curve changes or if v_{tar} is differently chosen in these cases, the result will be different. However, assuming that the simulated environment is a typical road environment, such as a parking lot.

The parameters for the lowest cross-track error were α_{max}: 5 and β_{max}: 0.2, and the error was 0.109. The worst performance was produced by α_{max}: 1 and β_{max}: 0.04, and the error was 0.117. Both parameter sets had similar errors at small curvatures, but at large curvatures, there existed a difference in error. Small curvatures exist near the starting point and the arrival point on the right side of Figure 7, and the large curvatures on the left side. The errors made by the '*Proportional to Speed*' and '*Offset Tolerance*' were 0.380 and 0.237, respectively. Experiments in this paper were conducted with α_{max}: 5 and β_{max}: 0.2. If the path has larger curvatures than the path used in the experiment, it is better to use a larger α_{max} and a lower β_{max}.

4.4. Results for effectiveness of *Step 1* in Section 3

4.4.1. Simulation test results

The vehicle starts in three directions at a distance of 6 m from the path. When tracking was performed at a low speed (1.33 m/s; see in Figure 8 (a)), some overshoots occurred in the results of the other methods. Also in Figure 8 (b) left, when the vehicle approached vertically from outside the path using the other methods, it deviated slightly from the path, because the look-ahead point is selected close to the vehicle using the velocity-proportional look-ahead distance. However, the proposed method selects it farther than the other methods so that the heading can be well adjusted in the path. Also when the vehicle approached the path in parallel, a less cross-track error occurred using the proposed method (see Figure 8 (b) middle). If the direction of the vehicle was opposite to that of the path, the vehicle that used the proposed method approached the path faster than the vehicle that used other methods (see Figure 8 (b) right). This was because the look-ahead point was selected close to the vehicle until it was aligned to the path the

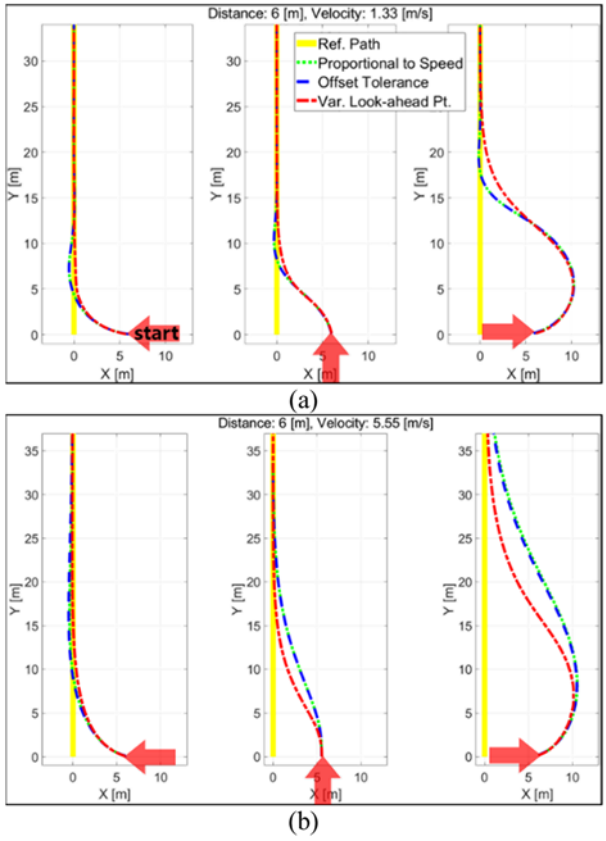

Figure 8. Simulation test results (effectiveness of *Step 1*): Approaching the reference path from three directions at two kinds of velocities. (a) v_{tar}: 1.33 m/s, and (b) v_{tar}: 5.55 m/s.

Step 1, and its heading could be directed toward the path.

