Received: September 3, 2025 | Revise: October 9, 2025 | Accepted: October 10, 2025

JOURNAL OF APPLIED RESEARCH ON SCIENCE AND TECHNOLOGY
JARST
INSTITUTE OF RESEARCH AND DEVELOPMENT
RAJAMANGALA UNIVERSITY OF TECHNOLOGY THANYABURI
ISSN ONLINE: 2773-9473
https://ph01.tci-thaijo.org/index.php/rmutt-journal

A unified adaptive pure pursuit speed controller with EKF Sensor Fusion for real-world ackermann mobile robots

Nattapong Promkaew[1], Pasan kulvanit[2] and Somboon Sukpancharoen[1*]
[1]Department of Agricultural Engineering, Faculty of Engineering, Khon Kaen University, Khon Kaen 40002, THAILAND
[2]Department of Science Service, Ministry of Higher Education, Science, Research, and Innovation, Bangkok 10400, THAILAND
[*]Corresponding author: sombsuk@kku.ac.th

ABSTRACT

Accurate trajectory tracking is critical for autonomous mobile robots navigating in outdoor environments. This research presents an enhanced version of the Pure Pursuit algorithm, referred to as Pure Pursuit with Dynamic Steering Control (PP-DSC), which modulates the robot's velocity according to the magnitude of the steering angle while maintaining a fixed lookahead distance. The algorithm was implemented on a four-wheeled robot with Ackermann steering, and localization was achieved by fusing data from the Global Navigation Satellite System Real Time Kinematic (GNSS-RTK), Inertial Measurement Unit (IMU), and encoder data using an Extended Kalman Filter (EKF). Field experiments were carried out on three representative paths: straight line, S-curve, and loop at operational speeds ranging from 1.0 to 3.0 m/s. The results demonstrated that PP-DSC consistently reduced lateral deviation compared to fixed-speed Pure Pursuit. Root Mean Square Error (RMSE) was reduced to 0.9 cm on the straight-line path, and on the S-curve path, RMSE was reduced to 2.1 cm. The RMSE decreased to 1.9 cm for the loop path with PP-DSC, while the 1 m loo kahead configuration exhibited a higher RMSE of 2.7 cm. These findings confirm that steering-based velocity modulation effectively improves path-tracking precision under real-world outdoor conditions.

Keywords: Pure pursuit, Adaptive speed control, ROS 2 Jazzy, GNSS-RTK, Sensor fusion

INTRODUCTION

Getting robots to navigate outdoors reliably has proven harder than expected. Global Navigation Satellite System Real Time Kinematic (GNSS-RTK) should give us the centim-level accuracy achieving approximately ±10 cm precision when conditions are ideal [1, 2]. Unfortunately, real-world deployment rarely offers such luxury. Buildings create shadow zones where signals simply don't reach. Tree canopies scatter Global Positioning System (GPS) signals unpredictably, while atmospheric disturbances add their own errors [3, 4]. Weather conditions also matter; heavy rain or dense fog can significantly degrade signal quality. These problems compound quickly. A robot might lose position fix entirely at the worst possible moment, or see errors balloon far beyond that ±10 cm specification [5].

The obvious solution involves using multiple sensors together [6]. Extended Kalman Filters have become the standard tool for this job, and for good reason. Under normal GPS conditions, the Extended Kalman filter (EKF) provides marginal benefits, perhaps 3% improvement at best. But watch what happens when GPS degrades: suddenly, that same EKF system outperforms GPS-only navigation by 50% [7]. The filter merges whatever Real Time Kinematic (RTK) data it can get with Inertial Measurement Unit (IMU) measurements (accelerations, rotation rates) and wheel encoder counts. Even if satellites vanish completely, the robot navigates [8, 9]. The mathematics behind EKF involves recursive Bayesian estimation, predicting states based on motion models, then correcting with sensor observations. Granted, the system still faces that fundamental ±10 cm RTK limit, not to mention delays from mechanical steering linkages and control loops. Still, the multi-sensor approach plays to each technology's strengths. GPS anchors the global position, IMUs track rapid movements, and encoders measure distance traveled regardless of satellite availability.

Beyond sensor fusion lies another problem entirely: path planning computation. Dijkstra's algorithm and A* guarantee optimal paths, which sounds great until you realize their computation time explodes with environmental complexity. Add moving obstacles or multiple optimization criteria, and these classical methods become computationally prohibitive for real-time control. Graph-based methods work well for structured environments, but outdoor navigation often involves continuous spaces with irregular obstacles.

DOI: 10.60101/jarst.2024.263358

This limitation has pushed researchers toward metaheuristic algorithms - optimization methods that mimic biological and physical processes [10,11]. Ant colonies, bird flocks, evolutionary selection - nature solved these problems long ago. These algorithms do not promise perfect solutions like A* does, but they find reasonable solutions much faster and rarely get trapped in local minima [12]. The exploration-exploitation balance proves crucial here; too much exploration wastes time, while excessive exploitation misses better solutions.

Promkaew's recent work makes a compelling case. Their Artificial Bee Colony algorithm found paths 7% shorter than A* while using just 10% of the computation time in indoor navigation tests [13]. The algorithm divides its artificial bees into scouts, workers, and onlookers, each playing different roles in the optimization process. For robots making split-second decisions, efficiency gain matters more than theoretical optimality.

When it comes to following paths, Pure Pursuit dominates the field - primarily because it's so straightforward [14, 15]. Point the robot at a spot ahead on the path, drive toward it, pick a new spot, repeat. Simple enough that basic microcontrollers can run it without breaking a sweat [16]. The geometric interpretation is elegant: the robot follows a circular arc connecting its current position to the lookahead point. The trade-offs become apparent quickly, however, Standard Pure Pursuit reacts sluggishly to sudden path changes and struggles when speeds vary [17, 18]. Various fixes exist: some researchers adjust lookahead distance dynamically [19, 20], others borrow ideas from animal locomotion [21]. Recent work has explored machine learning approaches to tune params automatically. Yet most still run at fixed speeds, which worsens tracking through curves and amplifies any positioning errors [22].

Physics tells us the solution: slow down for tight turns, speed up on straights. The relationship between curvature and safe speed isn't complicated; it follows from basic centripetal force considerations, but implementations rarely exploit it properly [23]. Consider that lateral acceleration equals velocity squared divided by turning radius; exceeding tire friction limits means losing control.

