Kinematic and Dynamic Controller Design for Autonomous Driving of Car-like Mobile Robot

Kiwon Yeom
Department of Human Intelligence Robot Engineering, Sangmyung University, South Korea
Email: pragman@naver.com

***Abstract*—Autonomous driving technologies are taken into account for numerous applications such as spraying agricultural chemicals, mowing golf course grass, unmanned military operations and commercial applications such as self-driving cars. This paper presents a kinematic and dynamic controller design of autonomous mobile robot for a car like vehicle to self-drive using GPS. In this paper, a simple kinematic bicycle model is introduced and Model Predictive Controller (MPC) is derived for controlling the mobile robot. Computational simulation results show that the robot can successfully navigate and drive toward a final destination reacting to the changes in the environment. Finally this study presents experimental results to show the effectiveness of the proposed controller.**

***Index Terms*—Autonomous driving, self-driving, kinematic control, model predictive controller, mobile robot**

I. Introduction

Autonomous self-driving vehicles have grown dramatically for various applications in past several years due to remarkable advances in computing and sensing technologies. Recently, the autonomous car-like robot called mobile robot is tried to apply to the agricultural applications such as spraying agricultural chemicals, crop distribution, and picking fruits off trees [1] and [2]. Another example is military operations with unmanned manner to observe the opposites. However, recent competitions in autonomous vehicles, most of all, have accelerated the research in commercial self-driving cars. These application fields of self-driving vehicle require a controller to navigate along a generated path avoiding obstacles and rejecting disturbance [3], [4], and [5].

The core technologies of a modern autonomous vehicle are localization and orientation of the vehicle including location and pose, detection/perception of the lane and objects such as cars, humans and other objects on the street, and control of speed and steering. This paper will discuss the control of the vehicle's localization, orientation, speed and steering using and a kinematic model of steerable mobile robot and Model Predictive Control (MPC).

MPC based controller designs have been successfully applied to autonomous driving system [7] and [8]. In fact, MPC is very suited for this kind of dynamic systems since it can handle multiple inputs/outputs. In addition, Through the MPC based model, the design of path generation or planning and path-following can be simplified.

In general, most MPC control schemes use dynamic vehicle models [9] and [10]. However, this approach has two disadvantages: it is computationally expensive and tire model becomes singular when a vehicle's speed is low. These phenomena make it difficult to use the same controller model of the vehicle for stop-and-go scenarios, which is very common in urban driving with low speed of vehicle.

To sum up, a great number of techniques have been applied for autonomous vehicle control. However, these techniques either do not consider the physical constraints or just consider some specific constraint implicitly.

This paper proposes a hybrid approach to address both disadvantages by using a kinematic bicycle model and MPC controller. This hybrid controller makes it possible for the mobile robot to explicitly handle the physical constraint and optimize the trajectories in unknown environment.

In this paper, mobile robot's navigation involves the identification of the mobile robot's location and orientation (heading). In addition, the mobile robot has to determine the location of the desired final destination. A series of sequential waypoints may be helpful for the robot to follow a pre-defined course to its final destination.

The coordinate data from GPS, latitude and longitude, is combined with the compass angle from an electronic compass sensor. From the GPS data the heading, bearing and distance can be easily achieved and those information are used for path planning.

In order to demonstrate the effectiveness of the proposed approach, in this paper, a computer simulation was conducted and a small scale autonomous car-like mobile robot was implemented and tested for the purpose of comparison.

The robot is equipped with several sensors onboard to provide location, orientation, and speed data to the microprocessor. Then, the proposed controller model is to produce the appropriate velocity and steering angle for the mobile robot to successfully reach the target location.

Experimental results and analysis show that the proposed controller model provides satisfactory control performance in a computer simulation and a wide range of experiments using a car-like mobile robot.

Manuscript received July 7, 2019; revised May 11, 2020.

doi: 10.18178/ijmerr.9.7.1058-1064

The paper is organized as follows. In Section II, an overview of the kinematic bicycle model and dynamic bicycle model are discussed. These two models are compared with computational performance in Section III. Section IV describes the proposed MPC controller design. Section V discusses the experimental results with comparison of a computer simulation and implemented a small car-like mobile robot. In Section VI, this paper draws a conclusion and suggests the future work.

II. Kinematic and Dynamic Momels of Mobile Robots

In this subsection, a general mobile robot model is introduced and more readings should be referred in [11].

