JOURNAL OF
EMERGING INVESTIGATORS

Article

Comparing the performance of lateral control algorithms on long rigid vehicles in urban environments

Artyom Boyarov[1], Eoin Shannon[1]
[1] Whitgift School, Croydon, United Kingdom

SUMMARY

Existing research on autonomous vehicle control is largely focused on how control algorithms perform on cars; long vehicles or buses have not been significantly investigated. Existing studies indicate that people display positive attitudes to using autonomous buses, indicating the value of researching control algorithms for autonomous buses. To provide insight on the performance of control algorithms on autonomous buses, we compared multiple lateral control algorithms on how well they maneuver a long vehicle around three courses resembling urban environments. We compared the Stanley and pure pursuit control algorithms and two new control algorithms which were improved versions of the Stanley and pure pursuit controllers. We compared the control algorithms in a kinematic simulation which recorded the steering angle and cross track error from the front axle for each controller driving the vehicle at 50 km/h around the course. We hypothesized that the Stanley control algorithm would perform the best due to its success in navigating existing vehicles. The Stanley control algorithm had a low cross track error, but it had a large steering angle and large changes in steering angle. The pure pursuit controller had smoother changes in steering angle, but the cross track error was larger than the Stanley controller. Our results suggest that none of the algorithms we tested were optimal, and that an algorithm which can utilize the ability of the Stanley controller to maintain a low cross track error while also keeping a low steering angle change will perform the best for long vehicles.

INTRODUCTION

Self-driving cars have been developing at an increasing rate over the past few decades, starting with basic highway driving in the 1980s to almost full autonomous driving capability showcased in the Defense Advanced Research Projects Agency (DARPA) Grand Challenges of the 2000s, where multiple teams from industry and academia developed vehicles to drive autonomously around courses in the desert and urban areas (1). Autonomous vehicles feature software handling perception, navigation, and vehicle control (2). The perception module uses sensor data to produce a map of the environment surrounding the vehicle (2). Navigation modules consist of mission and motion planners (2). The mission planner finds the route the car should take to reach its destination, while the motion planner generates a trajectory, based on the map of the surroundings provided by the perception module, for the car to follow to reach the next waypoint that the mission planner has given (2). The motion planner also generates a velocity profile for the car to follow on the current segment of its route (3). The controller of the car follows the trajectory (lateral control) and speed profile (longitudinal control) for the section generated by the motion planner, by setting the steering angle and throttle input for the car respectively (3).

Multiple algorithms for lateral control have been developed, based on kinematic models of the vehicle. Algorithms based on kinematic models use the bicycle model to represent the vehicle, where the car is represented as a two-wheeled body with the front wheel providing turning (3). Kinematic model-based controllers are simple to implement, involve a low computation overhead and provide reasonable performance in fixed scenarios at a moderate speed, such as in urban settings where the speed usually does not exceed 30 miles per hour (3). Kinematic controllers were used by multiple vehicles in the DARPA Urban Challenge, including the winning vehicle, Stanley, which used the Stanley controller. Three vehicles used the pure pursuit kinematic controller in the same challenge (3). The Stanley controller aligns the front wheel of the vehicle with the trajectory while steering towards the trajectory, whereas the Pure Pursuit controller picks a point on the trajectory a set distance from the vehicle and sets the steering angle to move to that point (3).

Existing research is focused on comparing the effectiveness of such algorithms when controlling conventional cars. Not much research focuses on autonomous buses, but they can be more beneficial for the environment and could improve public transport. When surveyed, people show positive attitudes towards using autonomous buses (4). If full autonomous capability is to be achieved, especially in urban environments, vehicles such as buses or trucks also need to be made autonomous. The motion of a bus is different from a car, and so existing lateral control algorithms may be less effective at navigating longer buses than small cars. For example, according to the bicycle kinematic model of the vehicle, the angular velocity of a bus is lower because of the greater length between axles, making turning harder (3). Investigating the performance of existing kinematic control algorithms on longer vehicles would give valuable insights on how the algorithms could be adjusted to cope with longer

lengths (such as how gain parameters could be adjusted) or whether new algorithms need to be designed. Some research has been conducted into various control algorithms for articulated vehicles (5), but for non-articulated vehicles of a long length the performance of control algorithms has not yet been discussed.