Ackermann-steered robots face extra complications. Unlike differential drive robots that pivot on the spot, Ackermann platforms have minimum turning radii dictated by their geometry [24]. The front wheels must turn at different angles to avoid tire scrubbing - the inner wheel turns sharper than the outer. Handle these constraints correctly and lateral tracking errors drop to 2.5 cm with 60% less computation - impressive gains, though still limited by positioning accuracy [25]. The kinematic model gets more complex too, involving wheelbase length, steering angle limits, and slip considerations. Speed control deserves more credit than it gets. Yes, faster operation improves productivity, but strategic speed reduction through curves maintains both accuracy and stability [26, 27]. Field tests show that agricultural robots operating at variable speeds complete tasks with fewer path deviations than constant-speed systems. The steering angle directly indicates how much to slow down, making adaptive speed control straightforward to implement [28].

Modern navigation systems work best when sensor fusion, path planning, and control all support each other. Each layer solves specific problems: fusion handles sensor noise and failures, planning finds efficient routes through complex spaces, and adaptive control executes those plans smoothly. The integration challenges shouldn't be underestimated, though - getting these subsystems to communicate effectively requires careful software architecture.

Our contribution is straightforward: the modified Pure Pursuit to slow down based on steering angle while keeping the lookahead distance fixed. An EKF combines GNSS-RTK, IMU, and encoder data to squeeze the best possible performance from that ±10 cm RTK limitation. All experiments were conducted on an Ackermann-steered robot running ROS 2 Jazzy, chosen for its improved real-time capabilities and better hardware abstraction than ROS 1. Three main outcomes emerged from this work. First, the EKF-based navigation system improves raw RTK positioning by fusing complementary sensors. Second, scaling speed with steering demand (while maintaining constant lookahead) enhances Pure Pursuit (PP) tracking, which is particularly noticeable in S-curves and sharp corners. Third, tests across different paths at 1-3 m/s confirm better accuracy and smoother motion than fixed-speed methods. The improvement becomes more pronounced at higher baseline speeds, where the adaptive system prevents the overshooting common with constant-velocity controllers. Section 2 explains our system design, Section 3 details the experiments, Section 4 analyzes results, and Section 5 discusses implications and next steps.

MATERIALS AND METHODS

Hardware architecture and localization sensors

The mobile robot consists of a four-wheel platform with rear-wheel drive and front-wheel Ackermann steering, and its chassis measures 600 mm wide by 1000 mm long. DC motors provide traction, while a servo motor is responsible for steering. An industrial computer manages the robot's control system. Electrical power comes from a 24 V lithium battery (Figure 1).

A router makes wireless communication and system monitoring possible. For localization, the robot uses two GNSS antennas, with an approximate baseline of 1.2 m, antenna A integrated with a 9-axis

DOI: 10.60101/jarst.2024.263358

JOURNAL OF APPLIED RESEARCH ON SCIENCE AND TECHNOLOGY

IMU (3-axis gyroscope, 3-axis accelerom, and 3-axis magnetom) to enhance positioning accuracy (Figure 2). The wheel encoders attached to both rear motors provide 1024 pulses per revolution, supplying velocity feedback to EKF.

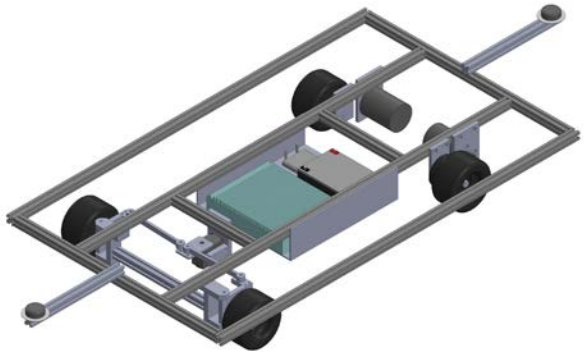

Figure 1 Autonomous Vehicle Component Layout.

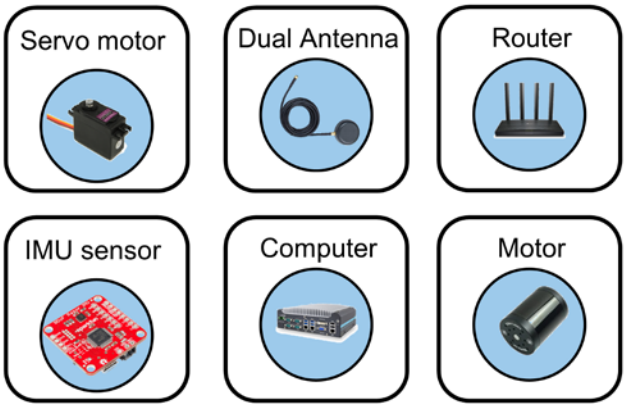

Figure 2 Hardware components of the autonomous vehicle.

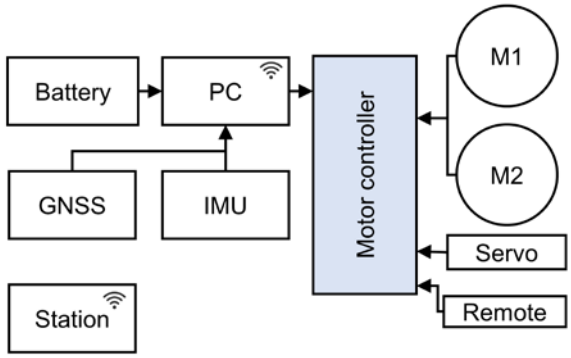

Figure 3 Robot system wiring diagram.

The internal wiring layout of the robot is shown in the robot system wiring diagram (Figure 3). A 24 VDC 25 Ah lithium battery supplies power to the entire system, with a DC-DC converter stepping down the voltage to 12 VDC for components that require it. The motor controller connects to two DC motors equipped with encoders for feedback and a servo motor used for steering, which can be operated both automatically and through remote RC signals. Communication between the main computer, motor controller, IMU, and GNSS modules is established through Controller Area Network (CAN) connections. The industrial computer processes sensor data, issues control commands, and is linked to the RTK receiver. The system communicates with a laptop and a base station via Ethernet and wireless networks. In addition, a 2.4 GHz wireless module is included to support remote operation and monitoring.