A. Heading for Mobile Robot Navigation

In this study, the heading angle of the mobile robot specifies the orientation of the mobile robot when the mobile robot moves. Φ_h is the heading angle, which is the angle between the mobile robot's forward velocity vector (X_r) and a directed vector (T_N) from the center of the mobile robot to the true North Pole. In this paper, True north provides the reference coordinate for the mobile robot's heading angle. In other words, if the mobile robot's velocity vector coincides with True north then the heading angle is zero (Φ_h = 0). The convention of the heading angle is defined as follows:

$$-\pi \leq \Phi_h \leq \pi \tag{1}$$

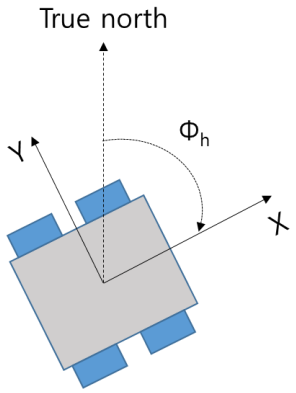

Figure 1. Heading angle in the coordinate frame of the mobile robot.

B. Bearing for Mobile Robot Navigation

The bearing angle is defined as the angle between T_N and a directed vector (G_P) along the goal position from the center of the mobile robot to the true North Pole. Like the heading angle, true north act as the reference for the bearing angles. The convention of the bearing angle is defined as follows:

$$-\pi \leq \Phi_b \leq \pi \tag{2}$$

C. Metrics of Distance and Bearing

This study assumes that there is no obstacle on the mobile robot's path and no disturbance rejection is taken into account. These two issues will be considered for the future projects. To calculate the distance from the start point to the final goal point which the mobile robot must traverse, this study employs latitude and longitude data from a commercial GPS sensor. Since the Earth is slightly ellipsoidal and most navigation formulas was developed on the basis of a spherical surface of the Earth, this study also uses the great-circle method to calculate the distance between two points.

The formula uses the haversine formula to calculate the great circle distance between two points. That is, the shortest distance over the earth's surface and ignoring any obstacles and hills.

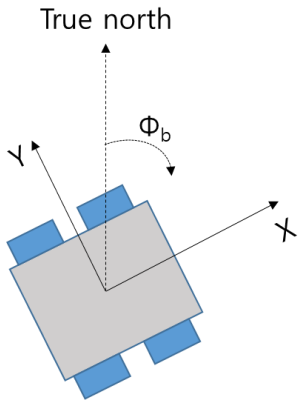

Figure 2. Bearing angle in the coordinate frame of the mobile robot.

The haversine is given by

$$a = sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos\varphi_1 \times \cos\varphi_2 sin^2\left(\frac{\Delta\lambda}{2}\right) \tag{3}$$

$$c = 2 \times \tan^{-1}\left(\frac{\sqrt{a}}{\sqrt{1-a}}\right) \tag{4}$$

$$Distance = R \times c \tag{5}$$

where φ is latitude, λ is longitude, R is the Earth's radius (≈ 6731 km).

In general, the initial heading will vary as the mobile robot follows a path which is generated by great circle method. Therefore, it is required to take the mobile robot from the start point to the end point if followed in a straight line along the given great circle arc.

$$\Phi_b = \tan^{-1}\left(\frac{\sin\Delta\varphi \times \cos\lambda_2}{\cos\varphi_1 \times \sin\lambda_2 - \sin\lambda_1 \times \cos\lambda_2 \times \cos\Delta\varphi}\right) \tag{6}$$

D. Kinematic Bicycle Model

In this subsection, a general kinematic bicycle model is introduced and more readings should be referred in [12].

A commonly used kinematic model with the low-speed of a four-wheeled car-like mobile robots is the kinematic bicycle model as shown in Fig. 3. The mechanism of steering and movement is similar to a general bicycle structure. Therefore, a four-wheeled vehicle model can be approximated to the bicycle model for the purpose of simplicity.

The bicycle has a rear wheel fixed to the chassis and rotated back and forth with no rotation of roll, pitch and yaw. The plane of the front wheel rotates about the

vertical axis from the ground to steer the vehicle. This study assumes that the wheel rolls without slipping sideways.

The pose of the mobile robot is represented by its body coordinate frame {B} as shown in Fig. 3, with its *x*-axis in the mobile robot's forward direction and its origin at the center of the rear axle. The configuration of the robot is represented by the generalized coordinates as follows

$$q = (x, y, \theta) \in C \tag{7}$$

where $C \subset R^2 \times S^1$, and C means configuration space of the mobile robot and S^1 means unit circle with a set of angles $[0, 2\pi)$.

The dashed lines show the direction along which the wheels cannot move, the lines of no motion, and these intersect at a point known as the Instantaneous Center of Rotation (ICR). The reference point of the vehicle thus follows a circular path and its angular velocity is

$$\dot{\theta} = \frac{v}{R_B} \tag{8}$$

and by simple geometry the turning radius is $R_B = L / \tan \gamma$ where L is the length of the vehicle or wheel base. As expected, the turning circle increases with vehicle length. The steering angle γ is typically limited mechanically and its maximum value dictates the minimum value of R_B.

