2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)
Banff, Alberta, Canada, July 12–15, 2016

Geometric Path Tracking Algorithm for Autonomous Driving in Pedestrian Environment

Hans Andersen*, Zhuang Jie Chong †,You Hong Eng‡, Scott Pendleton*, Marcelo H. Ang Jr.*
*Department Of Mechanical Engineering, National University of Singapore
{hans.andersen, scott.pendleton01}@u.nus.edu, mpeangh@nus.edu.sg
†nuTonomy, demian@nutonomy.com
‡Singapore-MIT Alliance for Research and Technology, youhong@smart.mit.edu

***Abstract*—This paper proposes an alternative formulation to the pure pursuit path tracking algorithm for autonomous driving. The current approach has tendencies to cut corners, and therefore results in poor path tracking accuracy. The proposed method considers not only the relative position of the pursued point, but also the orientation of the path at that point. A steering control law is designed in accordance with the kinematic equations of motion of the vehicle. The effectiveness of the algorithm is then tested by implementing it on an autonomous golf cart, driving in a pedestrian environment. The experimental result shows that the new algorithm reduces the root mean square (RMS) cross track error for the same given pre-programmed path by up to 46 percent, while having virtually no extra computational cost, and still maintaining the chatter free property of the original pure pursuit controller.**

I. Introduction

Autonomous vehicles offer potential for additional safety, increased productivity, greater accessibility, better road efficiency, and positive impact to the environment [1].

Research in autonomous vehicles has seen dramatic advances in recent years, due to the increases in available computing power and reduced cost in sensing and computing technologies. Recent competitions such as the 2007 DARPA Urban Challenge [2] have accelerated the field of autonomous vehicle design and the development.

The main functions in an autonomous vehicles are mapping, localization, perception, navigation, and control. Path tracking is one of the core elements of an autonomous vehicle navigation system. Path tracking's main objective is to compute the control commands, with the relevant motion constraints considered, such that the vehicle is able to follow a previously planned path. The resulting control input should minimize the difference between the planned and actual path with respect to the lateral distance and the vehicle heading, as well as complying to the motion constraints of the vehicle, and ensure that the resulting movements are smooth, while maintaining the stability of the controller.

There are various methods that have been presented in the literature. Two of the most popular types are geometric methods, and model based methods [3].

Geometric path tracking algorithms use simple geometric relations to come up with steering control laws. These techniques utilize look ahead distance to measure error ahead of the vehicle and can extend from simple circular arc calculations to much more complex geometric theorems, such as the vector pursuit method [4]. Compared to model based path tracking algorithms, geometric path tracking algorithms are relatively simpler to implement, more robust to path curvatures, and tend to work better for lower speed driving. However, tracking accuracy at higher speed, where the effects of non-linear vehicle dynamics have to be taken into account are generally better with model based algorithms compared to geometric methods.

Model based path tracking methods uses either first order (kinematic) or second order (dynamic) model of the vehicle. Model based controllers tend to perform well for higher speed driving applications such as autonomous highway driving, however, model based methods require the path to be continuous, and are not robust to disturbances and large lateral offsets, resulting in tendency to cut corners as the vehicle rapidly accelerates and decelerates and pursues paths with large curvature.

One of the most popular geometric path tracking algorithms is the pure pursuit path tracking algorithm. This algorithm will be discussed in detail in section III. The pure pursuit algorithm pursues a point along the path that is located at a certain distance away from the vehicle's current position. The algorithm is relatively simple and easy to implement, and is robust to disturbances and large lateral error. The input to the algorithm is also way points, rather than smooth curves, and is therefore less susceptible to discretization related issues. This algorithm still suffers from corner cutting and steady state error problems if the lookahead distance selected is not appropriate, especially at higher speed, when the lookahead distance increases.

The Stanley method is another popular geometric steering controller that was first introduced with Stanford University's entry in the DARPA Urban Challenge [5]. The Stanley method computes the steering command based on a non-linear control law that considers the cross track error of the vehicle to the path, as measured from the front axle of the vehicle, as well as the heading error of the vehicle with respect to the path. The algorithm has been proven to exponentially converges to zero cross track error.