4.4.2. Autonomous vehicle test results

The proposed method approached the path faster than the other methods. When the cross-track error was large, the proposed method selected the look-ahead point closer to the path than the other methods. Thus, the heading of the vehicle was directed further toward the path. As the vehicle gradually approached the path, the look-ahead point was set to a larger distance from the path. Therefore, the heading of the vehicle was aligned to the direction of the path. In particular, when the vehicle approached close to the path, other methods were slightly off the path, (see Figure 9 (a)). The results obtained from the trajectory of the actual vehicle tests were noisier than those in the simulation tests because of localization errors.

4.5. Results for effectiveness of (*Step 2*) in Section 3

4.5.1. Simulation test results

In Figure 10 (a), the vehicle entered the corner. While driving along the straight path, the other models chose the look-ahead point on the path when the vehicle approached the corner. The vehicle drove inside the corner, and it deviated from the straight

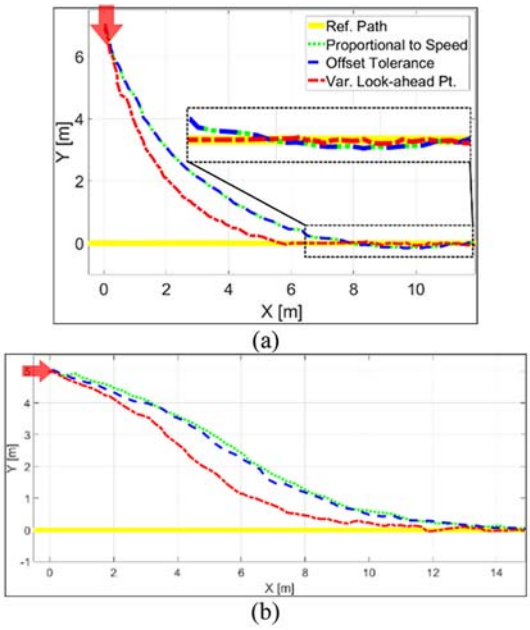

Figure 9. Vehicle test results (effectiveness of *Step 1*): Approaching the reference path from three directions at two velocities. (a) v_{tar}: 1.33 m/s, and (b) v_{tar}: 5.55 m/s.

Table 2. Simulation Test Results at Constant Curvature (Figure 10 (a))

			Method		
			Prop.	Offset	Var.
Cross -track Error	Cur.: 0.2	*Mean*	0.603	0.440	0.113
		Std.	0.777	0.629	0.186
	Cur.: 0.1	*Mean*	0.652	0.413	0.211
		Std.	0.659	0.435	0.177
	Cur.: 0.06	*Mean*	0.669	0.317	0.169
		Std.	0.485	0.382	0.229

Note. The unit of data is the meter and "Cur." is the curvature. "Prop.": '*Proportional to Speed*' method, "Offset": '*Offset tolerance*' method, "Var.": proposed method.

Table 3. Simulation Test Results at Various Curvature (Figure 10 (b))

		Method		
		Prop.	Offset	Var.
Cross-track Error	*Mean*	1.119	0.762	0.353
	Std.	0.596	0.461	0.243

Note. The unit of data is the meter.