Therefore, the aim of our investigation was to compare the performance of various lateral control algorithms when used to control a long vehicle on a trajectory similar to one in an urban environment. The trajectories used in our investigation would involve features usually present on roads in urban environments, such as frequent turns and roundabouts. We compared the Stanley and pure pursuit control algorithms, as well as two algorithms designed by ourselves based on the Stanley and pure pursuit controllers which aim to counteract their shortcomings. The first new algorithm designed, referred to as Stanley with Lookahead (SL), was a modified Stanley controller which used the heading of a point on the trajectory in front of the vehicle instead of the heading of the point on the trajectory closest to the vehicle. The second new algorithm, referred to as Hybrid Stanley and pure pursuit (SPP), either used pure pursuit control or Stanley control based on the current cross track error. If the cross track error was above a certain threshold, the vehicle used Pure Pursuit control to get back to the trajectory. Otherwise, Stanley control is used to ensure the vehicle stays aligned with the trajectory. We hypothesized that the Stanley control algorithm would be able to achieve smoother turning and a lower cross track error than the other algorithms and would thus prove to be a better control algorithm. In the 2005 DARPA Urban challenge, the winning vehicle uses the Stanley control algorithm, and the vehicle did not hit any obstacles (6). Furthermore, the authors of the paper on the Stanley control algorithm highlight the ability of the controller to operate with a low cross track error (6). In our investigation, we developed a simulation of the bicycle kinematic model of a vehicle which simulated the motion of a vehicle responding to changes in steering angle. The steering angle was set based on the current control algorithm being tested. We implemented all of the controllers tested based on the available literature and the *PythonRobotics* GitHub repository (3, 7). To compare each algorithm's performance, we measured the average cross track error (measured from the car's front axle) as well as the steering angle over time. The cross track error of the vehicle is defined as the distance from the vehicle's rear axle or the front axle to the nearest point on the trajectory (3). In our experiment, we measured the cross track error from the front axle of the car.

The major conclusion drawn from our research was that no control algorithm was able to provide a low cross track error as well as smooth steering, indicating that an optimal control algorithm is yet to be developed, or other controllers could perform better. Therefore, my conclusion did not support my hypothesis, as the Stanley control algorithm did not have better steering than the pure pursuit algorithm or the SL and SPP algorithms. My investigation could be extended to environments similar to urban ones, such as in a warehouse where robots may have to transport goods between stations along fixed paths, or in an airport where transit buses and other vehicles have designated roads for them to travel on.

RESULTS

To compare the cross track error and steering angle over time for the control algorithms on the test courses, we developed a simulation which used the bicycle kinematic