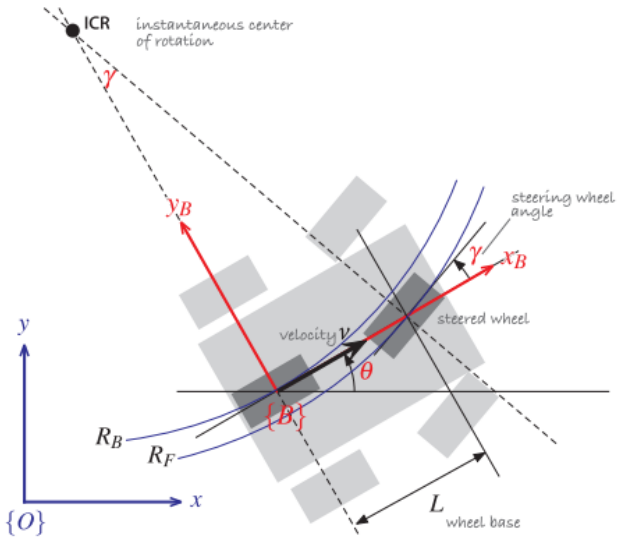

Figure 3. Bicycle model of a car (figure curtesy of [12]) .

Above kinematic bicycle model can be rewritten using the nonlinear continuous time equations and it is simply drowned in Fig. 4 and more detail contents can be referred in [11].

The dynamic bicycle model is as follows

$$\dot{x} = v\cos(\psi + \beta) \tag{9}$$

$$\dot{y} = v\sin(\psi + \beta) \tag{10}$$

$$\dot{\psi} = \frac{v}{l_r}\sin(\beta) \tag{11}$$

$$\dot{v} = a \tag{12}$$

$$\beta = \tan^{-1}\left(\frac{l_r}{l_f + l_r}\tan(\delta_f)\right) \tag{13}$$

where x and y are the coordinates of the center of mass in an inertial frame (X, Y). ψ is the inertial heading and v is the speed of the vehicle. l_f and l_r represent the distance from the center of the mass of the vehicle to the front and rear axles, respectively. β is the angle of the current velocity of the center of mass with respect to the longitudinal axis of the car. a is the acceleration of the center of mass in the same direction as the velocity.

The control inputs are the front and rear steering angles, namely, δ_f , and a. Since in most vehicles the rear wheels cannot be steered, namely $\delta_r = 0$.

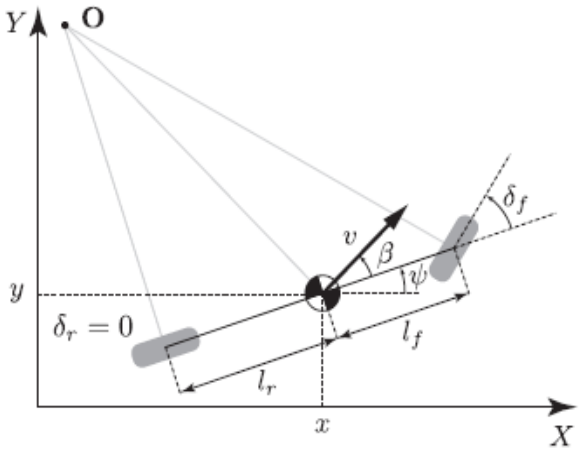

Figure 4. Kinematic bicycle model for the representation of nonlinear continuous time equation formulas.

Compared to other vehicle models [12], it is easier to identify the system on the kinematic bicycle model since there are only two control parameters, l_f and l_r. This makes it much simpler to apply the same controller design with minor modification to path planner or other vehicles with differently sized wheelbases.

E. Dyanmic Bicycle Model

For the dynamic bicycle model, it should be considered in terms of the inertia. The position coordinates and heading angle with respect to the inertia in the dynamic bicycle model are defined in the same manner as those in the kinematic bicycle model (refer [11], for more specific descriptions).

The differential equations are as follows

$$\ddot{x} = \dot{\psi}\dot{y} + a_x \tag{14}$$

$$\ddot{y} = -\dot{\psi}\dot{x} + \frac{2}{m}\left(F_{c,f}\cos\delta_f + F_{c,r}\right) \tag{15}$$

$$\ddot{\psi} = \frac{2}{I_z}\left(l_f F_{c,f} - l_r F_{c,r}\right) \tag{16}$$

$$\dot{X} = \dot{x}\cos\psi - \dot{y}\sin\psi \tag{17}$$

$$\dot{Y} = \dot{x}\sin\psi + \dot{y}\cos\psi, \tag{18}$$

where $\dot{x}$ and $\dot{y}$ denote the longitudinal and lateral speeds in the body frame, respectively and $\dot{\psi}$ denotes the yaw rate. m and I_z denote the vehicle's mass and yaw inertia, respectively. $F_{c,f}$ and $F_{c,r}$ denote the lateral tire forces at

the front and rear wheels, respectively, in coordinate frames aligned with the wheels.

