International Journal of Automotive Technology, Vol. 18, No. 6, pp. 1099–1107 (2017)
DOI 10.1007/s12239–017–0107–4

pISSN 1229–9138/ eISSN 1976–3832

APPLICATION OF ITERATIVE LEARNING CONTROL IN TRACKING A DUBIN'S PATH IN PARALLEL PARKING

Benjamas Panomruttanarug*

Department of Control System and Instrumentation Engineering, King Mongkut's University of Technology Thonburi, Bangkok 10140, Thailand

(Received 21 October 2016; Revised 11 May 2017; Accepted 15 May 2017)

ABSTRACT–Intelligent parking assist systems will soon be available for most vehicles on the market. Many optimal parking trajectories and control strategies have been proposed for reverse parking. However, most of these require intensive computation, causing difficulties in practical use. This paper makes use of a classical path planning method to find the shortest parking path, and establishes the possibility of integrating iterative learning control (ILC) to exploit the capability of learning from experience to track the designed path. The effectiveness of the ILC structure is demonstrated by simulation and experiments. Tracking performance is shown to be much improved by using a simple learning control law.

KEY WORDS : Parallel parking, Dubin's path, Iterative learning control, Steering control

1. INTRODUCTION

Nowadays, there are many commercial parking systems available on the market, for example, Intelligent Parking Assist (IPA) from Toyota, Active Park Assist (APA) from Ford, Parking Assistant from BMW, Parktronic from Audi, and ParkSense from Jeep. Recent automated parking systems from Bosch and BMW allow their users to activate autonomous parking from a smart device. These intelligent parking assist systems (IPAS) bring many benefits to a driver by removing difficulty and stress, allowing cars to fit into tight parking spaces, reducing the amount of time and traffic disruption required while parking, as well as preventing minor dents and scratches by less-skilled drivers.

Besides these commercial products, there are a number of literature studies on autonomous parallel parking system. Wang *et al*. (2014) provided a survey on the automatic parking of vehicles. In general, current parallel parking approaches can be divided into two categories: the path planning approach and the skill-based approach. The former generates a collision-free trajectory based on the vehicle's dynamics and geometric constraints of the environment, and control commands are then generated to follow the reference path. In path planning, accurate environmental information is required in advance to establish an off-line path. The latter makes use of intelligent control methods to generate actions based on the parking skill of an experienced human driver. Distance or vision sensors are used to send feedback signals to a vehicle controller to calculate a proper steering angle along the parking process. The parking trajectory is designed in real-time, based on a parking control algorithm programmed into the controller. Intelligent control methods, such as fuzzy logic control (Li *et al*., 2016; Vorobieva *et al*., 2015; Chen and Feng, 2009; Zhao *et al*., 2016; Panomruttanarug and Higuchi, 2010) and neural networks (Heinen *et al*., 2006; Oentaryo and Pasquier, 2004; Jenkins and Yuhas, 1993) are proposed for automatic parking. Between the two methods, fuzzy logic control seems to be more popular and effective. Kong and Kosko (1990) shows the advantages of using fuzzy control over neural networks to address the reversing problem. Meanwhile, Daxwanger and Schmidt (1995) combined the advantages of using fuzzy control and neural networks to accomplish automatic parking. Lee *et al*. (2009) demonstrated a combination of other control strategies: genetic algorithm, Petri net, and fuzzy logic control, to perform the optimal parking procedure.

Over the past two decades, path planning algorithms for parallel parking have been significantly proposed since skill-based algorithms require a large amount of calculation and are difficult to implement. In path planning, the optimal path between two oriented points can be obtained based on certain criteria, for example, shortest path, shortest time, or smoothest path. The most common shortest path method was proposed by Dubins (1957) whereby the shortest path is computed from minimum turning radius circles created by a maximum steering angle. Further studies have introduced some feasible paths, creating different shapes, for example, a set of cubic curves (Kanayama and Hartman, 1989) a combination of straight lines, arcs (Jiang and Seneviratne, 1999), and spirals

**Corresponding author.* e-mail: benjamas.pan@kmutt.ac.th

(Brussel and Schutter, 1991), a series of helical paths (Boer and Albada, 1993), a sinusoidal path (Paromtchik and Laugier, 1996), or a quantic polynomial curve (Xu *et al*., 2000). These path planning methods are categorized as geometric planners. On the other hand, there is another class of maneuver planning methodology based upon heuristic criteria. In heuristic planning, a parking path is established by searching for graphs formed out of straight lines via the vertices of obstacles or patches of free space. The algorithms employed in Vorashompoo *et al*. (2011), Dolgov *et al*. (2010), Stentz (1997) and Wang *et al*. (2015) are commonly designed based on a searching method which is computationally intensive for practical use.