Robot development and integration

The robot development process diagram illustrates the steps involved in developing the autonomous robot (Figure 4). Development begins with system design, where the hardware is selected, the power system is planned, and wireless communication is set up. Once these foundations are established, focus shifts to integrating sensors and creating the software. At this stage, data from the GNSS-RTK, IMU, and encoders are combined using an EKF to give the robot an accurate estimate of its position. For navigation purposes, latitude and longitude data are converted into Universal Transverse Mercator (UTM) coordinates. All of these tasks are handled within the ROS 2 Jazzy environment. In the next phase, the robot's path-tracking algorithms are implemented, combining the Pure Pursuit approach with speed adjustment based on steering angle. The process concludes with field testing and a thorough evaluation to confirm that the robot performs as intended (Figure 4).

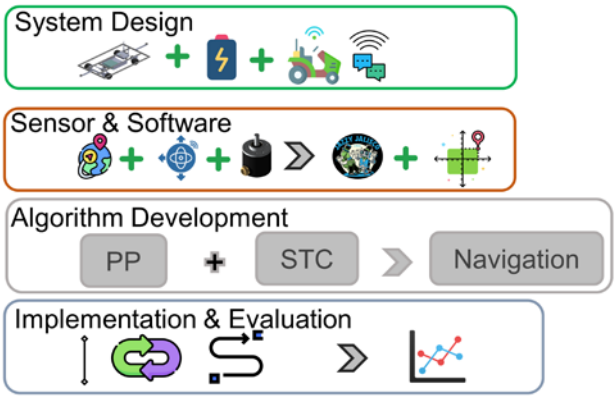

Figure 4 Robot Development Process.

Path tracking and velocity control algorithms

Path tracking for autonomous ground vehicles, most engineers tend to use a mix of geometric and kinematic models to figure out how the robot should steer and control its speed. The kinematic model, for example, is pretty straightforward, but it's proven to be a reliable foundation for many different control methods. One of the techniques that is often used is the Pure Pursuit algorithm. It's gained popularity mostly because it's simple to set up and tends to work well in real-time situations. But as just making the robot steer along the path usually isn't enough. There are plenty of times when the vehicle also needs to adjust its speed, especially if it's about to take a curve. Slowing down in tight turns can make a big difference

DOI: 10.60101/jarst.2024.263358

in terms of stability and overall safety, while on straight sections, you can generally get away with going a bit faster. That's why, in many cases, speed is tied to either how much the path is curving or what the steering angle is at a given moment. Putting these strategies together helps make the navigation system more flexible and robust so that the robot can handle all sorts of path shapes and situations it might encounter.

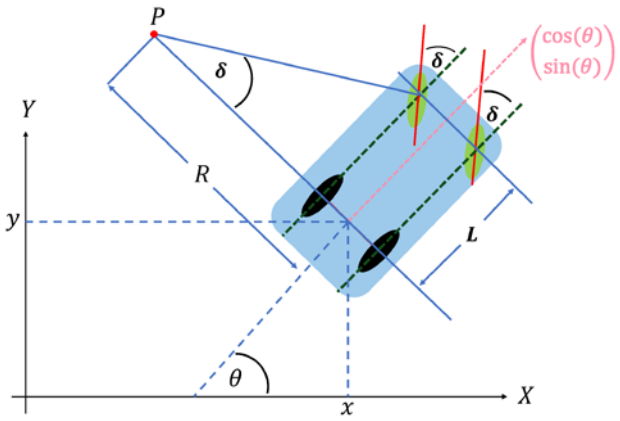

Figure 5 Kinematic bicycle model geometry.

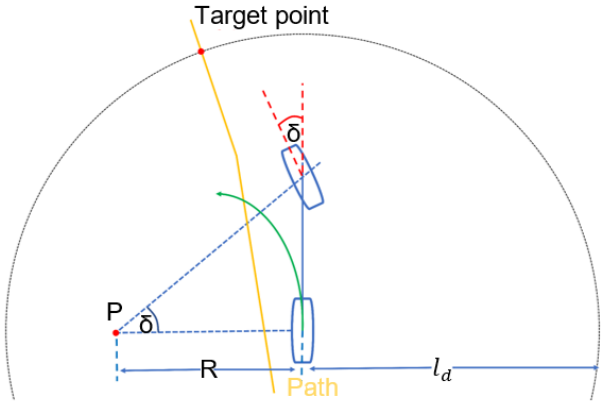

Figure 6 Pure pursuit tracking geometry.

The kinematic bicycle model provides the foundational framework for describing the motion of a car-like robot in the plane (Figure 5). The model is governed by the following continuous-time state-space equations (Eq. 1-4):

$$\frac{d}{dt}\begin{pmatrix} x \\ y \\ \theta \\ v \end{pmatrix} = \begin{pmatrix} v\cos(\theta) \\ v\sin(\theta) \\ v\tan(\delta)/L \\ a \end{pmatrix} \quad \begin{matrix} (1) \\ (2) \\ (3) \\ (4) \end{matrix}$$

where x and y denote the global position of the rear axle, θ is the heading angle of the vehicle, v is the linear velocity, L represents the wheelbase, δ is the front wheel steering angle, and a is the longitudinal acceleration input. The relationship between the turning radius R and steering angle δ is given by (Eq. 5):

$$R = \frac{L}{\tan(\delta)} \quad (5)$$

This relationship ensures that changes in the steering angle directly affect the curvature of the vehicle's trajectory.

The Pure Pursuit algorithm is widely adopted for lateral control due to its simplicity and effectiveness (Figure 6).

The algorithm first defines the lookahead distance l_d as a function of the robot's linear velocity (Eq. 6):

$$l_d = K_{dd}v \quad (6)$$

where K_{dd} is a proportional gain param. To ensure stable tracking, the lookahead distance is constrained within minimum and maximum bounds (Eq. 7):

$$l_{d,min} \le l_d \le l_{d,max} \quad (7)$$

Given a reference path, a target point at lookahead distance is identified in the vehicle frame, and the heading error α is calculated according to (Eq. 8):

$$\alpha = arctan2(y_{tp}, x_{tp}) \quad (8)$$

where (x_{tp}, y_{tp}) are the coordinates of the target point relative to the rear axle. The required turning radius to the target is then determined by the geometric relationship in (Eq. 9):

$$R = \frac{l_d}{2\sin\alpha} \quad (9)$$

Consequently, the steering angle command for the front wheel is obtained from the Pure Pursuit geometry, as given in (Eq. 10):

$$\delta = \arctan\left(\frac{2L\sin(\alpha)}{l_d}\right) \quad (10)$$

where L is the wheelbase of the vehicle (m), L_d is the look-ahead distance (m), and α is the geometric heading-error angle between the vehicle's current heading and the target point (rad).This equation determines the curvature of the circular arc that the robot must follow to reach the lookahead point. In practical navigation, stability and safety are further enhanced by dynamically adjusting the robot velocity according to the steering demand. The yaw rate ω is calculated as a function of the current velocity and steering angle (Eq. 3).