For the linear tire model, $F_{c,i}$ is defined by

$$F_{c,i} = -C_{\alpha_i}\alpha_i$$

where $i \in \{f, r\}$, α_i is the tire slip angle and $C\alpha_i$ is the tire cornering stiffness.

III. Model Predictive Controller Design for Path Tracking Mobile Robots

In Model Predictive Control (MPC) architecture, the control behavior is obtained by, at each sampling time step (i.e., instant), solving an optimal control problem from multiple inputs. The optimal solution for the control can be achieved on a finite horizon for which the initial state is the current state of the plant [13].

In MPC a plant model is used to predict the future output for a certain number of time steps (namely the prediction horizon). The objective function of the control system is minimized with respect to the future inputs through the predictions.

The MPC based control problem results in a kind of receding optimization problem. At each time step, the controller model computes control strategy by solving an open-loop optimization problem for the prediction horizon. Secondly, the first strategy (or values) computed is applied to the system and the state of the system is updated. At the next time step, the controller takes the updated system state with shifted prediction horizon and re-computes control values. This process is repeatedly performed until the system satisfies the objective function [14].

A. MPC Control Architecture

This study uses a simplified MPC based control architecture as shown in Fig. 5.

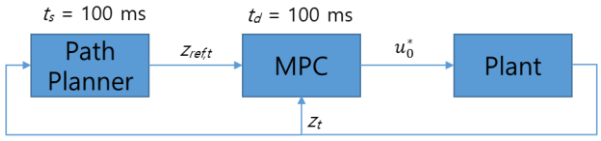

Figure 5. Simplified control architecture. The controller and the path planner are run at every 100 ms.

In MPC, the target system to be controlled is usually described by an ordinary differential equation (ODE). In general the control is piecewise constant, accordingly the system can be designed using the difference equation as follows

$$\boldsymbol{z}(k+1) = f(\boldsymbol{z}(k), \boldsymbol{u}(k)) \tag{19}$$

where $\boldsymbol{z}(k) \in R^n{}_z$ is the state vector, and $u(k) \in R^n{}_u$ is the control input vector (for more detail, refer [12] and [13]). This model is used to predict the behavior or find the optimal control input values.

As described before, since the MPC is based on solving an optimization problem during each time step (i.e., sampling period), the optimization problem is generally formulated as follows (see eq. 20~23).

Φ is the objective function to be minimized (eq. 20), the initial condition $\boldsymbol{z}_0$ is the state measured at the current instant (eq. 21). Eq. 22 represents the model used for system behavior prediction, and eq. 23 represents constraints on the control inputs [12].

$$\min_{\boldsymbol{u}(k)} \quad \boldsymbol{\Phi}(k) \tag{20}$$

$$\text{s. t.} \quad \boldsymbol{z}_0 = \boldsymbol{z}(k) \tag{21}$$

$$\boldsymbol{z}(k+1) = f(\boldsymbol{z}(k), \boldsymbol{u}(k)) \tag{22}$$

$$\boldsymbol{u}(k) \in \mathbb{U} \tag{23}$$

Minimizing the objective function, with control horizon H_c, will result in the optimal control sequence $u*(k) = \{u*(0), u*(1), \ldots, u*(H_c-1)\}$, where the first control $u*(0)$ is applied to the system.

The control objective is usually to steer the states to the origin, or to steer it to reference states z^r, often referred to as path tracking [13].

However, in this study, since the mobile robot should follow a prescribed path, the equation formula from eq. 20~23 should be slightly modified an error model.

B. Modified MPC Control Model for Path Tracking

As introduced in [12], the path planning step must generate a reference path to be followed by the car-like mobile robot. In terms of an autonomous driving or self-driving, the path should minimize the driving time, provide the shortest distance, or maximize the progress over a given time. In addition, it must have the constraint that the mobile robot should not go off-road.

In this study, the mobile robot's motion primitives are consisted of set constant speed (30 cm/s) and constant steering angles.

In order to create the reference path ($\boldsymbol{z}^r(k)$, i), the car's current position is projected onto the reference track/centerline of the street, and the reference path is generated from this position to follow the track. An illustration is shown in Fig. 8. The reference path is placed ahead of the vehicle. This controller steers the car towards the generated optimal trajectory and tries to minimize the deviation from it.