The next stage to consider after maneuver planning is control implementation. At this stage, the planned maneuver is executed. It is generally observed that the path planning method suggests a sudden change in the steering wheel to reverse the movement direction while backing up. In fact, it is unlikely that a mechanical steering system can perform such a rapid change and perfectly track the backing maneuver. Macek *et al*. (2008), D' Andrea-Novel *et al*. (1995), Khoshnejad and Demirli (2005), Kim *et al*. (2014) and Dong and Kuhnert (2005) provided examples of tracking control strategies for path following, including sliding mode control (SMC) (Macek *et al*., 2008), state feedback linearization (D' Andrea-Novel *et al*., 1995), neural fuzzy (Khoshnejad and Demirli, 2005), model predictive control (Kim *et al*., 2014), and robust adaptive control (Dong and Kuhnert, 2005). Among these, SMC has been considered an effective scheme due to its fast response, good transient tracking, and robustness with respect to system uncertainties and disturbances. Du and Tan (2015) proposed a tracking controller using SMC by taking into account the heading angle and driving velocity. Naderi Samani *et al*. (2016) developed another tracking controller based on feedback linearization control and SMC which can overcome the drawback of sliding mode control by generating a smooth control input.

This paper focuses on a tracking algorithm for nonholonomic systems, performing a parallel parking task in the absence of obstacles, except sideways, with two parked vehicles surrounding the parking space. This parking scenario is part of normal daily life since one of the possible obstacles in a real situation is another vehicle waiting behind the parking vehicle. It seems natural to observe that the waiting vehicle will leave enough room for the parking vehicle to successfully park. This situation is therefore formulated as an obstacle-free problem. Firstly, to generate a parking path, Dubin's algorithm (Dubins, 1957), the most common method for obtaining the shortest path, is presented. The path is then followed by using classical feedback control to command the desired steering angle. Due to an imperfect steering mechanism, the parking vehicle cannot perfectly track the Dubin's path and generate tracking errors along the trajectory. ILC is then introduced to improve tracking performance by adjusting the parking maneuver repeatedly until the best performance is achieved. The ILC law proposed in this paper is in a simple form and requires low computational complexity.

The remainder of the chapter is organized as follows: Section 2 presents a problem formulation when a vehicle requires reverse parking; Section 3 introduces a path planner using Dubin's method, and illustrates experimental parking maneuvers; Section 4 develops an ILC design to improve tracking performance, and evaluating simulation and experimental results; Section 5 presents the conclusions and suggestions for future research.

2. PROBLEM FORMULATION

The parking process can be simply divided into three stages. The first stage is parking space detection in which either vision sensors or distance sensors, such as infrared or ultrasonic sensors, can be combined to identify the parking space. Once the space is detected, the vehicle stops its motion and starts a path planning algorithm. Given an initial point (the current position of the vehicle) and a target point (the final location of the vehicle after finishing the parking process), a collision-free trajectory is generated according to the parking space information. Finally, the trajectory tracking phase begins. The vehicle starts the reverse motion maneuver and tracks the designed trajectory to complete the parallel parking.

The problem considered in this paper can be stated as follows. It is assumed that there is an absence of obstacles along the parking trajectory due to the reasons given in the previous section. Given the initial position and orientation of the rear wheel midpoint of the vehicle in a constrained workspace, the shortest, smoothest, and collision-free path must be found, leading to a target position and orientation while satisfying all nonholonomic constraints. An additional objective is to design a controller that can perfectly track the designed trajectory under the imperfection of steering mechanisms with constant velocity.

3. PATH PLANNING

Calculation of the shortest path for parallel parking using Dubin's method and details of experimental parameter settings for path planning design is discussed in the following subsections.

3.1. Parking Using Dubin's Method

In this section, a parking trajectory is designed based on Dubin's method (Dubins, 1957), giving the shortest path with no obstacles. For any given initial and final configuration, the Dubin's path consists of three arcs at most (referred to as "*C*") or straight lines (referred to as "*S*") of continuously connected maneuvers. Since this problem only considers reversing into a side road on the left, the *RSL* trajectory is generated as the optimal path where *R* and *L* respectively denote an extreme right-hand

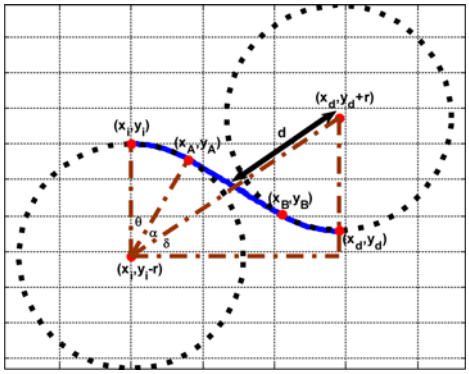

Figure 1. RSL trajectory connecting from initial point to destination.

turn and left-hand turn with a minimum turning radius r. Figure 1 shows the RSL path in the two dimensional plane given the coordinate of the initial point of the rear end on the left side of the vehicle (x_i, y_i) and (x_d, y_d) is for the destination or target point of the rear end on the left side of the vehicle.