The normalized steering demand S is then defined as (Eq. 11):

$$S = \left|\frac{\omega}{\omega_{max}}\right| \quad (11)$$

where ω_{max} represents the maximum allowable yaw rate (rad/s) determined by the vehicle's kinematic limits. This normalization maps the instantaneous turning rate into the range [0,1], where $S = 0$ corresponds to straight-line motion and $S = 1$ indicates the sharpest turn achievable by the vehicle. The value of S is subsequently used in Eqs. (12) - (14) to

DOI: 10.60101/jarst.2024.263358

JOURNAL OF APPLIED RESEARCH ON SCIENCE AND TECHNOLOGY

interpolate the commanded velocity between v_{min} and v_{max} .

$$\eta = \frac{S - S_{lower}}{S_{upper} - S_{lower}} \quad (12)$$

when S lies between S_{lower} and S_{upper} , the interpolated velocity is given by Eq. 13:

$$v_{interp} = v_{min} + (v_{max} - v_{min})\,\eta \quad (13)$$

Finally, the commanded velocity is expressed as the following piecewise function (Eq.14):

$$v = \begin{cases} v_{min} & S > S_{upper} \\ v_{max} & S < S_{lower} \\ v_{interp} & otherwise \end{cases} \quad (14)$$

where v_{min} and v_{max} are the minimum and maximum speeds, respectively, and S_{lower} and S_{upper} are threshold values for speed adaptation. By sequentially applying Eqs. 6-14 at each control step, the robot dynamically determines the lookahead distance, computes the target heading and steering commands, calculates the yaw rate and normalized steering demand, and finally sets the appropriate speed for the current path curvature. A rate limiter was applied to the commanded speed to prevent abrupt velocity transitions and ensure smooth motion. The maximum acceleration and deceleration rates were limited to $a_{\text{acc, max}} = 0.8\ \text{m/s}^2$ and $a_{\text{dec, max}} = 1.2\ m/s^2$, respectively, based on empirical tuning from field experiments to prevent wheel slip and maintain stability during sharp turns. This integrated control strategy, summarized in Eq. 15 below, enables smooth and robust trajectory tracking:

$$\begin{cases} l_d = clip(K_{dd}v, l_{d,min}, l_{d,max}) \\ \alpha = arctan2(y_{tp}, x_{tp}) \\ \delta = \arctan\left(\frac{2L\sin(\alpha)}{l_d}\right) \\ \omega = \frac{v}{L}\tan\delta \\ S = \left|\frac{\omega}{\omega_{max}}\right| \\ v = \begin{cases} v_{min} & S > S_{upper} \\ v_{max} & S < S_{lower} \\ v_{interp} & otherwise \end{cases} \end{cases} \quad (15)$$

In this formulation, the lookahead distance l_d is dynamically adapted to the robot's velocity (Eqs. 6, 7), while the target heading angle α and steering command δ guide the robot towards the reference path (Eqs. 8-10). The yaw rate ω and normalized steering demand S (Eqs. 3, 11) are used to interpolate the commanded speed, with η and v_{interp} defined in Eqs. 12 and 13, respectively. The resulting velocity command (Eq. 14) ensures that the robot decelerates smoothly in sharp curves and accelerates on straight or gentle paths, resulting in a comprehensive and unified control law for smooth, adaptive, and reliable path tracking in diverse environments.

Experimental Params and Controller Settings

The key experimental params and control settings used in all tests are summarized in Table 1. These values were kept constant across all experiments to ensure fair comparison and reproducibility.

Table 1 Experimental params and controller settings.

Parameter	Value
PurePursuit gain (K_{dd})	0.8 s
Lower steering threshold (S_{lower})	0.3
Upper steering threshold (S_{upper})	0.9
Minimum speed (V_{min})	1.0 m/s
Maximum speed (V_{max})	3.0 m/s
EKF process noise (Q)	diag (0.05, 0.05, 0.01)
EKF measurement noise (R)	diag (0.2, 0.2, 0.1)

These params were empirically tuned based on preliminary field tests and remained fixed for all subsequent trials.

RESULTS AND DISCUSSION

This section evaluates the performance of the proposed Pure Pursuit with Dynamic Steering Control (PP-DSC) algorithm against the conventional PP method. Experiments were conducted on various trajectories at 1 to 3 m/s using a four-wheeled Ackermann-steered robot (Table 2). Results show that PP-DSC significantly reduces lateral tracking errors and improves velocity stability, especially on curved paths. Compared to the standard PP, PP-DSC provides smoother speed transitions and more accurate path following, demonstrating its effectiveness for real-time autonomous navigation in outdoor environments (Table 3).

In the straight-line tracking experiment, the robot stayed well aligned with the target path. The controller achieved an average lateral error of about 0.7 cm and a maximum deviation below 1 cm during the test (Figure 7, Table 3). This outcome shows that the control system, supported by sensor fusion, can effectively counter minor disturbances and hardware inaccuracies, allowing the robot to navigate smoothly in simple field conditions.

Table 2 Experimental conditions.

Param	Value
Coordinate System	UTM
Control Format	Cartesian (x, y)
Test Scenarios	line path, s-curve path, loop path
Environment	Real-world conditions
Positioning Method	GNSS-RTK

DOI: 10.60101/jarst.2024.263358

JOURNAL OF APPLIED RESEARCH ON SCIENCE AND TECHNOLOGY

Table 3 Comparison of tracking errors for different algorithms and paths.