Compared to the pure pursuit method, the Stanley method, has better tracking results and does not cut corners as it uses

978-1-5090-2065-2/16/$31.00 ©2016 IEEE

cross track error and yaw heading error information of the vehicle with respect to the path as measured from the front axle rather than pursuing a point that is located at a certain distance ahead of the vehicle. It also performs better at high speed driving compared to the pure pursuit method.

However, the Stanley method is not as robust to disturbances, and has higher tendency for oscillation as compared to the pure pursuit method. The Stanley method also requires continuous curvature path rather than way points, which makes it susceptible to discretization related problems [6].

In this paper, we consider the problem of driving in pedestrian environment, where the driving velocity is relatively low, but the curvature of the tracked path can be relatively high, in this case, geometric steering algorithms are generally preferred compared to model based path tracking algorithms. We propose a modified formulation of the pure pursuit algorithm that addresses the cutting corner and overshoot problem, by considering the heading orientation of the pursuit point. We examine the effectiveness of the proposed approach by conducting autonomous driving experiments in National University of Singapore's university town plaza area. The experiment result shows that this method reduces the corner cutting and overshoot problems that we previously encountered when implementing the original pure pursuit algorithm.

The paper is organized as follows. In section II the kinematic vehicle model will be briefly reviewed. The standard pure pursuit algorithm will be reviewed in section III. Section IV presents the alternative formulation to the pure pursuit algorithm. Section V discusses the experimental setup, while section VI presents the experiment results. The paper is concluded in section VII.

II. Vehicle Model

In this section, a brief review of the vehicle model used to derive the control laws will be discussed.

In deriving the geometric path following algorithm, the kinematic car model is used. The fundamental basis for the kinematic car model is the Ackermann steering geometry [7]. The Ackermann steering geometry assumes that during a turn, each of the vehicle's tires moves along a circular arc with common center of rotation. This means that each tire is in pure rolling motion with zero lateral acceleration.

The assumption that there is zero lateral motion on the tires may have negative impact on the tracking performance of the controller as the velocity increases and path curvature varies. Alternative approaches using dynamic models of the vehicle to design steering controllers have been widely introduced in literature such as in [6] . However, the dynamics of the car can be complicated, and therefore many assumptions have to be made. The controllers based on these dynamic models can be very non linear, and therefore may result in significant increase in computational cost. These controllers are also susceptible to errors in modelling and parameter identifications. Depending on the applications, the trade-offs of implementing these more complicated controllers may not be desirable.

A common simplification of the Ackermann steering geometry is the bicycle model. The bicycle model simplifies the four wheeled Ackermann steered car into two wheeled model by combining the two front wheels together and the two rear wheels together. This model is often used in literature to derive steering control laws as it approximates the motion of the vehicle reasonably well at low speeds and moderate steering angles.

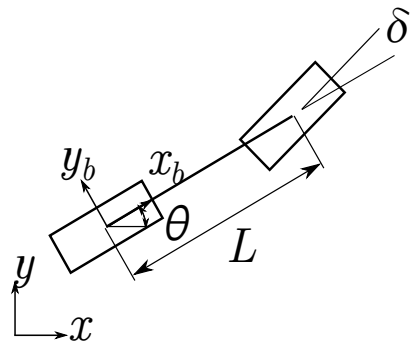

Fig. 1. Kinematic bicycle model of an Ackermann-steered vehicle

Referring to Fig. 1, a coordinate system is assigned to the rear axle of the vehicle., and therefore the velocity of the vehicle (v) in the body frame is purely in x_b direction. The vehicle's yaw (θ) and yaw rate $(\dot{\theta})$ are geometrically related to the front wheel steering angle (δ) by the following equation:

$$\dot{\theta} = \frac{v}{L} tan(\delta) \tag{1}$$

where L is the vehicle wheelbase (distance between the front axle and the rear axle of the vehicle).