According to Figure 1, the two circles have the same minimum turning radius of r and the distance between each center position is $2d$ where

$$d = \frac{1}{2}\sqrt{(x_i - x_d)^2 + (y_i - y_d - 2r)^2}\ .$$

The associated vertical distance and horizontal distance are $y_d - y_i + 2r$ and $x_d - x_i$, respectively. The straight line, connecting point (x_A, y_A) to point (x_B, y_B), is tangent to the circles. (x_A, y_A) is the point where the right turning arc connects to the straight line and (x_B, y_B) is similarly the point connecting the straight line to the left turning arc. The angle formed by each arc is equal to the heading angle of the vehicle with respect to the x-axis, θ, which can be calculated from the summation of the

3 angles, $\theta = \frac{\pi}{2} - \delta - \alpha$. The angles δ and α can easily be obtained from Pythagorean theorem as

$$\delta = \tan^{-1}\left(\frac{y_d - y_i + 2r}{x_d - x_i}\right) \text{ and } \alpha = \cos^{-1}\left(\frac{r}{d}\right).$$

As a result, the coordinates of (x_A, y_A) and (x_B, y_B) are as follows:

$$\left.\begin{aligned}(x_A, y_A) &= \left[x_i + r\sin(\theta), y_i - r + r\cos(\theta)\right],\\ (x_B, y_B) &= \left[x_d - r\sin(\theta), y_d + r - r\cos(\theta)\right]\end{aligned}\right\} \quad (1)$$

The total distance of the *RSL* trajectory is therefore calculated from the two arcs and straight line as:

$$D = 2r\theta + \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2} \quad (2)$$

Since the Dubin's path has been planned in advance, the vehicle has to control the front wheels to adjust its steering angle accordingly. The kinematic model of a rear wheel drive vehicle is represented as follows:

$$\left.\begin{aligned}\dot{x} &= v\cos\theta,\\ \dot{y} &= v\sin\theta,\\ \dot{\theta} &= \frac{v}{L}\tan\phi,\\ x_0 &:= x(0),\ y_0 := y(0),\ \theta_0 := \theta(0)\end{aligned}\right\} \quad (3)$$

where (x, y) denotes the coordinate of the rear wheel midpoint, L is the wheelbase of the vehicle, v is the transitional velocity of the vehicle, and ϕ gives the steering wheel's angle with respect to the vehicle's longitudinal axis. In planar motion, the vehicle is subjected to a constraint, $-\phi_{max} \le \phi \le +\phi_{max}$. In (3), the control input ϕ gives the desired trajectory to follow the designed parking path.

3.2. Experimental Parameters Used in Dubin's Method

In order to create a Dubin's path as described in the previous subsection, some parameters are required from a real parking situation. Figure 2 presents the go-kart sized vehicle used in experimental tests. It is a rear wheel driven, steer-by-wire system with a maximum turning angle of $-31.8^\circ \le \phi \le +32.1^\circ$. Note that the minus and plus signs indicate the right-hand and left-hand turns, respectively. The range of maximum turning angle is experimentally obtained from Ackermann steering geometry, i.e.,

$$\phi_{max} = \tan^{-1}\left(\frac{L}{r}\right) \text{ where } L \text{ is 108 cm.}$$

Since the maximum turning angles in both directions are slightly different, it would be preferable to set the maximum turning angle at 30° in each direction to get Dubin's path and allow some tolerance for measurement error. As a result, $-30^\circ \le \phi \le +30^\circ$ is set in Dubin's method to generate a parking path. For the sake of simplicity, let us assume that the vehicle moves along the parking trajectory with a constant velocity of 0.11 m/s.

In the parking process, it is assumed that an available parking lot is on the left and can be detected by the vehicle

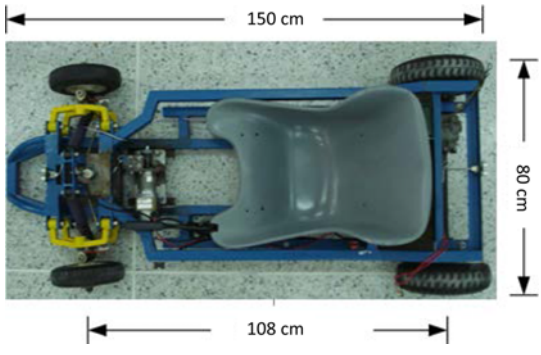

Figure 2. Go-kart sized vehicle and its dimensions.

using an infrared sensor mounted on the left rear wheel. If the vehicle finds a free space of sufficiently long length, it will stop at the initial point with the heading angle of 0° and start following the Dubin's path. Many parking control approaches recommend that the length of the parking bay is at least 1.6 times the vehicle's length in order to park successfully. Unlike those algorithms, Dubin's method needs to consider both the length and width of the vehicle to calculate the minimum length of the parking bay. According to Figure 1, to minimize the length of the parking bay, the two circles have to be externally tangent, i.e., $d = r$. This leads to the fact that x_d fully depends on y_d according to the relationship, $x_d = \sqrt{-y_d^2 - 4y_d r}$, given that the initial point (x_i, y_i) is (0, 0). Since the vehicle is 80 cm wide, the destination point to park the vehicle (x_d, y_d) can be set to (232, − 80) in order to park in the smallest parking bay with Dubin's path. However, it is very common to park a vehicle and leave some room all around it in a real parking situation. The destination point is therefore set to (290, − 120). The Dubin's path is then generated in the sense that it is the shortest distance from (0, 0) to (290, − 120), not the shortest route to enable the vehicle to be parked.