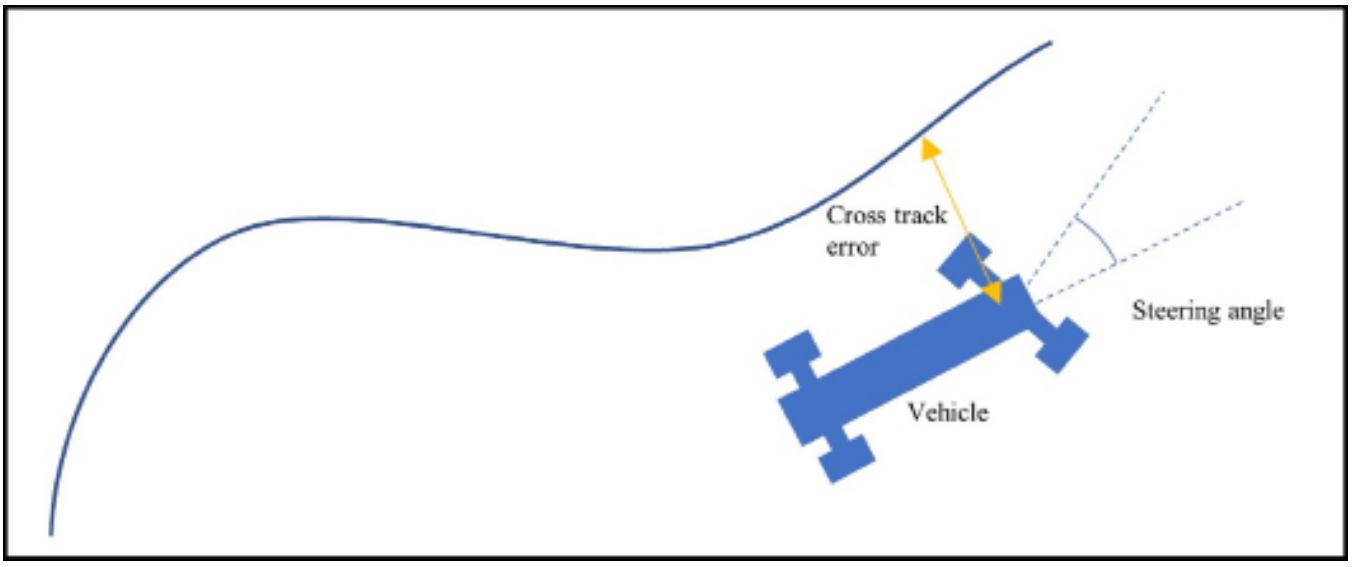

Figure 1: A diagram of a two-axled vehicle illustrating the cross track error and steering angle. The vehicle is represented as a solid body with two axles with wheels. The steering angle is the angle between the heading of the vehicle and the heading of the front axle. The cross track error is the distance from the center of the front axle to the closest point on the trajectory. The values for the cross track error and steering angle were recorded every 0.1 seconds during the simulation, and these values were used as experimental data in our investigation to evaluate the performance of the control algorithms. The figure was produced using Microsoft Word.

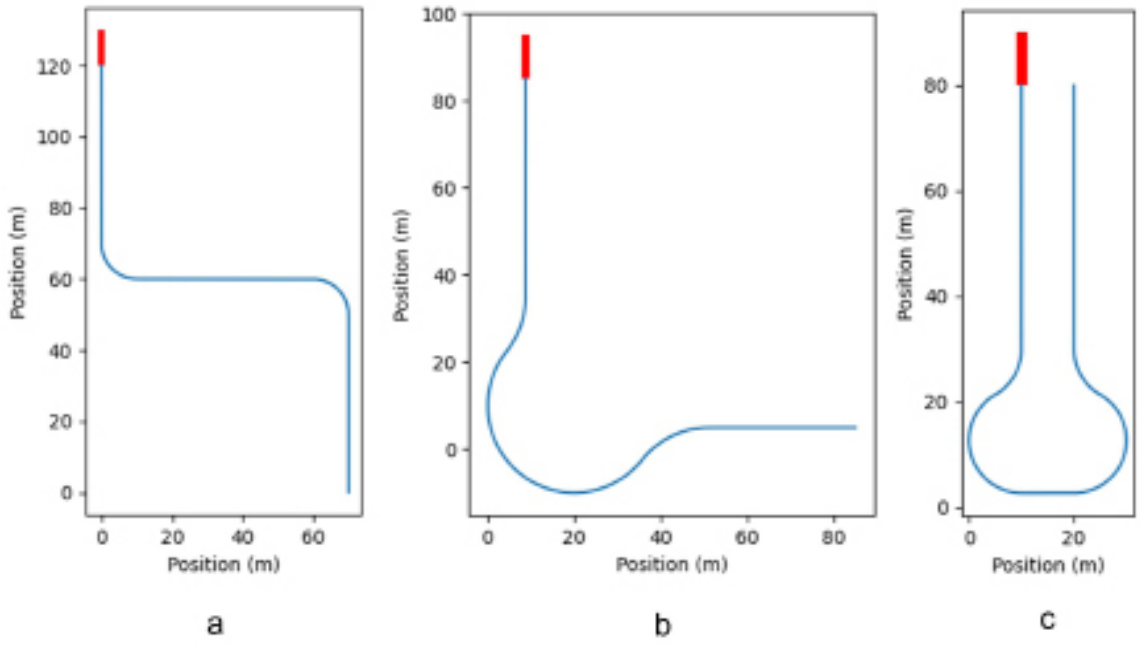

Figure 2: Course maps of the three courses used for the simulation. Figure 2a is the straight track course. Figure 2b is the three-quarter turn course. Figure 2c is the roundabout course. Units are in meters, with the values on the axis indicating the position of the point on the trajectory given as a coordinate. The origin, point (0,0), was chosen arbitrarily for the courses. The bus is shown as a red rectangle at the start of the course for scale. The figures were produced using the Python programming language and Matplotlib library.