The translational equations of the vehicle body fixed frame origin with respect to the inertial frame can then be approximated as follows:

$$\dot{x} = vcos(\theta) \tag{2}$$

$$\dot{y} = vsin(\theta) \tag{3}$$

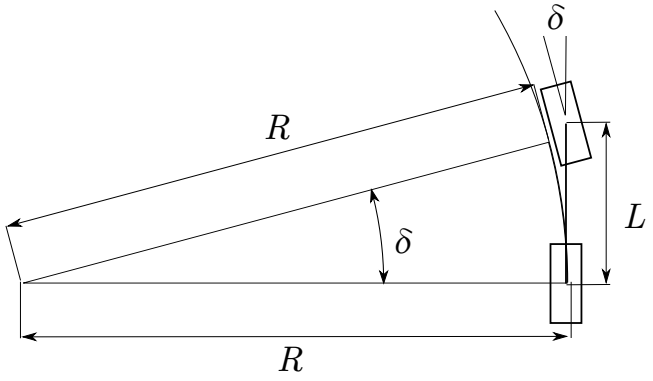

Fig. 2. Steering geometry of the kinematic bicycle model of an Ackermann-steered vehicle

Referring to Fig. 2, the geometric relation between the wheel steering angle and the curvature radius of the path that the rear wheel will follow can be approximated as:

$$tan(\delta) = \frac{L}{R} \tag{4}$$

III. Pure Pursuit Path Tracking

In this section, a brief review of the pure pursuit algorithm will be discussed, and the relevant equations will be derived.

A. Path Representation

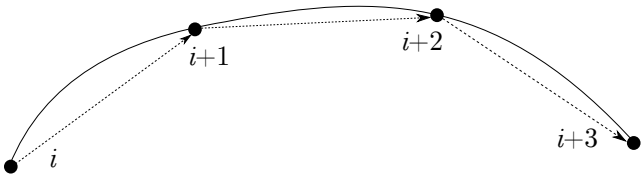

Fig. 3. Piecewise linear representation of a smooth trajectory

In order to avoid constricting the way that the paths are generated, the paths shall be represented in a piece wise linear way [7]. A smooth trajectory having continuous first and second order derivatives, e.g. a bezier curve should be represented as a sequence of dense points. A point is joined to the following point in the path by linear path segments. In order to closely approximate the continuous path, denser discretization of the path has to be implemented. The discrete nodes are contained in an ordered list, and the way points are tracked sequentially (way point i is tracked before way point $i+1$). Referring to Fig. 3, all of the piecewise linear segments are connected.

B. Pure Pursuit Algorithm

The pure pursuit algorithm has a single tunable parameter, the lookahead distance L_{fw}. The algorithm defines a virtual circle arc that connects the anchor point (the rear axle) to the tracked point along the path that is located L_{fw} away from the anchor point. A variation of this algorithm has been introduced by [8], where the anchor point is not necessarily selected as the rear axle, but can be located at a distance ϵ away from the rear axle along x_b. However, in this paper, the anchor point will be assumed to be the rear axle.

The virtual arc is constrained to be tangential to the velocity vector at the origin of the body fixed frame and to pass through the tracked point along the path. Geometrically, using the law of sines, the radius of the curvature of the arc R can be computed as:

$$\frac{L_{fw}}{sin(2\eta)} = \frac{R}{sin(\frac{\pi}{2} - \eta)}$$
$$\frac{L_{fw}}{2sin(\eta)cos(\eta)} = \frac{R}{cos(\eta)}$$
$$\frac{L_{fw}}{sin(\eta)} = 2R$$