3.3. Simulation and Experimental Results

To demonstrate the designed parking maneuver, it is assumed that the parking bay is 125 cm wide and 310 cm long. The initial parking point is 5 cm away from the parking bay. Table 1 provides the Dubin's parameters to compute the path using the vehicle parameters set in the previous subsection.

Figure 3 illustrates the Dubin's path from simulation according to the Dubin's parameters in Table 1. The total distance from (0, 0) to (290, −120) is equal to 320.3346 cm. The corresponding turning angle ϕ and heading angle ϕ are shown in Figure 4. Firstly, the turning angle is set to the vehicle's maximum turn or − 30° while the vehicle is moving along the first circle, and then suddenly changes to 0° when following the straight line. The turning angle is promptly changed to 30° when touching the arc of the second circle, following it until reaching the destination point.

Table 1. Dubin's parameters.

r = 187.0615 cm
d = 96.3971 cm
θ = 0.6068 rad
δ = 0.7196 rad
α = 0.2445 rad
$x_0 = 0,\ y_0 = 0,\ \theta_0 = 0$
(x_A, y_A) = (106.6648, − 33.3910)
(x_B, y_B) = (183.3352, − 86.6090)
D = 320.3346 cm

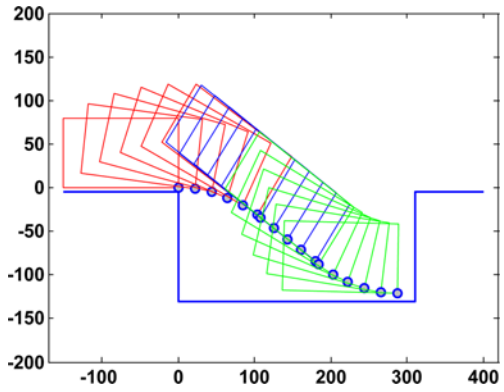

Figure 3. Reversing by Dubin's path from (0, 0) to (290, − 120).

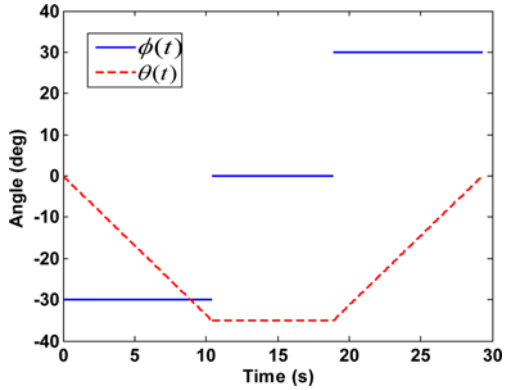

Figure 4. Desired heading and turning angles.

In the experiments, the vehicle follows the Dubin's path by tracking the desired turning angle as shown in Figure 4. A classical feedback control system was constructed with an Arduino microcontroller to control the steering angle. The PID parameters were tuned to achieve the best performance. The resulting turning angle was measured by a potentiometer-based linear sensor mounted at the end of the steering column in every 0.1 second. It was fed into the feedback control system as a signal and also used in the

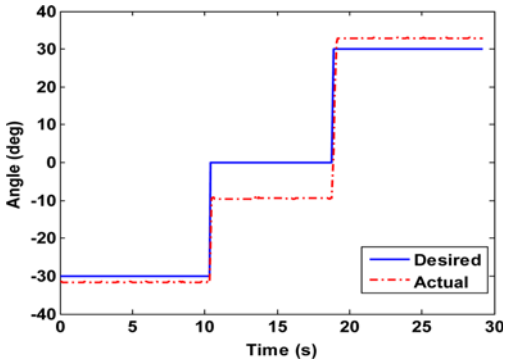

Figure 5. Desired angle compared to the actual angle.

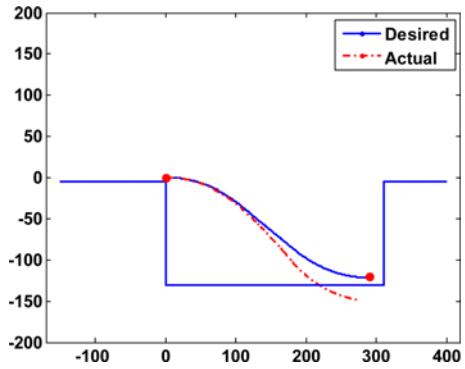

Figure 6. Paths corresponding to Figure 5.

kinematic model in (3) to calculate the corresponding coordinates (x, y). Figure 5 illustrates the tracking performance in the steering system. The solid line shows the desired steering angle in Figure 4 and the dashed line presents the measured steering angle. It can be observed that the steering angle can rapidly change during transition but leaves noticeable offset errors in each interval. Even though the offset errors can simply be improved by the integral term of the controller, a fast response is the key factor when considering this problem. The resulting path is plotted and compared with the desired Dubin's path in Figure 6. As shown, the vehicle can neither successfully park nor follow the Dubin's trajectory. The next section introduces a solution to the tracking problem using an advanced control method.