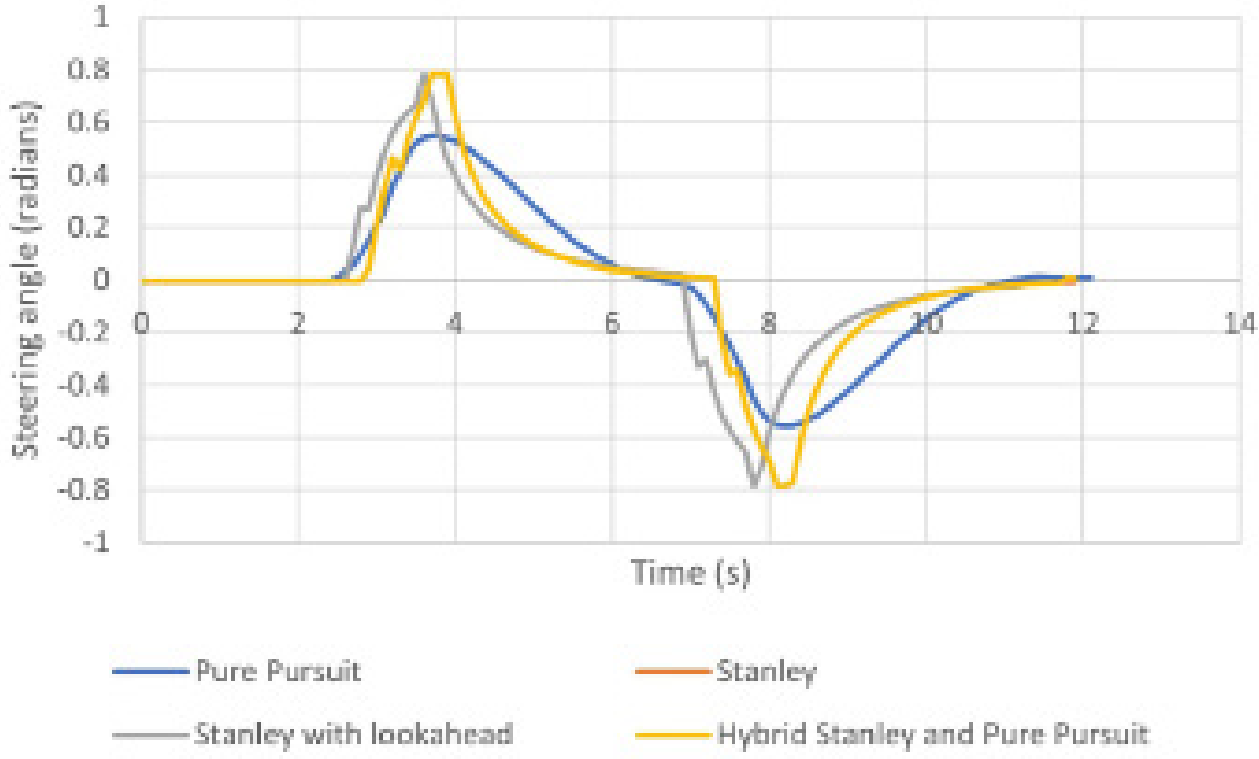

Figure 3: Steering angle of vehicle at 50 km/h on the straight track course using different controllers. Line graph showing steering angle of front axle every 0.1 seconds. Steering angle (in radians) was recorded from the simulation with pure pursuit controller (blue), Stanley controller (orange), Stanley with lookahead (SL) controller (gray), and hybrid Stanley and pure pursuit (SPP) controller (yellow). The results for the Stanley controller were identical to the SPP controller, hence the orange line for the Stanley controller is not visible. The experiment was performed once.

Course	Average cross track error for control algorithm (meters)			
	Pure Pursuit	Stanley	Stanley with lookahead	Hybrid Stanley and Pure Pursuit
Roundabout	1.34	0.54	1.08	0.96
Straight track	1.04	0.21	0.93	0.21
Three-quarter turn	1.03	0.59	1.00	0.77

Table 1: Average cross track error of vehicle at 50 km/h on each course. Table showing the average cross track error in meters of vehicle on each course for each of the control algorithms tested (pure pursuit, Stanley, Stanley with lookahead, and hybrid Stanley with pure pursuit). Cross track error was recorded as the distance to the nearest point on the trajectory from the front axle and summed every 0.1 seconds. At the end of the simulation, the total cross track error was divided by the time taken to calculate the average cross track error. The experiment was performed once for each course.