Where R is the turning radius, and η is the lookahead heading. The curvature κ of the arc is defined as:

$$\kappa = \frac{2sin(\eta)}{L_{fw}} \tag{5}$$

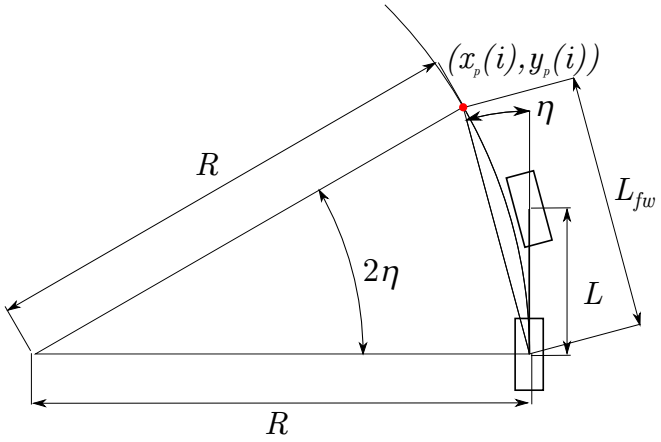

Fig. 4. Pure pursuit steering geometry

And therefore the vehicle's steering angle δ can be computed using the kinematic bicycle model derived in section II as:

$$\delta = tan^{-1}(\kappa L)$$
$$\delta(t) = tan^{-1}\left(\frac{2Lsin(\eta(t))}{L_{fw}}\right) \tag{6}$$

The lookahead distance is commonly formulated as a function of longitudinal velocity, and saturated at certain maximum and minimum value, and thus equation (7) can be rewritten as

$$\delta(t) = tan^{-1}\left(\frac{2Lsin(\eta(t))}{kv(t)}\right) \tag{7}$$

At lower speed, when the lookahead distance is smaller, the vehicle is expected to track the path closely, and oscillatory behaviour is also expected; meanwhile at higher velocity, when the lookahead distance is larger, the vehicle is expected to track the path smoothly, however this will result in the cutting corner problem.

Since the lookahead distance is a function of the gain k, selecting the appropriate value for the gain will result in significant trade-offs in the tracking performance. On one hand, if the gain k is set to be too low, instability will occur, on the other hand, if the gain k is set to bee too large, poor tracking performance will be expected. Tuning the pure pursuit controller to achieve a good path tracking result which minimizes the corner cutting and overshoot problems can be a tedious and course dependent challenge. One of the main reasons is that the pure pursuit path tracking algorithm does not consider the curvature, and the heading of the tracked point along the path. This issue will be addressed in the next section.

IV. Modified Pure Pursuit Path Tracking

Consider the piecewise linear path segment depicted in Fig. 3. The heading of way point i can be computed as

$$\theta_p(i) = tan^{-1}\left(\frac{y_p(i+1) - y_p(i)}{x_p(i+1) - x_p(i)}\right) \tag{8}$$

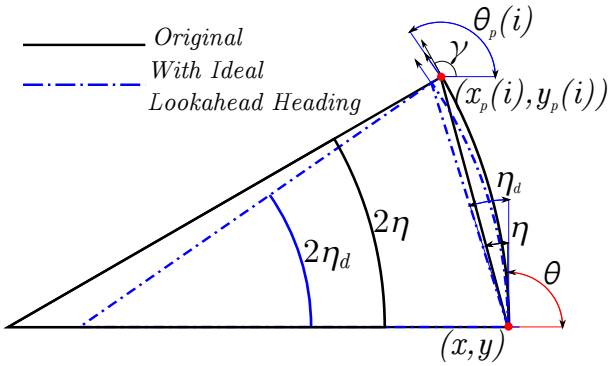

Fig. 5. Tangential heading of pure pursuit circular arc

Referring to Fig. 5, the tangential heading of the circular arc segment γ at the tracked way point $(x_p(i), y_p(i))$ can be graphically obtained as:

$$\gamma = \theta + 2\eta \tag{9}$$

In order to pursue the correct heading of way point i $(\theta_p(i))$ from the current position (x, y) and heading (θ), the corresponding lookahead angle η_d can be computed by substituting $\gamma = \theta_p(i)$, and $\eta = \eta_d$ to be:

$$\eta_d = \frac{\theta_p(i) - \theta}{2} \tag{10}$$

In order to calculate the corresponding steering angle, the tracked way point is now translated by some unknown distance d perpendicular to the path segment from $(x_p(i), y_p(i))$. The offset distance d can take either positive or negative value.