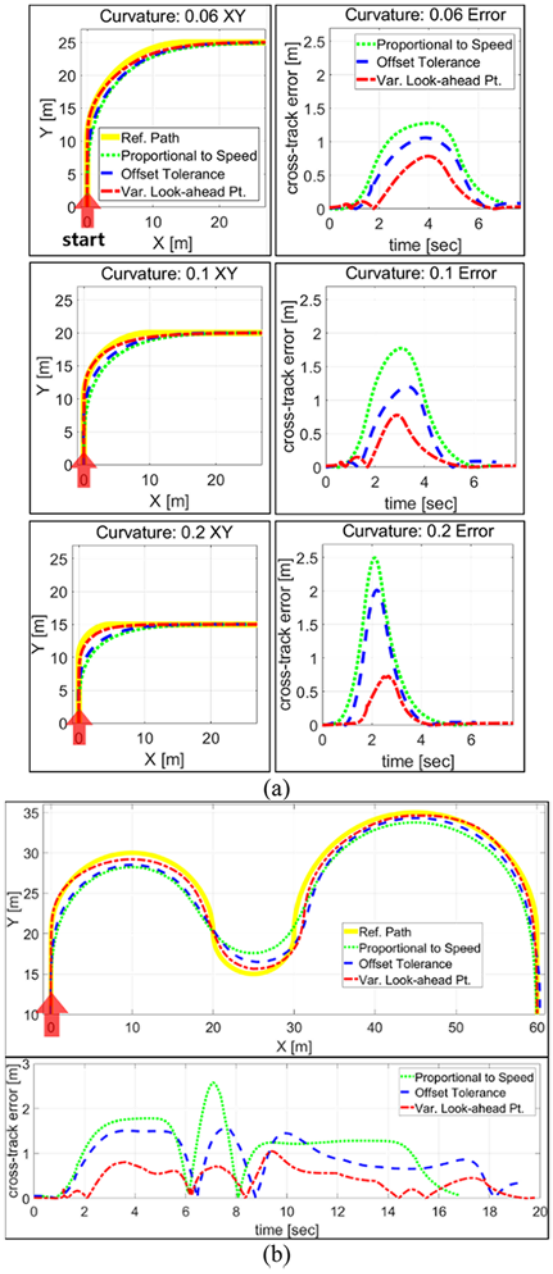

Figure 10. Simulation test results (v_{tar}: 5.55 m/s). The curvatures of the corners used in this experiment were 0.2, 0.1 and 0.06; v_{tar} was set to 5.55 m/s.: (a) Entering corners with curvatures of 0.06, 0.1, and 0.2, and (b) entering corners with various curvatures.

path. However, our proposed method chose the look-ahead point on the path so that the vehicle could track the straight path. Therefore, the vehicle did not drive inside the corner, and it could be tracked closer than all of the other methods. Table 2 shows that the cross-track error of our method is 3.76 and 2.28 times lower than the error of the '*Proportional to Speed*' method and '*Offset Tolerance*' method.

Figure 10 (b) shows the path where various curvatures are mixed. When the vehicle came out of one corner and then went into the other corner (see Figure 10 (b)) from 5.5 to 6.5 s and from 8.5 to 9.5 s, the proposed method had better tracking

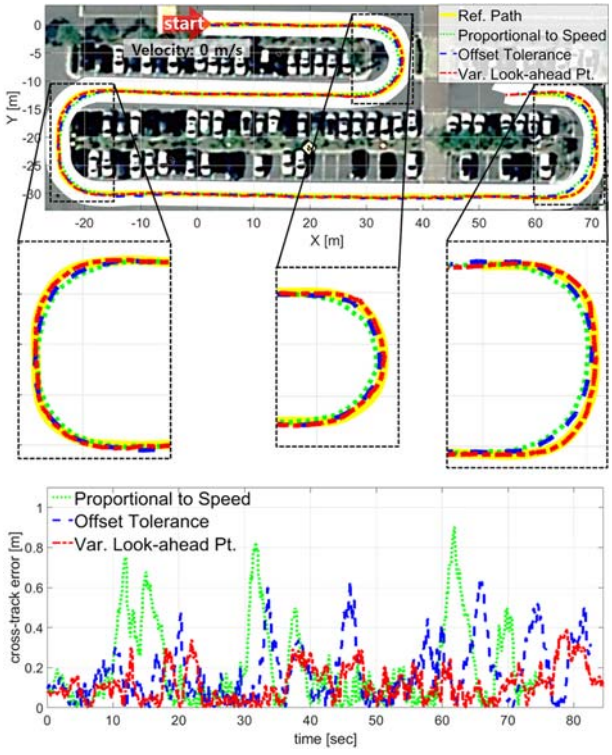

Figure 11. Vehicle test results in the parking lot (v_{tar}: 4.16 m/s). The path used in this experiment was 294 meters long, and the curvatures of the corners ranged from 0.12 to 0.2. (a) Pre-planned path and trajectory where the vehicle actually moved; (b) the cross-track error.