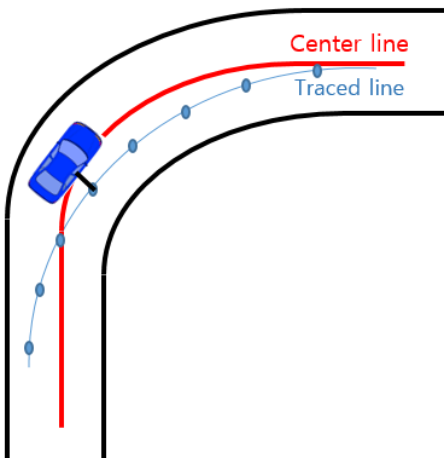

Figure 6. Path tracking errors Illustration.

In this paper, a modified model which was suggested by [13] is recalled and simply re-used.

According to [12], the modified model can be expressed in discrete time as follows.

$$\tilde{\boldsymbol{z}}(k+1) = \boldsymbol{A}(k)\tilde{\boldsymbol{z}}(k) + \boldsymbol{B}(k)\tilde{\boldsymbol{u}}(k) \tag{24}$$

where the error between the states $\boldsymbol{z}(k)$ and the state references $\boldsymbol{z}^r(k)$, is represented by $\tilde{\boldsymbol{z}}(k) = \boldsymbol{z}(k) - \boldsymbol{z}^r(k)$, and $\tilde{\boldsymbol{u}}(k) = \boldsymbol{u}(k) - \boldsymbol{u}^r(k)$ represents the error between the control inputs $\boldsymbol{u}(k)$ and the control input references $\boldsymbol{u}^r(k)$.

$$\boldsymbol{A}(k) = \left.\frac{\partial f(\boldsymbol{z}(k),\boldsymbol{u}(k))}{\partial \boldsymbol{z}(k)}\right|_{\substack{\boldsymbol{z}(k)=\boldsymbol{z}^r(k)\\ \boldsymbol{u}(k)=\boldsymbol{u}^r(k)}} \tag{25}$$

$$\boldsymbol{B}(k) = \left.\frac{\partial f(\boldsymbol{z}(k),\boldsymbol{u}(k))}{\partial \boldsymbol{u}(k)}\right|_{\substack{\boldsymbol{z}(k)=\boldsymbol{z}^r(k)\\ \boldsymbol{u}(k)=\boldsymbol{u}^r(k)}} \tag{26}$$

Based on the above modified formulas the objective function is also re-formulated so that the deviation from reference states and control inputs is penalized.

$$\tilde{\boldsymbol{\Phi}}(k) = \sum_{i=1}^{H_p-1} \tilde{\boldsymbol{z}}^{\intercal}(k+i)\boldsymbol{Q}\tilde{\boldsymbol{z}}(k+i) + \tilde{\boldsymbol{z}}^{\intercal}(k+H_p)\boldsymbol{P}\tilde{\boldsymbol{z}}(k+H_p) + \sum_{i=0}^{H_c-1} \tilde{\boldsymbol{u}}^{\intercal}(k+i)\boldsymbol{R}\tilde{\boldsymbol{u}}(k+i) \tag{27}$$

where H_p and H_c is the prediction horizon and control horizon, respectively. Furthermore, $H_c < H_p$, and the control input is assumed constant for all $i \geq H_c$ [16]. $\boldsymbol{Q}$ is the state penalization matrix, $\boldsymbol{P}$ is the terminal state penalization matrix, and $\boldsymbol{R}$ is the control input penalization matrix. These are positive definite [17].

The control input constraints also have to be modified. With $\boldsymbol{u}_{min}$ as the minimum control input and $\boldsymbol{u}_{max}$ as the maximum control input, the constraints can be expressed as follows

$$\tilde{\boldsymbol{u}}_{min}(k) = \boldsymbol{u}_{min} - \boldsymbol{u}^r(k) \tag{28}$$

$$\tilde{\boldsymbol{u}}_{max}(k) = \boldsymbol{u}_{max} - \boldsymbol{u}^r(k) \tag{29}$$

The rate at which the control inputs are allowed to change can also be constrained. This is done by introducing

$$\Delta\tilde{\boldsymbol{u}}(k) = \tilde{\boldsymbol{u}}(k) - \tilde{\boldsymbol{u}}(k-1), \tag{30}$$

$$\Delta\tilde{\boldsymbol{u}}_{min}(k) = \Delta\boldsymbol{u}_{min} - (\boldsymbol{u}^r(k) - \boldsymbol{u}^r(k-1)) \tag{31}$$

$$\Delta\tilde{\boldsymbol{u}}_{max}(k) = \Delta\boldsymbol{u}_{max} - (\boldsymbol{u}^r(k) - \boldsymbol{u}^r(k-1)) \tag{32}$$

where $\Delta\boldsymbol{u}_{min}$and $\Delta\boldsymbol{u}_{max}$ are the lower and upper bounds, respectively. Therefore, the control input range can be re-formulated as follows

$$\tilde{\boldsymbol{u}}_{min} \leq \tilde{\boldsymbol{u}}(k) \leq \tilde{\boldsymbol{u}}_{max} \tag{33}$$

$$\Delta\tilde{\boldsymbol{u}}_{min} \leq \Delta\tilde{\boldsymbol{u}}(k) \leq \Delta\tilde{\boldsymbol{u}}_{max} \tag{34}$$