4. TRAJECTORY TRACKING USING ilc

As indicated in the previous section, the typical controller cannot do a perfect job in tracking the desired turning angle, and this leads to an unsuccessful parking maneuver. This section investigates the possibility of using ILC to track the Dubin's path.

4.1. Overview of ILC

Many applications of ILC are devoted to tracking tasks in robotic manipulators (Ngo *et al*., 2014; Delchev, 2013; Ye and Wang, 2006). Even though ILC is rarely used commercially since its structure is quite complicated and requires a lot of memory to store historic information, a simple structure of ILC is considered in this research to learn the Dubin's path. This is an innovative application for using ILC in an autonomous driving vehicle.

A block diagram of ILC is illustrated in Figure 7. At iteration number 0, there is only a feedback control system or a closed-loop steering system performing the reverse maneuver as in the previous section. The desired trajectory, $y_d(k)$ is the turning angle shown in Figure 5. The control input at zero iteration, $u_0(k)$ is exactly the same as the desired trajectory since there is no learning control law used in the initial iteration. Note that the subscript j represents the jth iteration and k is an index of time step. The corresponding output $y_0(k)$ or the actual turning angle is then recorded to calculate the tracking angle error $e_0(k)$ which is used in the next iteration. Beginning at iteration number 1, the ILC starts to be used, the ILC problem seeks to iteratively adjust the command $u_0(k)$, k = 0, 1, ..., $N-1$ from run j to run $j+1$, aiming to converge as $j \to \infty$ on that command which produces output $y(k)$ matching the desired output history, $y_d(k)$ $k = 1, 2, ..., N$. A one-time step delay going through the system is assumed. Considering a discrete time state space system as follows:

$$x_j(k+1) = Ax_j(k) + Bu_j(k) \tag{4}$$

$$y_j(k) = Cx_j(k) \tag{5}$$

with $x_j(0) = x_0$ for all j. The equivalent form can be written as:

$$y_j(k) = \underbrace{C(qI - A)^{-1}B}_{P(q)}u_j(k) + CA^k x_0 \tag{6}$$

where q is the forward time-shift operator, i.e., $qx(k) \equiv x(k+1)$. Assuming a zero initial condition, the output for every time step of any run j can be packaged as:

$$\underbrace{\begin{bmatrix} y_j(1) \\ y_j(2) \\ \vdots \\ y_j(N) \end{bmatrix}}_{\underline{y}_j} = \underbrace{\begin{bmatrix} p_1 & 0 & \cdots & 0 \\ p_2 & p_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ p_N & p_{N-1} & \cdots & p_1 \end{bmatrix}}_{P} \underbrace{\begin{bmatrix} u_j(0) \\ u_j(1) \\ \vdots \\ u_j(N-1) \end{bmatrix}}_{\underline{u}_j} \tag{7}$$

where N is the number of time steps in one iteration and the coefficients p_k are Markov parameters in which $p_1 \neq 0$ since a one-time step delay is assumed. The tracking error can be obtained from:

$$\underbrace{\begin{bmatrix} e_j(1) \\ e_j(2) \\ \vdots \\ e_j(N) \end{bmatrix}}_{\underline{e}_j} = \underbrace{\begin{bmatrix} y_d(1) \\ y_d(2) \\ \vdots \\ y_d(N) \end{bmatrix}}_{\underline{y}_d} - \underbrace{\begin{bmatrix} y_j(1) \\ y_j(2) \\ \vdots \\ y_j(N) \end{bmatrix}}_{\underline{y}_j} \tag{8}$$

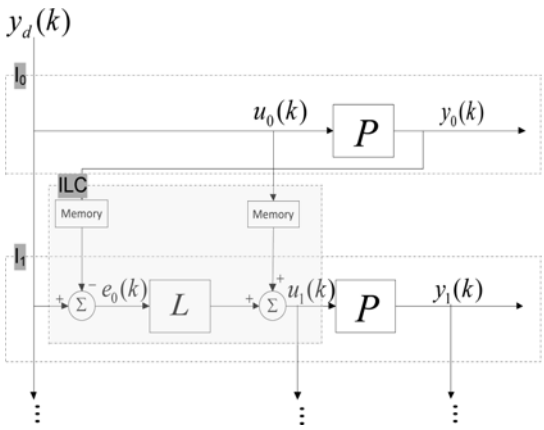

Figure 7. Block diagram of ILC at the first two iterations.

General learning control law takes the form:

$$\underline{u}_{j+1} = \underline{u}_j + L\underline{e}_j \tag{9}$$

where L is defined as a $N \times N$ matrix of learning gains l_{ij}, for row i and column j. In a typical learning matrix, all of the entries along each diagonal are identical. The next subsection discusses the stability criterion of ILC.