Table 4. Vehicle Test in Parking Lot (Figure 11)

		Method		
		Prop.	Offset	Var.
Cross-track Error	*Mean*	0.197	0.174	0.131
	Std.	0.202	0.143	0.101

Note. The unit of data is the meter.

performance. Table 3 shows that the cross-track error of our method is 3.169 times lower than the error of '*Proportional to Speed*' and 2.15 times lower than the error of '*Offset Tolerance*'.

4.5.2. Autonomous vehicle test results

The vehicle was tested in the real parking lot (see Figure 11 and video: https://youtu.be/1nRdlEsYopY). The proposed method tracked the corner more closely than the other methods. Table 4 shows that the cross-track error of our method is 1.50 times lower than the error of the '*Proportional to Speed*' method and 1.32 times lower than the error of the '*Offset Tolerance*' method. In the vehicle test, more errors may occur depending on the speed of the steering angle change at the point where the curvature changes largely. It occurs a maximum error of 0.31 m when entering the corner and 0.38 m when exiting the corner. Overall, the cross-track errors in the vehicle test are unsteady than those in the simulation test because of the localization drift and the disturbance.

5. CONCLUSIONS

This paper presents a heuristic method to select the look-ahead point in the pure pursuit method for accurate path tracking in complex and narrow road environments. The problem of the selection of a look-ahead point was investigated into two steps. First, the look-ahead point was chosen on the path by considering the position and the direction of the vehicle and the path. Second, when the vehicle was close to the path and the curvature change became large, the look-ahead point was selected outside the path by the proposed method.

The effectiveness of this method was verified through the simulation and the autonomous vehicle experiments in the various paths. The experimental results show that our method can reduce the cross-track error by up to 41 % over the previous pure pursuit methods. When the vehicle approaches the path from outside, it approaches the path rapidly and converges without the overshoot problem as compared with the previous pure pursuit methods. The cutting-corner problem was significantly reduced when the vehicle approached the path with a larger curvature than the current one, for example, at an intersection. Moreover, the vehicle could track the paths with various curvatures more accurately than the previous pure pursuit methods.

To track the path more accurately in various situations, especially with different curvatures and velocities, it is necessary to obtain the appropriate values of α_{max} and β_{max} depending on situations. In future investigations, these parameters will be determined online using driving data.

REFERENCES

Andersen, H., Chong, Z. J., Eng, Y. H., Pendleton, S. and Ang, M. H. (2016). Geometric path tracking algorithm for autonomous driving in pedestrian environment. *2016 IEEE Int. Conf. Advanced Intelligent Mechatronics (AIM)*. Banff, Alberta, Canada.

Bu, X., Hou, Z. and Chi, R. (2013). Model free adaptive iterative learning control for farm vehicle path tracking. *IFAC Proc. Volumes* **46**, **20**, 153–158.

Campbell, S. (2007). *Steering control of an autonomous ground vehicle with application to the DARPA urban challenge*. Ph.D. Doctoral dissertation. Massachusetts Institute of Technology, Cambridge, MA, USA.

Challenge, D. U. (2007). Route network definition file (RNDF) and mission data file (MDF) formats. *Tech. Rep.*, *Defense Advanced Research Projects Agency*, Tech. Rep.

Chen, C. and Tan, H. S. (1999). Experimental study of dynamic look-ahead scheme for vehicle steering control. *Proc. 1999 American Control Conf.* San Diego, CA, USA.

Coulter, R. C. (1992). *Implementation of the pure pursuit path*

tracking algorithm. Carnegie-Mellon UNIV Pittsburgh PA Robotics INST.

d'Andréa-Novel, B., Campion, G. and Bastin, G. (1995). Control of nonholonomic wheeled mobile robots by state feedback linearization. *Int. J. Robotics Research* **14**, **6**, 543–559.

Dubins, L. E. (1957). On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American J. Mathematics* **79**, **3**, 497–516.

Hingwe, P. and Tomizuka, M. (1998). A variable look-ahead controller for lateral guidance of four wheeled vehicles. *Proc. 1998 American Control Conf.* Philadelphia, PA, USA.