With these modifications, the problem can now be

$$\min_{\tilde{\boldsymbol{u}}(k)} \quad \tilde{\boldsymbol{\Phi}}(k) \tag{35}$$

$$\text{s. t.} \quad \tilde{\boldsymbol{z}}_0 = \tilde{\boldsymbol{z}}(k) \tag{36}$$

$$\tilde{\boldsymbol{z}}(k+i+1) = \boldsymbol{A}(k+i)\tilde{\boldsymbol{z}}(k+i) + \boldsymbol{B}(k+j)\tilde{\boldsymbol{u}}(k+j) \tag{37}$$

$$\tilde{\boldsymbol{u}}_{min} \leq \tilde{\boldsymbol{u}}(k+j) \leq \tilde{\boldsymbol{u}}_{max} \tag{38}$$

$$\Delta\tilde{\boldsymbol{u}}_{min} \leq \Delta\tilde{\boldsymbol{u}}(k+j) \leq \Delta\tilde{\boldsymbol{u}}_{max} \tag{39}$$

IV. Vehicle Implementation

In this section, the hardware platform of the implemented car-like mobile robot is introduced.

A. The Car-like Mobile Robot Platform

The mobile robot chassis employed a car-like design with 4 wheels. As described in Section II, rear wheels are fixed to the body and rotate only back and forth without pitch, roll and yaw. For direction control, the simplified Ackerman steering geometry of the front wheels was employed and the front wheels are forced into parallel alignment at all times by an axle.

The axle pivots about its center on a vertical steering rod, which is connected to a steering servo-motor through a rack and pinion system. The steering servomotor is controlled by the microprocessor (Raspberry Pi 3 with Quad Core 1.2GHz) using a pulse-width modulation (PWM) signal and is capable of steering angles of -45 °≤ δ ≤ 45°.

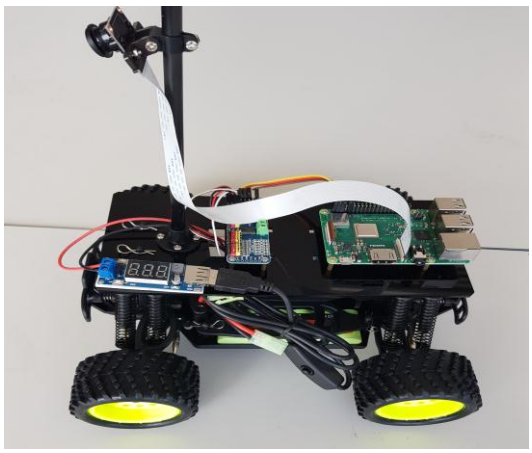

Figure 7. The autonomous car-like mobile robot platform.

For actuating, a BLDC motor was employed and was connected to the rear axle with a gear system. The gearing system provides a gain in rotational velocity to the rear axle to increase the linear velocity of the robot.

The DC motor is also controlled by the microprocessor (Raspberry Pi 3 with Quad Core 1.2GHz) using a PWM signal with a duty cycle of 0-100% corresponding with 0% when the motor is stopped to 100% when the motor is at its maximum speed which is approximately 55 cm/s.

B. Hardware Implementation for Mobile Robot Control

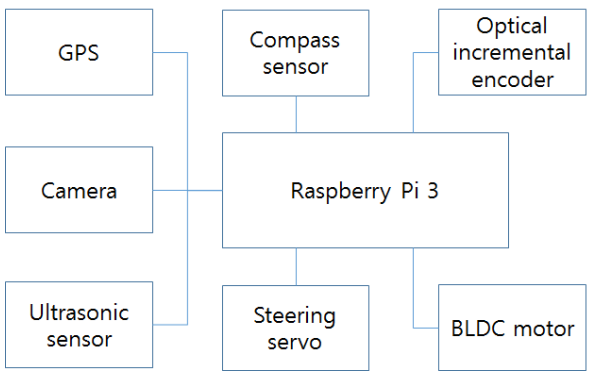

Figure 8. Autonomous car-like robot hardware interface.

The robot is equipped with several sensors to measure all state variables. These sensors include a global positioning system (GPS) receiver, a digital compass mounted with its reference along the front of robot, and an optical incremental encoder mounted on the drive shaft.

The GPS and the digital compass are used to derive the state variables required for location, heading and steering angles. The incremental encoder has been calibrated to measure the instantaneous linear velocity and the distance.