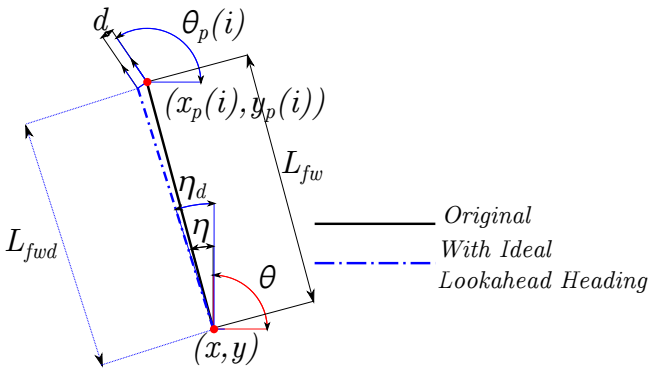

Fig. 6. Offset distance and lookahead distance of the pursued path with desired circular arc heading

Referring to Fig. 6, the offset distance d and the new lookahead distanceL_{fwd} can be computed as

$$\begin{bmatrix} x - x_p(i) \\ y - y_p(i) \end{bmatrix} = \begin{bmatrix} cos(\theta_p(i) + \frac{\pi}{2}) & -cos(\theta + \eta_d) \\ sin(\theta_p(i) + \frac{\pi}{2}) & -sin(\theta + \eta_d) \end{bmatrix} \begin{bmatrix} d \\ L_{fwd} \end{bmatrix}$$

$$\begin{bmatrix} d \\ L_{fwd} \end{bmatrix} = \begin{bmatrix} cos(\theta_p(i) + \frac{\pi}{2}) & -cos(\theta + \eta_d) \\ sin(\theta_p(i) + \frac{\pi}{2}) & -sin(\theta + \eta_d) \end{bmatrix}^{-1} \begin{bmatrix} x - x_p(i) \\ y - y_p(i) \end{bmatrix} \tag{11}$$

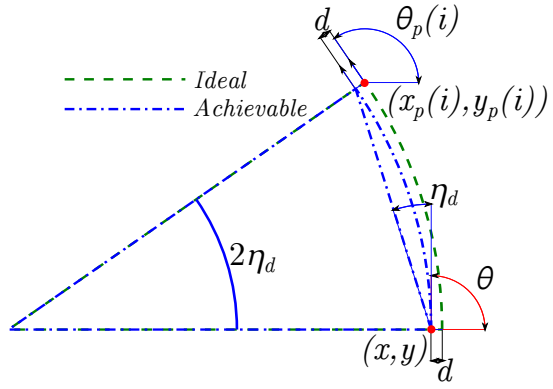

Fig. 7. Offset distance d as expected cross track error for constant curvature path

Referring to Fig. 7, the distance d would be the expected current cross track error if it were pursuing a constant curvature path. Which means that ideally, in order for the vehicle to reach the tracked way point i with the correct heading, its position has to be located at a distance $-d$ away in the direction tangential to its current heading $(\theta + \frac{\pi}{2})$ from its current position (x, y).

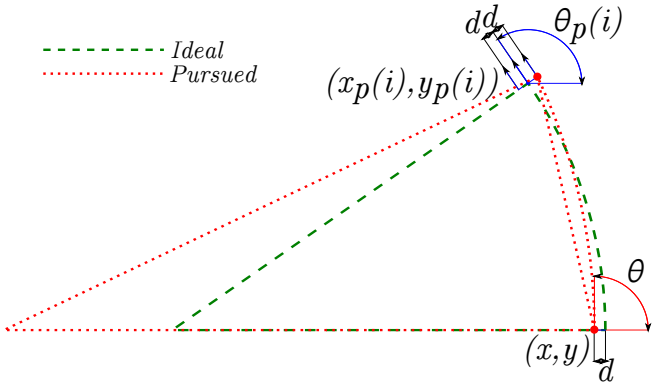

Fig. 8. Offset distance d compensation for pure pursuit path tracking with desired path heading considered