Path	Algorithm	Mean (m)	RMSE (m)	STD
Line	PP + DSC	0.007	0.008	0.004
	PP L_d = 1 m	0.001	0.011	0.011
	PP L_d = 5 m	0.01	0.008	0.008
S-curve	PP + DSC	0.006	0.015	0.014
	PP L_d= 1 m	-0.004	0.021	0.021
	PP L_d= 5 m	0.01	0.023	0.021
Loop	PP + DSC	-0.001	0.019	0.019
	PP L_d= 1 m	-0.002	0.027	0.027
	PP L_d= 5 m	-0.001	0.019	0.019

Adjusting the lookahead distance in the Pure Pursuit algorithm noticeably influenced the robot's ability to follow the path accurately. When a 1 m lookahead was used, the robot reacted quickly to directional changes and stayed close to the reference trajectory, with a root mean square error (RMSE) of approximately 1.1 cm. In contrast, a 5 m lookahead resulted in slightly smoother motion but slower response, with an RMSE of about 0.8 cm and a maximum lateral error within 2 cm. These results, illustrated in the tracking comparison (Figure 8), highlight that both lookahead settings provided accurate performance, though the shorter distance yielded more responsive control.

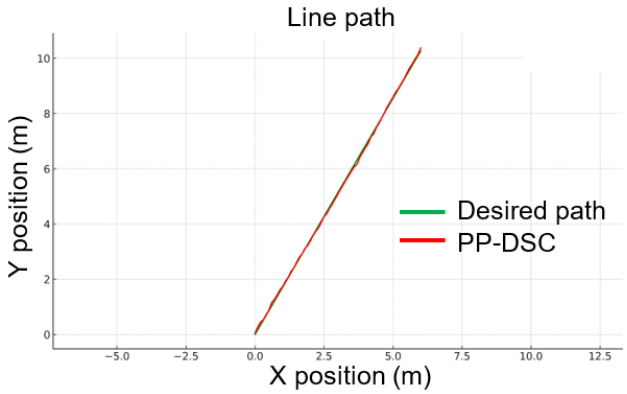

Figure 7 Tracking result on straight-line path.

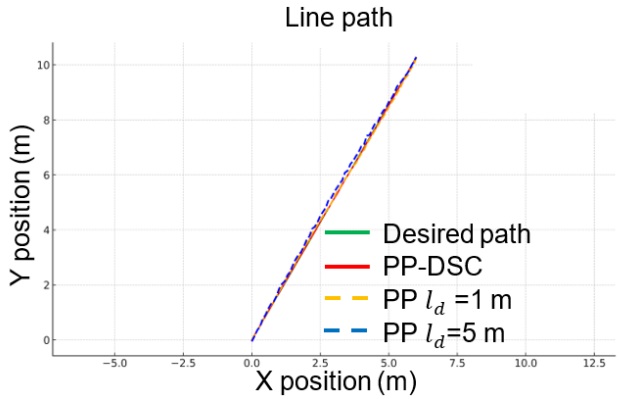

Figure 8 Line path tracking comparison with different lookahead distances.

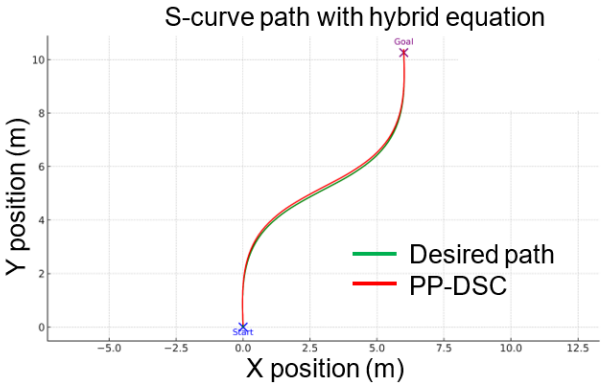

Figure 9 S-curve trajectory tracking using hybrid equation.

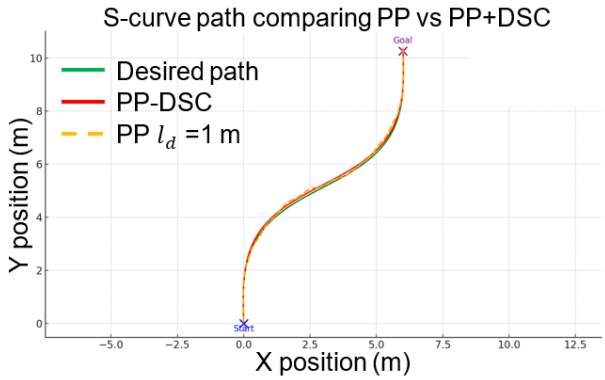

Figure 10 S-curve tracking comparison between PP and PP + DSC.

On the S-shaped trajectory, the robot managed to follow the desired path with steady performance. Despite the continuous change in curvature, the actual path stayed fairly close to the planned one, with lateral deviations mostly within 6-11 cm (Figure 9). These results show that the baseline control approach handles smooth curves reasonably well, though minor overshoot was observed during turning transitions.

To see whether adjusting speed based on steering demand could improve tracking, a comparison was made between standard Pure.

The pursuit and the version were enhanced with speed regulation (Figure 10). When the robot was allowed to slow down in curves, the tracking

DOI: 10.60101/jarst.2024.263358

line became more stable and stayed closer to the desired path especially in areas with tighter turns. The maximum lateral error was reduced by about 20% compared to the fixed-speed case, and the robot's motion appeared smoother overall.

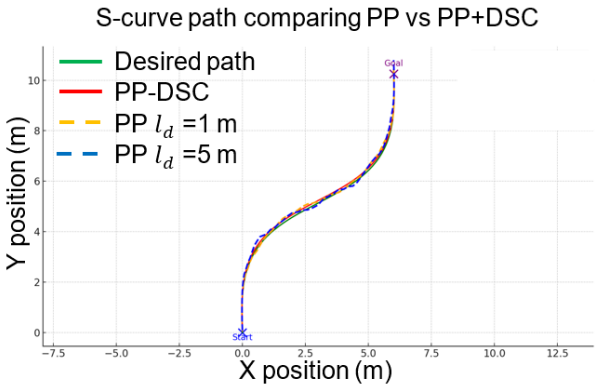

Figure 11 S-curve tracking: PP vs PP + DSC.

As the evaluation continued, both 1 m and 5 m lookahead distances were tested with the speed-adjusted controller (Figure 11). The robot successfully followed the S-curve in both cases. However, the shorter lookahead allowed it to stay slightly closer to the reference trajectory. The average tracking error with 1 m was approximately 2.1 cm, while the longer distance produced a comparable error of 2.3 cm. In addition, the 5 m setting resulted in smoother motion but slightly delayed responses near the curve transitions where quick steering corrections were required. These observations indicate that a shorter lookahead, combined with speed adaptation, provides marginally better responsiveness without sacrificing overall path smoothness.