model to simulate a bus travelling at 50 km/h while also responding to changes in steering angle given by the control algorithm being tested. In the simulation, we sampled the vehicle's cross track error from the front axle and the steering angle every 0.1 seconds. We chose to measure the cross track error from the front axle of the car as the front axle guides the vehicle and having the front axle of the car close to the trajectory ensures that the car is driving towards the trajectory (6). In existing control algorithm reports, the cross track error was also measured from the front axle when evaluating the control algorithm's performance (6). **Figure 1** shows the cross track error from the front axle. A lower average cross track error indicates that the controller successfully follows the trajectory and remains close to the trajectory. The steering angle was measured for two reasons. First, large turns will cause a vehicle to risk overturning, where a vehicle turns too much and tips over (2). Second, having sudden changes in steering angle will cause the vehicle to jolt, thereby reducing passenger comfort (2). Having a lower steering angle with smoother changes will provide a more comfortable and safer experience for passengers in the vehicle. After results were collected in the simulation, they were processed and analyzed in Microsoft Excel.

The first course **(Figure 2a)** featured two sections of straight track connected with two turns. The pure pursuit controller offered low steering angles and smooth turns, whereas the Stanley, SL and SPP controllers had a larger steering angle and more sudden changes in steering angle compared to the pure pursuit controller **(Figure 3)**. However, the Stanley and SPP controllers had a much lower average cross track error than the pure pursuit and SL controllers **(Table 1)**.

In the second course **(Figure 2b)**, the vehicle took a three-quarter turn around a roundabout with a diameter of

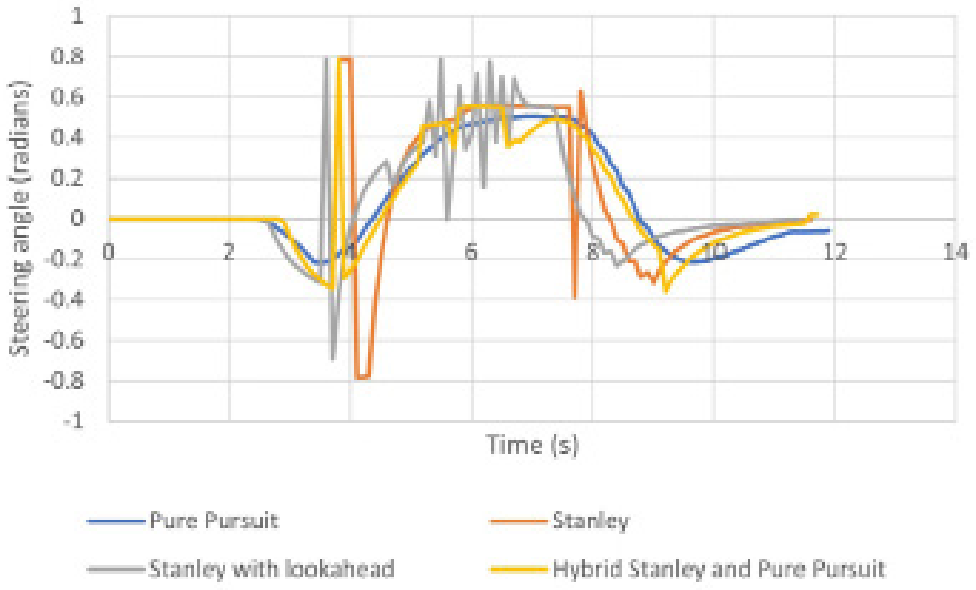

Figure 4: Steering of vehicle at 50 km/h on the three-quarter turn course using different controllers. Line graph showing steering angle of front axle every 0.1 seconds. Steering angle (in radians) was recorded from the simulation with pure pursuit controller (blue), Stanley controller (orange), Stanley with lookahead (SL) controller (gray), and hybrid Stanley and pure pursuit (SPP) controller (yellow). The experiment was performed once.