An ultrasonic sensor was mounted on the front of the vehicle to stop if the robot encounters an obstacle within 30 cm. For this project, the purpose of the ultrasonic sensor is simply to prevent damage to the robot. They are intended for future work where they may be used to implement an active obstacle avoidance feature.

A camera was mounted on the top of the front pole to calculate the variables for lane keeping during driving. However, in this study it was not used and will be used for the future work.

V. Experiment Results

The experiment results for the mobile robot are presented in Fig. 9. In this experiment, the initial conditions for ρ, β were set as 100 and π radians, respectively. α was intently increased by $\pi/4$ within range from $-\pi$ radians to $+\pi$ radians. The robot started from the start point and stop at the goal position. As shown in Fig. 9, when the value of α was increased, the path length traveled by the mobile robot was increased and some errors are observed at the early stage. In addition, in this experiment, since the path tracking was not severely considered, the related work should be handled in the future work to improve the path tracking errors. In this experimental setup, θ was set as 0 (i.e., $\theta=0$) , all of the state variables will be zero ($\rho=0$ feet, $\alpha=0$ radians, and $\beta=\pi$) when the robot has reached the goal position and orientation.

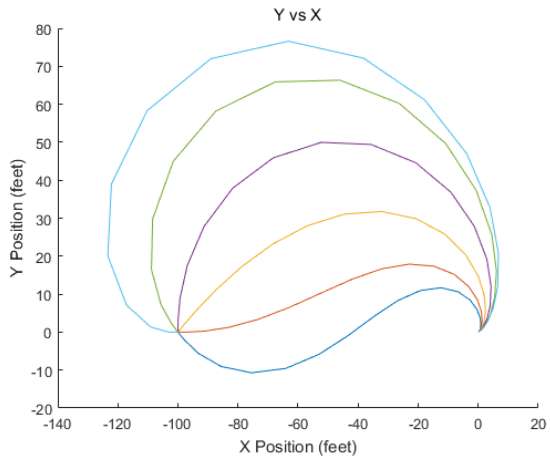

Figure 9. Experiment results of the robot trajectories.

Fig. 10 shows the state variable ρ which begins initially at $t=0$ with a value of 100 meters. Then, ρ sharply increases for a short period and it is considered that the state variable α greatly influence on the state variable ρ. In other words, ρ is directly proportional to the state variable α. However, ρ begins to severely descend and approach zero due to the influence of the feedback of prediction control. As discussed before, the effect of ρ approaching zero means that the robot approaches the final destination.

Fig. 11 shows the state variable α which means the difference between the mobile robot's heading and bearing angles.

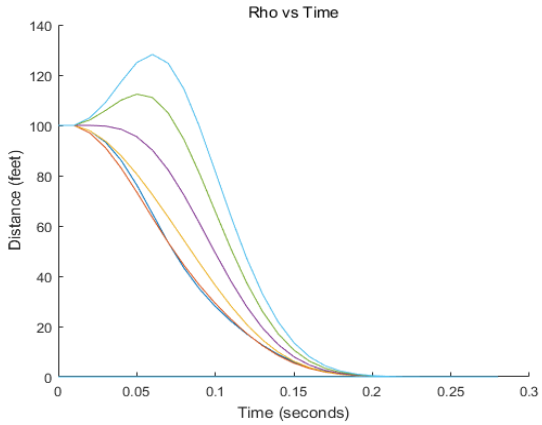

Figure 10. Experiment results of the state variable ρ.

If $\alpha = 0$, the mobile robot is heading in direct path to the goal position. Therefore, when the robot reaches the goal, it will be oriented in the direction of the goal position.

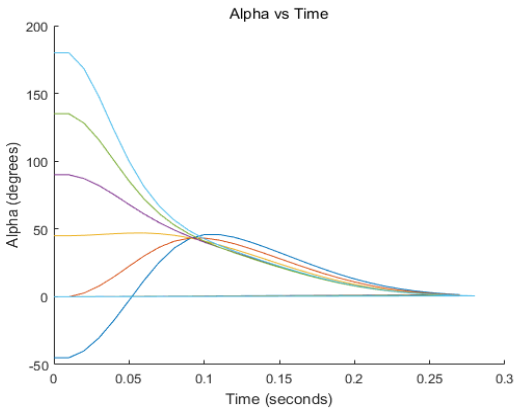

Figure 11. Experiment results of the state variable α.

In Fig. 11 and 12, it is shown that the peak value is dependent on the state variable α, β. If θ is zero, then β will approach zero. The β has the opposite effect to the state variable α. In other words, the state variable β makes for the mobile robot accomplish its desired pose at the final goal position specified by θ. Therefore, the to the system opposes the direct tracking of the bearing contributed by α, and causes the robot to reach its desired orientation at the goal specified by θ. Apparently, the state variable β is a function of two input variables of α and θ.