As the evaluation continued, both 1 m and 5 m lookahead distances were tested with the speed-adjusted controller (Figure 11). The robot successfully followed the S-curve in both cases. However, the shorter lookahead allowed it to stay slightly closer to the reference trajectory. The average tracking error with 1 m was approximately 2.1 cm, while the longer distance produced a comparable error of 2.3 cm. In addition, the 5 m setting resulted in smoother motion but slightly delayed responses near the curve transitions where quick steering corrections were required. These observations indicate that a shorter lookahead, combined with speed adaptation, provides marginally better responsiveness without sacrificing overall path smoothness.

The robot was tested on a loop path to evaluate how well it could maintain accuracy through continuous curvature. As shown in the first plot (Figure 12), the actual path closely followed the planned one. Most of the lateral error remained within a 5-8 cm range, and there were no sharp oscillations or drift, even after multiple turns. The controller handled smooth transitions well without overcorrection or instability.

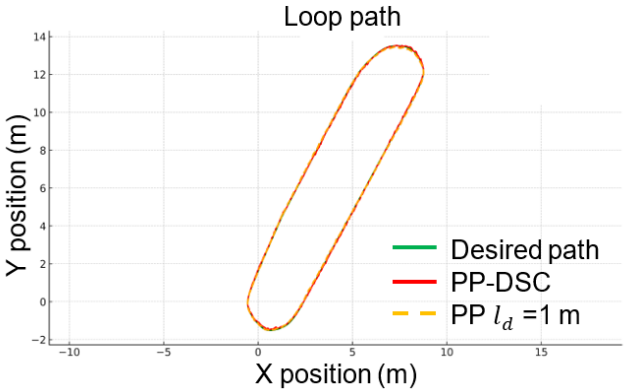

Figure 12 Loop path tracking using 1 m lookahead.

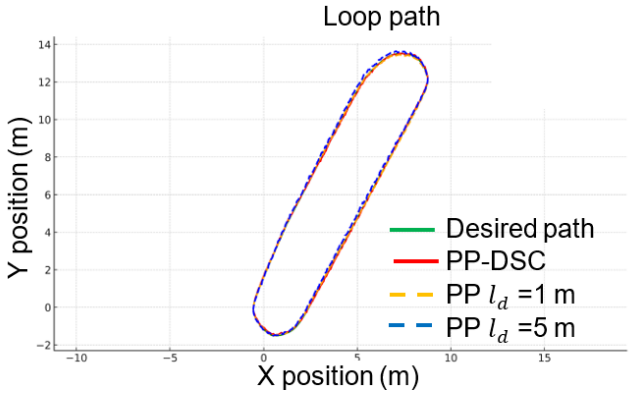

Figure 13 Loop path tracking with different lookahead distances.

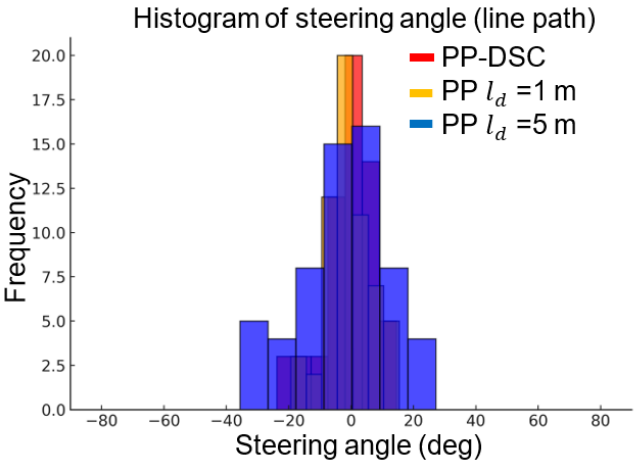

Figure 14 Steering angle distribution on line path.

To understand how the lookahead parameter influenced this behavior, both 1 m and 5 m settings were compared under the same conditions (Figure 13). Although both settings allowed the robot to complete the loop, the shorter lookahead kept the trajectory slightly tighter in curved segments. With the 1 m configuration, the RMSE was approximately 2.7 cm, while with the 5 m configuration it was 1.9 cm. The longer lookahead produced smoother steering and less oscillation, particularly near the top and bottom of the loop where heading corrections were more pronounced. Overall, the shorter lookahead yielded quicker response, whereas the longer distance

DOI: 10.60101/jarst.2024.263358

provided greater smoothness and stability through the curves.

In the straight-line test, steering angles remained close to zero throughout the trajectory, consistent with expectations for a path with no curvature (Figure 14). The actual robot data and both simulated cases using 1 m and 5 m lookahead distances showed similar results. Most values clustered tightly between -5 and +5 degrees, with minimal spread. Since the path did not require significant turning, all methods produced nearly identical steering behavior.

The loop path, which includes continuous curved segments, resulted in a wider range of steering angles (Figure 15). The actual steering values were concentrated mostly between ±20 degrees. When comparing the two lookahead settings, the 1 m configuration showed a narrower distribution, while the 5 m version introduced more variance and occasional steering commands that extended beyond ±30 degrees. This suggests that the longer lookahead led to less responsive adjustments, especially in curve entry and exit zones.

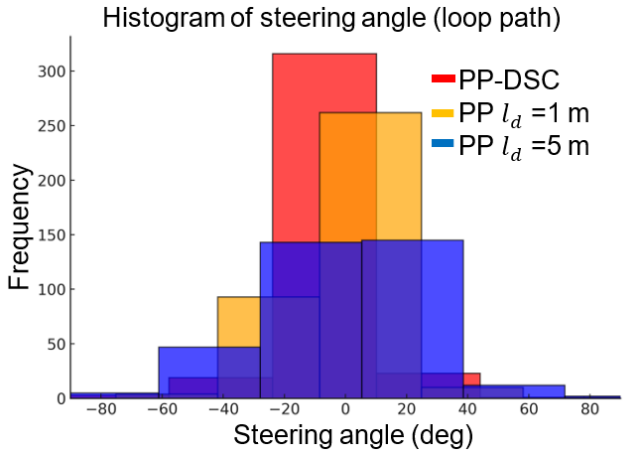

Figure 15 Steering angle histogram during loop tracking.