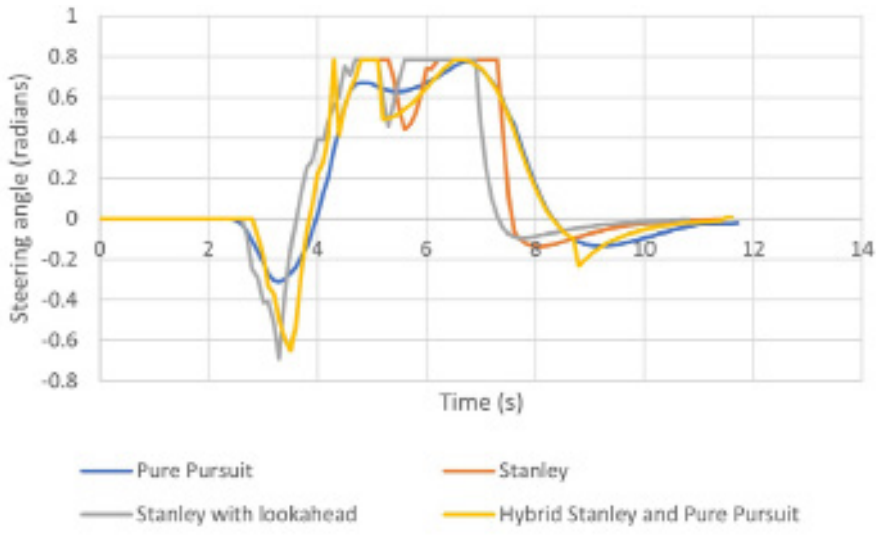

Figure 5: Steering of vehicle at 50 km/h on the roundabout course using different controllers. Line graph showing steering angle of front axle every 0.1 seconds. Steering angle (in radians) was recorded from the simulation with pure pursuit controller (blue), Stanley controller (orange), Stanley with lookahead (SL) controller (gray), and hybrid Stanley and pure pursuit (SPP) controller (yellow). The experiment was performed once.

20 meters. The pure pursuit controller had smoother steering than the other controllers **(Figure 4)**. The Stanley controller had sudden changes in steering angle and a high steering angle. The SL and SPP controllers had more sudden changes in steering angle. However, they did not have as high of a steering angle as the Stanley controller. The Stanley and the SPP controllers had the lowest average cross track error, and the pure pursuit and SL controllers had the highest average cross track error **(Table 1)**.

The last course (Figure 2c) involved a vehicle traversing a roundabout fully. Similar to the previous experiments, the pure pursuit controller had smooth steering **(Figure 5)**. However, at times it had a large steering angle. The other controllers all had large steering angles, as well as sudden changes in steering angle. The Stanley controller had the lowest cross track error, and the SPP had a cross track error which was not significantly higher **(Table 1)**. The SL and pure pursuit controller had the largest cross track error.

DISCUSSION

Each of the experiments presented a similar pattern of results: the pure pursuit controller had smooth steering and a low steering angle, and the Stanley controller and SPP controller had the lowest cross track error. For some experiments the cross track error of the other controllers compared did not differ greatly from the cross track error of the Stanley controller, namely the three-quarter turn on the roundabout course. However, in the experiments, the Stanley, SPP and SL controllers had sudden changes in steering angle as well as a large steering angle. The pure pursuit controller was unable to steer round the turns without having a large cross track error. For the Stanley controller and the two new controllers, the sharp steering angle resulted from the heading of the trajectory changing suddenly, as the Stanley and new controllers use the current heading of the trajectory to set the steering angle.

The simulation was quite accurate as it was not affected by external factors. The steering angle was stored as a variable in the simulation, meaning there is no measurement error in recording the steering angle. When the experiments were repeated, the results obtained were identical. However, the simulation had the speed fixed to 50 km/h, but in an actual scenario, the speed of the vehicle will change, especially while turning. The speed will decrease to help a vehicle round the turn. In future studies, a more realistic simulator can be employed to provide more realistic results. For example, the CARLA simulator, an open-source vehicle simulator, features advanced vehicle physics and so the effect of turning on the speed of the vehicle can be observed (8).

The results therefore indicate that no single controller offers optimal control of a vehicle, thus not providing support to our hypothesis that the Stanley controller offers good steering and a low cross track error. The Stanley controller offers a low cross track error but at the cost of sudden changes in steering, which would be uncomfortable for passengers or goods. The pure pursuit controller had smoother turns, but it had a large cross track error. The SL and SPP controllers had a lower cross track error than the pure pursuit controller, but they had poor steering. A controller which can avoid the large change in steering angle while having a low cross track error is a possible next step for development. A possible implementation may be a modified Stanley controller which outputs a value for the rate of change of steering angle, which is clamped within a margin, so that the vehicle does not change the steering angle suddenly.