Referring to Fig. 8 in order to correct for this error, a modified tracked way point can be used. The point is translated from the original tracked way point by a distance of $-d$ away in the direction $(\theta + \frac{\pi}{2})$, inverting the expected cross track error.

$$d = \begin{cases} sgn(d)\ D, & \text{if } |d| \geq D \\ d, & \text{otherwise} \end{cases}$$

A saturation constant D can be introduced to limit the maximum offset that is tolerated by the algorithm. This

constant has to be set at a reasonable amount in order to avoid instability that can be caused by huge initial cross track and/or heading error, or otherwise a bad representation of the continuous path.

The modified tracked way point can then be written as:

$$\begin{bmatrix} x_{pm}(i) \\ y_{pm}(i) \end{bmatrix} = \begin{bmatrix} x_p(i) \\ y_p(i) \end{bmatrix} - \begin{bmatrix} d\ cos(\theta_p(i) + \frac{\pi}{2}) \\ d\ sin(\theta_p(i) + \frac{\pi}{2}) \end{bmatrix} \quad (12)$$

and the corresponding steering angle can be computed using the pure pursuit algorithm.

V. Experimental Setup

The experiment is conducted with a Yamaha YDREX3 electric golf buggy retrofitted for autonomous functionality. Key elements of the retrofitted systems are highlighted in Fig. 9, and the full system overview is discussed in detail in [1].

Fig. 9. Hardware overview, highlighting primary retrofit additions to a Yamaha YDREX3 golf buggy in order to enable autonomous capabilities, [Source: [1]]

The autonomous driving experiment is conducted in the plaza area of National University of Singapore's University Town. Fig. 10 presents the previously mapped area. The clockwise path was created by manually driving the golf buggy and recording the localization poses. The poses are fitted with bezier curves. The path is generated by discretizing the bezier curves with 10 cm resolution. The total length of the path is 192 m, and its maximum curvature is 0.16 m^{-1}.

The maximum driving speed in the pedestrian environment is set at 2.0 m/s. The pure pursuit lookahead distance is dependent on the commanded velocity v and is formulated as:

$$L_{fw}(v) = \begin{cases} L_{min}, & \text{if } L_{min} < 2.24v \\ 2.24v, & \text{if } L_{min} \leq 2.24v \leq 12 \\ 12, & \text{otherwise} \end{cases}$$

Where L_{min} is the minimum lookahead distance. The experiment is carried with varied minimum lookahead distance, as well as varied offset tolerance D.

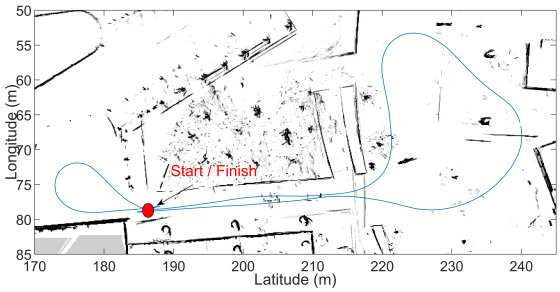

Fig. 10. Reference path in pedestrian environment, top down view. White pixels represent empty space, while black pixels represent vertical features extracted with the synthetic 2D LIDAR [9].

VI. Experimental Results

The cross track error is defined as the minimum distance between the anchor point to the path. The RMS errors of each of the experiment is summarized in Table 1. Fig. 12 plots the cross track error histogram, Fig. 12 plots the cross track error on different sections of the path, and Fig. 13 plot the cross track error as a function of the distance travelled along the path, for various D with L_{min} set to be constant at 3 m.