Kanayama, Y., Kimura, Y., Miyazaki, F. and Noguchi, T. (1990). A stable tracking control method for an autonomous mobile robot. *Proc., IEEE Int. Conf. Robotics and Automation*. Cincinnati, OH, USA.

Kilinç, A. S. and Baybura, T. (2012). Determination of minimum horizontal curve radius used in the design of transportation structures, depending on the limit value of comfort criterion lateral jerk. *TS06G-Engineering Surveying*, *Machine Control and Guidance*, *Rome*, *Italy*.

Kim, E., Kim, J. and Sunwoo, M. (2014). Model predictive control strategy for smooth path tracking of autonomous vehicles with steering actuator dynamics. *Int. J. Automotive Technology* **15**, **7**, 1155–1164.

Kuwata, Y., Teo, J., Karaman, S., Fiore, G., Frazzoli, E. and How, J. (2008). Motion planning in complex environments using closed-loop prediction. *AIAA Guidance, Navigation and Control Conf. Exhibit*. Honolulu, Hawaii.

Paden, B., Čáp, M., Yong, S. Z., Yershov, D. and Frazzoli, E. (2016). A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Trans. Intelligent Vehicles* **1**, **1**, 33–55.

Peng, H., Wang, W., An, Q., Xiang, C. and Li, L. (2020). Path tracking and direct yaw moment coordinated control based on robust MPC with the finite time horizon for autonomous independent-drive vehicles. *IEEE Trans. Vehicular Technology*.

Piao, C., Liu, X. and Lu, C. (2019). Lateral control using parameter self-tuning LQR on autonomous vehicle. *2019 Int. Conf. Intelligent Computing, Automation and Systems (ICICAS)*. Chongqing, China.

Qinpeng, S., Zhonghua, W., Meng, L., Bin, L., Jin, C. and Jiaxiang, T. (2019). Path tracking control of wheeled mobile robot based on improved pure pursuit algorithm. *2019 Chinese Automation Congress (CAC)*. Hangzhou, China.

Rajamani, R. (2011). *Vehicle dynamics and control*. Springer Science & Business Media. Berlin, Germany.

Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., Stang, P., Strohband, S., Dupont, C., Jendrossek, L.-E., Koelen, C., Markey, C., Rummel, C., van Niekerk, J., Jensen, E., Alessandrini, P., Bradski, G., Davies, B., Ettinger, S., Kaehler, A., Nefian, A. and Mahoney, P. (2006). Stanley: The robot that won the DARPA Grand Challenge. *J. Field Robotics* **23**, **9**, 661–692.

Urmson, C., Anhalt, J., Bagnell, D., Baker, C., Bittner, R., Clark, M. N., Dolan, J., Duggins, D., Galatali, T., Geyer, C., Gittleman, M., Harbaugh, S., Hebert, M., Howard, T. M., Kolski, S., Kelly, A., Likhachev, M., McNaughton, M., Miller, N., Peterson, K., Pilnick, B., Rajkumar, R., Rybski, P., Salesky, B., Seo, Y.-W., Singh, S., Snider, J., Stentz, A., Whittaker, W., Wolkowicki, Z., Ziglar, J., Bae, H., Brown, T., Demitrish, D., Litkouhi, B., Nickolaou, J., Sadekar, V., Zhang, W., Struble, J., Taylor, M., Darms, M. and Ferguson, D. (2008). Autonomous driving in urban environments: Boss and the urban challenge. *J. Field Robotics* **25**, **8**, 425–466.

Wit, J. S. (2000). *Vector pursuit path tracking for autonomous ground vehicles*. Ph.D. Doctoral dissertation. University of Florida. Gainesville, FL, USA.

Xu, S. and Peng, H. (2019). Design, analysis, and experiments of preview path tracking control for autonomous vehicles. *IEEE Trans. Intelligent Transportation Systems* **21**, **1**, 48–58.

Zhang, J. and Singh, S. (2014). LOAM: Lidar Odometry and Mapping in Real-time. *2014 Robotics: Science and Systems Conf.* Berkeley, CA, USA.