4.2. Stability of ILC

Considering the ILC system stability, one can write the error propagation from one iteration to the next as:

$$\underline{e}_{j+1} = (I - PL)\underline{e}_j \tag{10}$$

where I represents an identity matrix with the same dimension of the system matrix. According to (10), the error vector converges to zero for all possible initial error history vectors, if and only if, the spectral radius of $(I - PL)$ is less than unity, and for a monotonically decay in error, all singular values of the matrix $(I - PL)$ have to be less than unity. These are stability conditions for the ILC system.

4.3. Simulation and Experimental Results

According to the step response of the steering system shown in Figure 5, one can approximate the discrete time transfer function as:

$$G(z) = \frac{0.3962z - 0.3618}{z^2 - 1.155z + 0.1878} \tag{11}$$

In order to use a simple learning control system to track the Dubin's path, the learning matrix L is designed as a diagonal matrix containing only the gain of 1, i.e., $l_{ii} = 1$, along the main diagonal. The maximum spectral radius of the matrix $(I - PL)$ is 0.6038 and the corresponding maximum singular value is 0.6764, which guarantees both stability and a monotonically decay in error of the ILC system. The tracking performance is demonstrated in Figure 8.

It can be seen that the ILC can obviously improve tracking performance, especially at the change from -30^o to 0^o. Figure 9 presents the root mean square (RMS) error of the tracking in each iteration. The RMS error decreases from 5.6479 to 4.4198 after applying ILC and has a subtle variation as the number of iterations progresses.

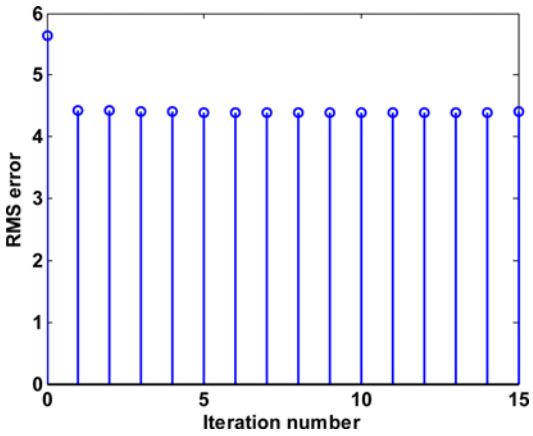

Figure 9. RMS error of tracking angle in each iteration simulation.

Figures 10 and 11 provide analogous experimental results in which the tracking performance improves when applying the ILC. Since the steering system can perfectly perform a sudden change in angle as desired and only the steady-state error seems to be an issue in the real application, ILC can almost completely eliminate the steady-state error as seen in iteration 1 – 5. In the experiment, the vehicle takes 29.3 seconds to complete a command trajectory. By using a sample time of 0.1 second, the microcontroller (Arduino) needs to memorize 293 historical control inputs and 293 historical errors to update the control input for the next run according to the control law (9). Since the simple control law does not require a large amount of data to update the control input, it takes approximately 39 KB to compile the code used in each run. Figure 11 compares the RMS errors obtained from an original PID system, or steering system, and the ILC system. It is obvious that the PID system produces a similar tracking error in each iteration. ILC can successfully learn from the history and

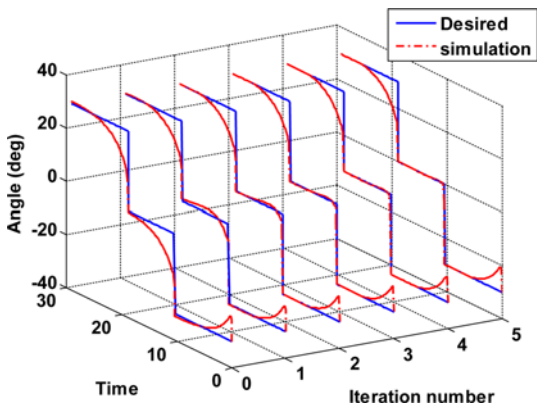

Figure 8. Tracking angle in the first six iterations.

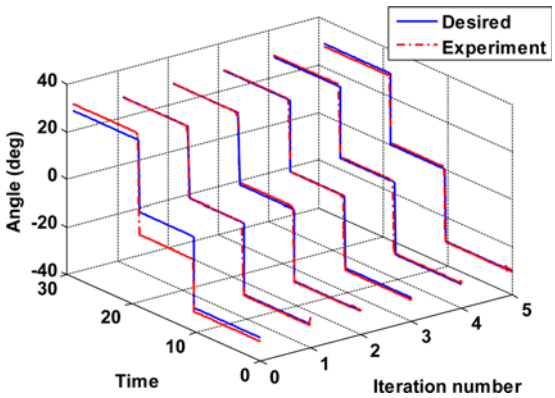

Figure 10. Tracking angle in the first six iterations.