TABLE I
RMS error (m) for experiments with different parameters

Offset Tolerance (cm)	Minimum Lookahead Distance (m)		
	1.5	3	4.5
0	0.1408	0.1936	0.3451
10	0.1163	0.1264	0.2684
20	0.0921	0.1172	0.2582
30	0.0842	0.1037	0.2284

As it has been discussed in section III, tuning the pure pursuit algorithm is situational, and the lookahead distance values that is formulated in section V is hand tuned for our specific environment. The increase in RMS error value as the minimum lookahead distance is increased is expected since the corner cutting effect increases as the lookahead distance is increased.

However, it can be concluded from table 1 that the performance of the modified pure pursuit algorithm is better for each of the different minimum lookahead distances. This is true even for $L_{min} = 4.5$ m, where the lookahead distance does not depend on the velocity any more because $2.24v$ will always be smaller than L_{min} in this case.

The histogram of the error (Fig. 11) also shows that there is a significant shift in the distribution of the cross track error. Given the same lookahead distance formulation, the modified pure pursuit algorithm reduces the maximum cross track error magnitude, narrows down the spread of the distribution, and generally has a more pronounced peak in its distribution as compared to the original pure pursuit algorithm. Still we cannot conclude that the distribution is normal and it is path specific.

Referring to Fig. 12, It can be concluded that the original pure pursuit algorithm performs very well on long straight

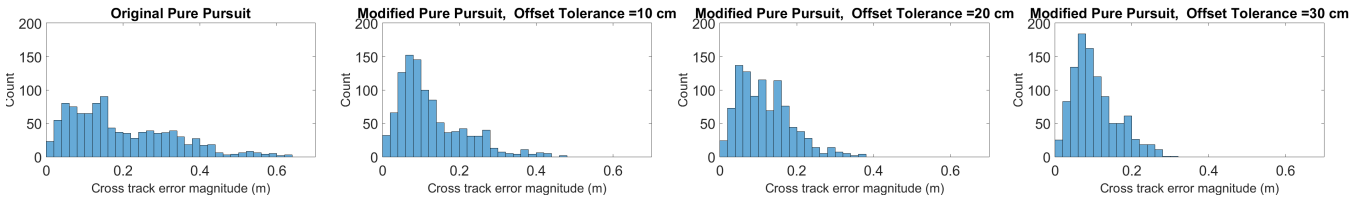

Fig. 11. Cross track error histogram for experiment with L_{min} =3.0 m

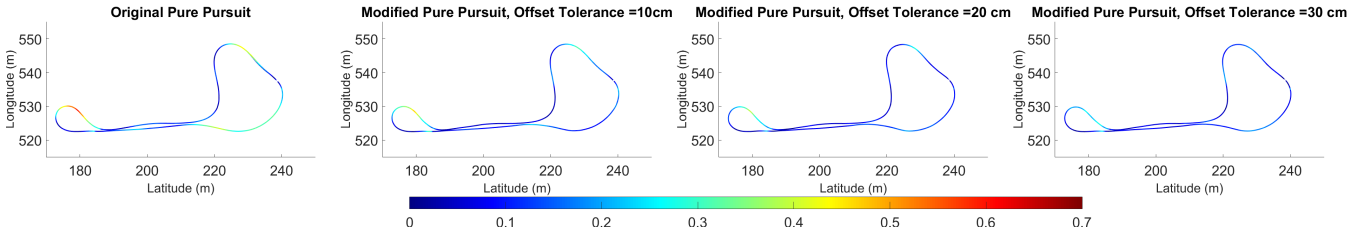

Fig. 12. Cross track error intensity plot for experiment with L_{min} =3.0 m

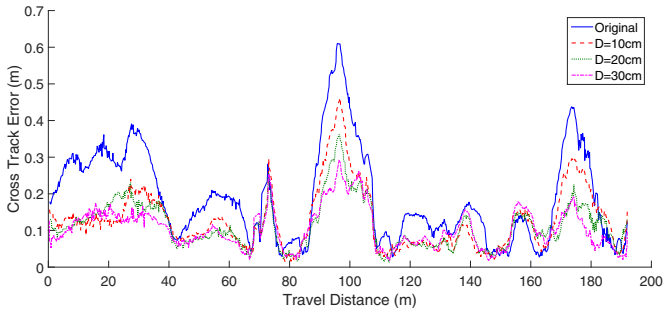

Fig. 13. Cross track error for experiment with L_{min} =3.0 m

sections of the road, however the corner cutting effects are apparent as the curvature of the tracked path increases.