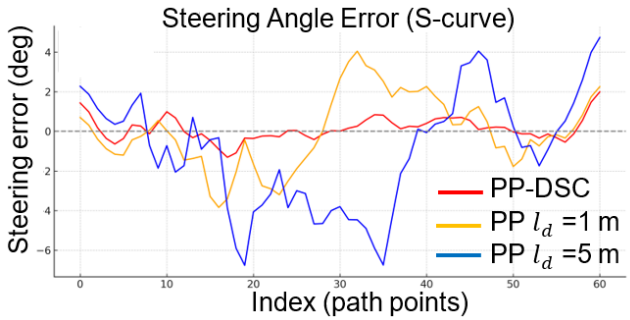

Figure 16 Steering angle error on the S-curve path.

For the S-curve path, steering angle error was analyzed over the entire trajectory to capture how the controller handled varying curvature (Figure 16). The robot's actual steering tracked the reference line steadily, with deviations staying mostly within ±3 degrees. The 1 m lookahead followed the same trend with only minor fluctuations.

On the other hand, the 5 m case produced larger swings, with peaks reaching up to ±6 degrees at curve transitions. This indicates that shorter lookahead distances tend to produce more stable and accurate steering, especially when the path includes frequent changes in direction.

After running the different tests, there are some differences in how well each method worked. It seems that the path-following got more accurate overall when the robot could change its speed and lookahead distance, like with the PP-DSC method. This was true whether the robot was moving along a straight line, weaving through an S-curve, or circling around a loop in Table 3.

Take the straight-line case first. Here, PP-DSC mean tracking error was 0.007 m, and the RMSE was 0.008 m. That's pretty tight tracking. Pure Pursuit, with a fixed lookahead of 1 m, was fairly close in terms of mean error, though its RMSE was a bit higher at 0.011 m. When the lookahead jumped to 5 m, the mean error increased to 0.010 m. Although the RMSE for that setting matched PP-DSC, the higher mean error might point to some drift or bias over time.

Things changed a bit when the robot followed a curvier path, like the S-curve. PP-DSC still gave better results: a mean error of 0.006 m and RMSE of 0.015 m. With a 1 m lookahead, Pure Pursuit had a negative mean error and the highest RMSE, at 0.021 m. It's interesting because a negative mean doesn't always mean better performance it can just mean errors are in the opposite direction. For the 5 m lookahead, The mean error was back up to 0.010 m, and the RMSE went even higher.

On the loop, PP-DSC mean error was close to zero, at -0.001 m, and RMSE was 0.019 m. Both fixed lookahead options did not quite match up, with either higher mean errors or higher RMSE, or sometimes both. These little gaps can matter, especially when you want the robot to follow a path as closely as possible.

One thing that stands out is that adjusting both speed and lookahead makes a real difference, especially on more complicated paths. When the robot's path curves or changes direction suddenly, it helps if the control algorithm can respond right away, instead of sticking to a single rule for all situations. The fixed lookahead strategies sometimes let errors build up, which is clear when you look at the bigger standard deviations or RMSE values.

Looking beyond the mean, you can see that RMSE and standard deviation tell more of the story. A low mean error could just result from errors canceling each other out, but a high RMSE suggests that big deviations are still happening. In these trials, PP-DSC generally kept both numbers lower, so the path-following was not only more centered, but also less erratic.

These results make a good case for using adaptive algorithms like PP-DSC for robots, especially

DOI: 10.60101/jarst.2024.263358

outdoors or in places where the route is unpredictable. This method seems to help the robot handle changes more smoothly and keep the tracking tight, which could be really valuable in real-world applications. There's still room to try these methods under even tougher conditions, but from what's here, the benefits of a flexible approach are already pretty clear.

CONCLUSIONS

In this work, the study set out to solve some persistent issues with path tracking for field robots, especially when GPS readings are not always stable. The approach used, called PP-DSC, is pretty straightforward. The main idea is to have the robot slow down when it needs to turn more sharply, rather than using the same speed everywhere. The study wanted to see if this would help, especially when GNSS-RTK alone isn't enough to keep the robot perfectly on track The proposed method was experimentally evaluated on three types of paths using a four-wheeled robot platform. The tracking error was very low for straight lines, with an RMSE of about 0.9 cm. On the S-curve, the error was higher at 2.1 cm, while on the loop path, RMSE reached 1.9 cm. For comparison, the regular Pure Pursuit algorithm with a 1 m fixed lookahead produced an RMSE of about 2.7 cm, highlighting the improvement achieved by the proposed method. PP-DSC worked because it automatically reduced speed as the steering angle increased. This allowed the control system more time to respond when the robot needed to turn sharply.

One thing that helped a lot was sensor fusion using an EKF. By combining GNSS-RTK, IMU, and wheel encoder data, the robot was able to keep a steady estimate of its position, even when the satellite signal wasn't great. This reliability is beneficial in real farming, where losing GPS for a few seconds can easily happen.

The significance of this precision is not merely theoretical; precision matters greatly for farming jobs. You don't want to overlap or miss spots if you're spraying. For planting, you need accurate spacing. When harvesting, the machine has to move carefully between rows or trees to avoid damaging the crops. An additional advantage of the proposed solution is that it works on regular computers and does not require expensive hardware. This means smaller farms could afford to use robots like this, not just large industrial farms.

Some challenges were encountered during the experiments. The optimal speed thresholds were determined through iterative testing and may require further tuning when applied to rougher terrain or muddy field conditions. The experiments were primarily conducted on flat terrain, and weather effects were not explicitly considered in this study. Also, the robot was not loaded with extra equipment. Adding tools for spraying, planting, or harvesting will change how it moves, so future tests should include that.

Several aspects remain for future work. Testing on real farms with different soil types and crops is important because corn fields and orchards present very different challenges. Soil itself can make a big difference, too, as loose sand is not the same as sticky clay. In the future, it might be possible to use machine learning to adjust parameters automatically or to use cameras to help the robot see and follow crop rows more precisely. Future research will also extend the comparison to other path-tracking controllers such as Stanley and MPC to benchmark the proposed method further and validate its advantages under more diverse control frameworks. Ultimately, it will be important to get feedback from farmers to see if the system is safe and practical to use at normal working speeds.