VI. Conclusion

This study presented a Model Predictive Control (MPC) framework using the kinematic bicycle model of a

car-like mobile robot for self-driving keeping a given path. For this, all of required parameters for the self-driving robot were defined for the car-like mobile robot with a differential steering mechanism (i.e., Ackerman steering system).

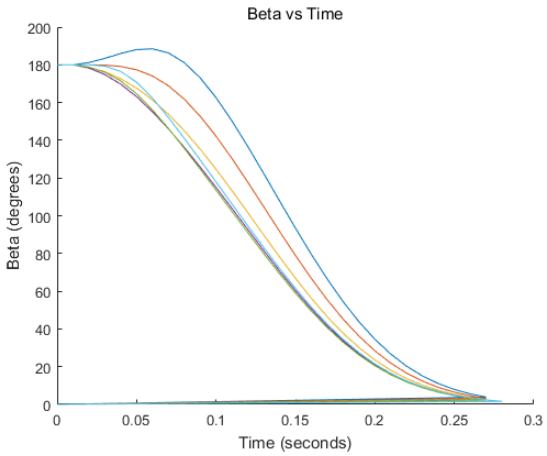

Figure 12. Experiment results of the state variable β.

The critical aspects of the controller design were presented with experimental results that show the proposed controller is able to control the car-like mobile robot. The predictive capability of the kinematic bicycle model of the mobile robot demonstrated that the mobile robot can follow the reference trajectory well at lower speeds.

For the future work, an obstacle avoidance and varied speed environment should be studied. Finally, the comparison with other control algorithms will be possible.

CONFLICT OF INTEREST

The author declares o conflict of interest.

AUTHOR CONTRIBUTIONS

The author conducted the research.

ACKNOWLEDGMENT

This research was funded by a 2019 research Grant from Sangmyung University.

REFERENCES

[1] N. H. Amer, H. Zamzuri, K. Hudha, and Z. Kadir, "Modelling and control strategies in path tracking control for autonomous ground vehicles: A review of state of the art and challenges," *Journal of Intelligent & Robotic Systems*, vol. 86, no. 2, pp. 225-254, 2017.

[2] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Kolter, D. Langer, O. Pink, V. Pratt, et al., "Towards fully autonomous driving: Systems and algorithms," *Intelligent Vehicles Symposium*, pp. 163–168, 2011.

[3] C. Shen, Y.Shi and B. Buckham, "Trajectory tracking control of an autonomous underwater vehicle using Lyapunov-based model predictive control," *IEEE Transactions on Industrial Electronics*, vol. 65, no 7, pp. 5796-5805, 2017.

[4] C. Urmson, J. Anhalt, D. Bagnell, and C. Baker, R. Bittner, et al., "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[5] Y. Gao, T. Lin, F. Borrelli, E. Tseng, and D. Hrovat, "Predictive control of autonomous ground vehicles with obstacle avoidance on slippery roads," *Dynamic Systems and Control Conference. American Society of Mechanical Engineers*, pp. 265–272, 2010.

[6] C. Beal and J. Gerdes, "Model predictive control for vehicle stabilization at the limits of handling," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 4, pp. 1258–1269, 2013.

[7] F. Borrelli, A. Bemporad, and M. Morari. Predictive control for linear and hybrid systems. [Online]. Available: http://www.mpc.berkeley.edu/mpc-coursematerial. 2015.

[8] C. Liu, S. Lee, S. Varnhagen, and H. E. Tseng, "Path planning for autonomous vehicles using model predictive control," *IEEE Intelligent Vehicles Symposium*, pp. 174-179, 2017.

[9] A. Carvalho, Y. Gao, A. Gray, H. Tseng, and F. Borrelli, "Predictive control of an autonomous ground vehicle using an iterative linearization approach," *Intelligent Transportation Systems*, pp. 2335–2340, 2013,

[10] A. Carvalho, Y. Gao, S. Lef`evre, and F. Borrelli, "Stochastic predictive control of autonomous vehicles in uncertain environments," *International Symposium on Advanced Vehicle Control (AVEC)*, 2014.

[11] C. Peter, "Robot Vision : Second Edition,"

[12] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, " Kinematic and dynamic vehicle models for autonomous driving control design," *IEEE Intelligent Vehicles Symposium*, pp. 1094-1099, 2015.

[13] P. Belven, "Implementation of model predictive control for path following with the KTH research concept vehicle," *KTH Royal Institute of Technology*, 2015.

Kiwon Yeom received his Ph.D. from the Department of Human Computer Interaction (HCI) and Robotics at University of Science and Technology, Daejeon, South Korea, in August 2007. The subject of his research concerns an adaptive and evolvable robotic architecture inspired by biological systems