The effectiveness of the modified pure pursuit algorithm is consistently shown for every value of minimum lookahead distance L_{min}. It achieves similar performance to the original pure pursuit algorithm at the straight line sections, and improves the cross track error magnitude significantly when it is tracking path sections with large curvature.

While keeping the minimum lookahead constant, increasing the offset tolerance D decreases the RMS error. This demonstrates that on some sections of the path where the offset d is large, compensating the way point closer to the original value of d is beneficial. However, in order to avoid instability in the controller, appropriate value has to be assigned to D.

VII. Conclusion

In this paper, we describe an alternative formulation for the pure pursuit geometric path tracking algorithm. The modified pure pursuit algorithm uses the orientation information at the tracked way point to compute the steering input, rather than just the position of the tracked point. The effectiveness of the method has been tested by driving a golf buggy autonomously in a pedestrian environment. The experimental results have shown that the modified algorithm can reduce the RMS cross track error by up to 46 percent. We leave the rigorous theoretical derivation of the performance limitations of the controller, as well as the calculation of optimal values for D for future works, as these information can be very useful for local path planning.

Acknowledgement

This research was supported by the Future Urban Mobility project of the Singapore-MIT Alliance for Research and Technology (SMART) Center, with funding from Singapore's National Research Foundation (NRF).

References

[1] S. Pendleton, T. Uthaicharoenpong, Z. J. Chong, G. Ming, J. Fu, B. Qin, W. Liu, X. Shen, Z. Weng, C. Kamin, M. A. Ang, L. T. Kuwae, K. A. Marczuk, H. Andersen, M. Feng, G. Butron, Z. Z. Chong, M. H. Ang, E. Frazzoli, and D. Rus, "Autonomous Golf Cars for Public Trial of Mobility-on-Demand Service," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 1164–1171.

[2] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. N. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. M. Howard, S. Kolski, A. Kelly, M. Likhachev, M. McNaughton, N. Miller, and D. Peterson, "Autonomous driving in urban environments: Boss and the Urban Challenge," *J. Field Robotics*, vol. 25, pp. 425–466, 2008.

[3] M.-W. Park, S.-W. Lee, and W.-Y. Han, "Development of Lateral Control System for Autonomous Vehicle Based on Adaptive Pure Pursuit Algorithm," in *International Conference on Control, Automation and Systems*, no. 14th, 2014, pp. 369–372.

[4] J. S. . Wit, "Vector Pursuit Path Tracking for Autonomous Ground Vehicles," Ph.D. dissertation, University of Florida, 2000.

[5] G. M. Hoffmann, C. J. Tomlin, M. Montemerlo, and S. Thrun, "Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing," *Proceedings of the American Control Conference*, pp. 2296–2301, 2007.

[6] J. M. Snider, "Automatic Steering Methods for Autonomous Automobile Path Tracking," Tech. Rep., 2009.

[7] S. F. Campbell, "Steering Control of an Autonomous Ground Vehicle with Application to the DARPA Urban Challenge," Ph.D. dissertation, Massachusetts Institute of Technology, 2005.

[8] Y. Kuwata, J. Teo, and S. Karaman, "Motion planning in complex environments using closed-loop prediction," *Proc. AIAA Guidance, ...*, 2008.

[9] Z. J. Chong, B. Qin, T. Bandyopadhyay, M. H. Ang, E. Frazzoli, and D. Rus, "Mapping with synthetic 2D LIDAR in 3D urban environment," *IEEE International Conference on Intelligent Robots and Systems*, vol. 3, no. 2, pp. 4715–4720, 2013.