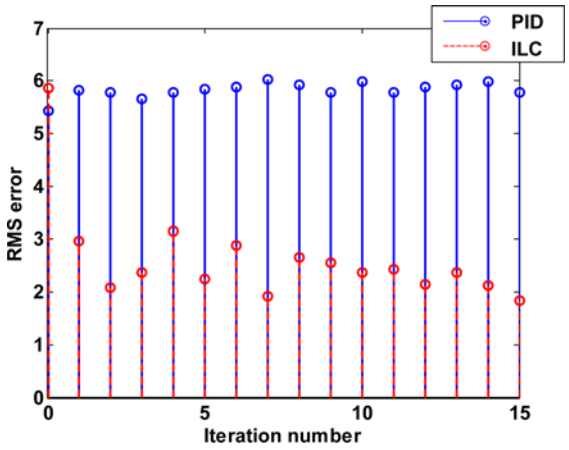

Figure 11. RMS error of tracking angle in each iteration experiment.

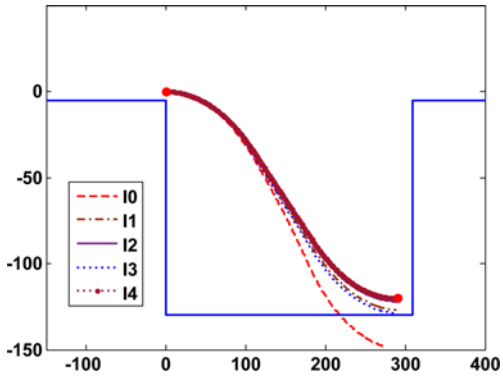

Figure 12. Learning path in the first five iterations.

Figure 13. Snapshots of the learning path.

reduce the tracking errors in each iteration. Unlike the simulation results in Figure 9, where the RMS error stays still as the number of iterations progresses, the experimental results swing back and forth due to the imperfectly repeated error in each run. However, these results confirm that ILC can track the Dubin's path better than the conventional approach.

Figure 12 illustrates the learning trajectory in each iteration, showing that the vehicle can succeed in parking by following the desired Dubin's path with ILC. Figure 13 demonstrates the snapshots of an effective learning path.

5. CONCLUSION

The primary aim of this study is to develop a reverse parking maneuver for a vehicle by specifying an initial point and a target point. The classical method is proposed to find the shortest route between the given points. The second aim is to investigate whether a typical feedback control system used in the vehicle steering system can perfectly follow the desired path. A control structure using ILC has been developed to improve the tracking performance. Stability of the ILC system has been investigated and confirmed by simulation results. An experimental study evaluated the approach and demonstrated that by using a simple learning control law, the tracking performance substantially improved. It is seen from the results that ILC can improve the performance of a typical closed-loop control system by learning from historical errors in previous runs. Therefore, some learning iterations are required to reach the user's level of satisfaction. However, the repeated learning experiments can be done during the manufacturing process before launching the product since it learns from a specific trajectory or Dubin's path. Once the error level is satisfactory to the user, one can stop the learning process and make use of the current trajectory command as a reference to park the vehicle. Future research will focus on a more complicated design of ILC, based on a system model which can perfectly track the desired path.

REFERENCES

Boer, G. A. D. and Albada, G. D. V. (1993). The MARIE autonomous mobile robots. *Proc. Conf. Intelligent Autonomous Systems - IAS 3*, 164–173.

Brussel, H. V. and Schutter, J. D. (1991). Hierarchical control of free-navigation AGVs. *Proc. Int. Workshop on Information Processing in Autonomous Mobile Robots*, 105–119.

Chen, C.-Y. and Feng, H.-M. (2009). Hybrid intelligent vision-based car-like vehicle backing systems design. *Expert Systems with Applications* **36**, **4**, 7500–7509.

D'Andrea-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.

Daxwanger, W. A. and Schmidt, G. K. (1995). Skill-based visual parking control using neural and fuzzy networks.

Proc. IEEE Int. Conf. System, Cybernetics, **2**, 1659–1664.

Delchev, K. (2013). Iterative learning control for robotic manipulators: A bounded-error algorithm. *Int. J. Adaptive Control and Signal Processing* **28**, **12**, 1454–1473.

Dolgov, D., Thrun, S., Montemerlo, M. and Diebel, J. (2010). Path planning for autonomous vehicles in unknown semi-structured environments. *Int. J. Robotics Research* **29**, **5**, 485–501.

Dong, W. and Kuhnert, K.-D. (2005). Robust adaptive control of nonholonomic mobile robot with parameter and nonparameter uncertainties. *IEEE Trans. Robotics* **21**, **2**, 261–266.

Du, X. and Tan, K. K. (2015). Autonomous reverse parking system based on robust path generation and improved sliding mode control. *IEEE Trans. Intelligent Transportation Systems* **16**, **3**, 1225–1237.

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

Heinen, M. R., Osorio, F. S., Heinen, F. J. and Kelber, C. (2006). SEVA3D: Using articial neural networks to autonomous vehicle parking control. *Neural Networks, IJCNN Int. Joint Conf., IEEE*, 4704–4711.