To sum up, making farm robots practical is more important than perfecting them. Farmers need reliable, affordable automation that supports routine agricultural operations. The results show that with some simple changes to a well-known algorithm, path-tracking performance can be significantly improved, even without top-level GPS or high-speed computers. As farm work gets more complex and labor shortages increase, this solution will become more valuable.

DECLARATION OF AI AND AI-ASSISTED TECHNOLOGIES IN THE WRITING PROCESS

The authors utilized ChatGPT-5 pro and Claude Opus 4.1 to improve the academic tone and language clarity of the manuscript during its preparation. All content was critically reviewed and edited by the authors, who assume full responsibility.

ACKNOWLEDGEMENT

This research was made possible through the support of Gen serv Company Limited, and the authors also express their appreciation to Khon Kaen University for granting access to their research facilities.

REFERENCES

1. Radočaj D, Plaščak I, Jurišić M. Global navigation satellite systems as state-of-the-art solutions in precision agriculture: A review of studies indexed in the web of science. Agriculture. 2023;13:1417.
2. SIZA Robotics. SIZA Robotics launches autonomous robot for vegetable and beet crops. Future Farming; 2024.
3. Moeller R, Deemyad T, Sebastian A. Autonomous navigation of an agricultural robot using RTK GPS and pixhawk. In: Proceedings of 2020 Intermountain Engineering, Technology and Computing; IETC; 2020.

DOI: 10.60101/jarst.2024.263358

4. Advanced Navigation. Autonomous Agriculture, Precision Farming & Robotics. 2025.
5. Wang Q, Zhang Q, Rovira-Más F, Tian L. Stereovision-based lateral offset measurement for vehicle navigation in cultivated stubble fields. Biosyst Eng. 2023;109:258-65.
6. Ning X, Wang F, Fang J. Sensor Fusion of GNSS and IMU Data for Robust Localization via Smoothed Error State Kalman Filter. Sensors. 2023;23(7):3676.
7. Kaczmarek A, Rohm W, Klingbeil L, Tchórzewski J. Experimental 2D extended Kalman filter sensor fusion for low-cost GNSS/IMU/Odoms precise positioning system. Measurement. 2022;193:110963.
8. Zhang A, Atia MM. An efficient tuning framework for Kalman filter param optimization using design of experiments and genetic algorithms. NAVIGATION: Journal of the Institute of Navigation. 2020;67(4): 775-93.
9. Moore T, Stouch D. A generalized extended kalman filter implementation for the robot operating system. In: Intelligent Autonomous Systems 13: Proceedings of the 13th International Conference IAS-13; 2015 Sep 3; Cham: Springer International Publishing; 2015. p. 335-48.
10. Liu Y, Jiang Y. Motion planning of differential driven robot based on tracking. Control Decis. 2023;38:2529-36.
11. Dorigo M, Birattari M, Stutzle T. Ant colony optimization. IEEE computational intelligence magazine. 2007;1(4):28-39.
12. Yang XS. Nature-inspired metaheuristic algorithms. Luniver press; 2010.
13. Promkaew N, Thammawiset S, Srisan P, Sanitchon P, Tummawai T, Sukpancharoen S. Development of metaheuristic algorithms for efficient path planning of autonomous mobile robots in indoor environments. Results in Engineering. 2024;22: 102280.
14. Coulter RC. Implementation of the pure pursuit path tracking algorithm. 1992.
15. Samuel M, Hussein M, Mohamad MB. A review of some pure-pursuit based path tracking techniques for control of autonomous vehicle. Int J Comput Appl. 2016;135(1):35-8.
16. Baltazar JD, Coelho AL, Valente DS, Queiroz DM, Villar FM. Development of a Robotic Platform with Autonomous Navigation System for Agriculture. AgriEngineering. 2024;6(3).
17. Wang L, Chen Z, Zhu W. An improved pure pursuit path tracking control method based on heading error rate. Industrial Robot: the international journal of robotics research and application. 2022; 49(5):973-80.
18. Jiang X, Kuroiwa T, Cao Y, Sun L, Zhang H, Kawaguchi T, et al. Enhanced Pure Pursuit Path Tracking Algorithm for Mobile Robots Optimized by NSGA-II with High-Precision GNSS Navigation. Sensors. 2025;25(3):745.
19. Ge LI, Yu WA, Liufen GU, Junhua TO. Improved pure pursuit algorithm for rice transplanter path tracking. Nongye Jixie Xuebao/Transactions of the Chinese Society of Agricultural Machinery. 2018;49(5).
20. Xu L, Yang Y, Chen Q, Fu F, Yang B, Yao L. Path tracking of a 4WIS-4WID agricultural machinery based on variable look-ahead distance. Applied sciences. 2022 Aug 29;12(17):8651.
21. Zhao S, Zhao G, He Y, Diao Z, He Z, Cui Y, et al. Biomimetic adaptive pure pursuit control for robot path tracking inspired by natural motion constraints. Biomimetics. 2024;9(1):41.
22. Ding H, Liu H, Zhuang Y, Kan M, Xia D, Ding S. Multi-robot path planning based on four-wheel differential speed model. Control Eng China. 2023;30:730-8.
23. Rajamani R. Vehicle dynamics and control. Boston, MA: Springer US; 2006.
24. Carpio RF, Potena C, Maiolini J, Ulivi G, Rosselló NB, Garone E,et al. A navigation architecture for ackermann vehicles in precision farming. IEEE Robotics and Automation Letters. 2020;5(2): 1103-10.
25. Lei C, Li J, Deng Y, Tan X. RRT* ASV: Improved RRT* path planning method for Ackermann steering vehicles. Expert Systems with Applications. 2025;279:127349.
26. Yan J, Zhang W, Liu Y, Pan W, Hou X, Liu Z. Autonomous trajectory tracking control method for an agricultural robotic vehicle. International Journal of Agricultural and Biological Engineering. 2024;17(1):215-24.
27. Mahmud MS, Abidin MS, Mohamed Z, Abd Rahman MK, Iida M. Multi-objective path planner for an agricultural mobile robot in a virtual greenhouse environment. Comput Electron Agric. 2019;157:488-99.
28. Kiani F, Seyyedabbasi A, Nematzadeh S, Candan F, Çevik T, Anka FA, et al. Adaptive metaheuristic-based methods for autonomous robot path planning: sustainable agricultural applications. Applied Sciences. 2022;12(3):943.

DOI: 10.60101/jarst.2024.263358