Jenkins, R. E. and Yuhas, B. P. (1993). A simplified neural network solution through problem decomposition: The case of the truck backer-upper. *IEEE Trans. Neural Network* **4**, **4**, 718–720.

Jiang, K. and Seneviratne, L. D. (1999). A sensor guided autonomous parking system for nonholonomic mobile robots. *Proc. IEEE Int. Conf. Robotics and Automation*, 311–316.

Kanayama, Y. and Hartman, B. I. (1989). Smooth local path planning for autonomous vehicles. *Proc. IEEE Int. Conf. Robotics and Automation*, 1265–1270.

Khoshnejad, M. and Demirli, K. (2005). Autonomous parallel parking of a car-like mobile robot by a neuro-fuzzy behavior-based controller. *Annual Meeting of the North American Fuzzy Information Processing Society*, 814–819.

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.

Kong, S. G. and Kosko, B. (1990). Comparison of fuzzy and neural truck backer-upper control. *IJCNN Int. Joint Conf. Neural Networks*, 349–358.

Lee, C.-K., Lin, C.-L. and Shiu, B.-M. (2009). Autonomous vehicle parking using hybrid artificial intelligent approach. *J. Intelligent and Robotic Systems* **56**, **3**, 319–343.

Li, T.-H. S., Lee, M.-H., Lin, C.-W., Liou, G.-H. and Chen, W.-C. (2016). Design of autonomous and manual driving system for 4WIS4WID vehicle. *IEEE Access*, **4**, 2256–2271.

Macek, K., Philippsen, R. and Siegwart, R. (2008). Path following for autonomous vehicle navigation with inherent safety and dynamics margin. *IEEE Intelligent Vehicles Symp*., 108–113.

Naderi Samani, N., Ghaisari, J. and Danesh, M. (2016). Parallel parking of a car-like mobile robot based on the P-domain path tracking controllers. *IET Control Theory & Applications* **10**, **5**, 564–572.

Ngo, T., Nguyen, M. H., Wang, Y., Ge, J., Wei, S. and Mai, T. L. (2014). An adaptive iterative learning control for robot manipulator in task space. *Int. J. Computers, Communications & Control (IJCCC)* **7**, **3**, 518–529.

Oentaryo, R. J. and Pasquier, M. (2004). Self-trained automated parking system. *8th Control, Automation, Robotics and Vision Conf*., 1005–1010.

Panomruttanarug, B. and Higuchi, K. (2010). Fuzzy logic based autonomous parallel parking system with kalman filtering. *SICE J. Control, Measurement, and System Integration* **3**, **4**, 266–271.

Paromtchik, I. E. and Laugier, C. (1996). Motion generation and control for parking an autonomous vehicle. *Proc. IEEE Int. Conf. Robotics and Automation*, 3117–3122.

Stentz, A. (1997). Optimal and efficient path planning for partially known environments. *Intelligent Unmanned Ground Vehicles*, 203–220.

Vorashompoo, A., Panomruttanarug, B. and Higuchi, K. (2011). Bidirectional best first based autonomous parallel parking system. *8th Electrical Engineering Electronics, Computer, Telecommunications and Information Technology*, 593–596.

Vorobieva, H., Glaser, S., Minoiu-Enache, N. and Mammar, S. (2015). Automatic parallel parking in tiny spots: Path planning and control. *IEEE Trans. Intelligent Transportation Systems* **16**, **1**, 396–410.

Wang, Q., Wulfmeier, M. and Wagner, B. (2015). Voronoi-based heuristic for nonholonomic search-based path planning. *Advances in Intelligent Systems and Computing*, 445–458.

Wang, W., Song, Y., Zhang, J. and Deng, H. (2014). Automatic parking of vehicles: A review of literatures. *Int. J. Automotive Technology* **15**, **6**, 967–978.

Xu, J., Chen, G. and Xie, M. (2000). Vision-guided automatic parking for smart car. *Proc. IEEE Int. Vehicles Symp*., 725–730.

Ye, Y. and Wang, D. (2006). Implementation of ILC batch update using a robotic experimental setup. *Microprocessors and Microsystems* **30**, **5**, 259–267.

Zhao, Y., Ding, F., Guo, L. and Yuan, Y. (2016). Navigation controller design using fuzzy logic theory for vehicle parallel automatic parking. *J. Balkan Tribological Association* **22**, **2**, 1289–1298.

APPENDIX

The discrete time state space system of the vehicle and its Markov parameters are provided as:

$$A = \begin{bmatrix} 1.1547 & -0.1878 \\ 1.0000 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 0.3962 & -0.3618 \end{bmatrix}$$

$$P = \begin{bmatrix} 0.3962 & & & \\ 0.0957 & 0.3962 & & \\ \vdots & \ddots & \ddots & \\ 1.0870\times10^{-7} & \cdots & 0.0957 & 0.3962 \end{bmatrix}_{293\times293}$$

Note that the elements in the lower triangle are the same along its diagonal line and the value gets closer and closer to zero when reaching the bottom left corner.

The learning control matrix is given as:

$$L = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix}_{293\times293}$$