Apart from looking for an improved kinematic lateral

control algorithm, the possible next step may be to compare predictive algorithms for following a trajectory, which use more complex algorithms to find the steering angle and speed to follow individual segments of the trajectory (3). Predictive algorithms may offer improved performance than the kinematic algorithms, albeit at a higher computational cost. Predictive algorithms can also adjust the vehicle's speed while following the trajectory, and so vehicles may reduce their speed while going round a corner. Articulated vehicles may also be investigated. Articulated trucks are very common, and articulated buses are used around the world. Articulated buses offer higher passenger capacity and faster transportation and are already used in many cities (9).

Overall, in contrast to initial hypothesis, our research showed that no control algorithm offered optimal performance. Each controller either had a low cross track error but high steering angles, or smooth steering with a high cross track error. This indicates that none of the kinematic control algorithms we tested provides optimal results, and so an algorithm which addresses the shortcomings of the Stanley and pure pursuit controllers or a predictive algorithm such as model predictive control may produce better results.

MATERIALS AND METHODS

Overview of the Simulation and Experiment

A kinematic simulation of a two-axled vehicle representing the bus was developed in the Python programming language with the Matplotlib library providing a graphical frontend. The kinematic simulation used was the bicycle model of a two-axled vehicle, where the front axle of the vehicle provides turning **(Figure 1)** (3). The bus was 10 m long in the simulation, which is a similar length to existing buses. Each control algorithm was implemented as a function which returned the required steering angle at each iteration of the simulation. The simulation was run at a fixed time step of 0.1 seconds. At each iteration, the function for the control algorithm being tested was invoked to calculate the steering angle of the vehicle for the current iteration. The vehicle's position was then updated based on the speed and steering angle using the formulas provided for the bicycle kinematic model (3). In the simulation, the vehicle's cross track error and steering angle were sampled every iteration. A flowchart indicating how the simulation worked is provided **(Figure 6)**. Control algorithms were implemented using the available literature as well as the *PythonRobotics* GitHub repository (3, 7). In the simulation, the pure pursuit and Stanley control algorithms produced a steering angle using the formulas available from the literature. The SPP controller and the SL controller also used the Stanley and pure pursuit functions as part of their processing. We set the speed in the simulation to 50 km/h, roughly 30 mph, as this is the speed limit in urban areas in many countries around the world (10).

Each of the simulated courses were chosen to represent a situation a vehicle might encounter in an urban environment. The CARLA Simulator, an industry-grade simulator for

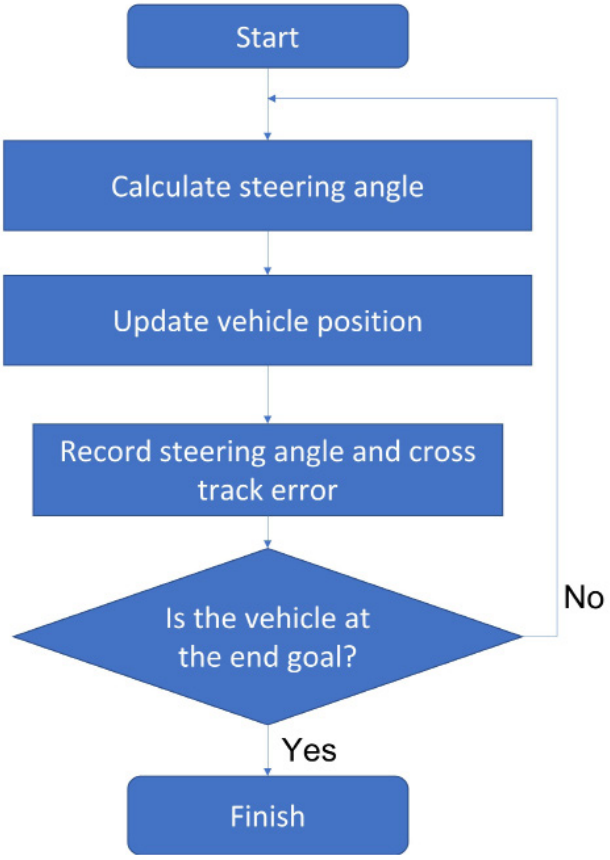

Figure 6: A flowchart of the simulation used in the experiment. Each control algorithm was defined as a function which was called during every iteration of the simulation. The function would return a value for the steering angle for the vehicle. The vehicle position was updated using kinematic equations for the motion of the vehicle based on the speed and steering angle. The cross track error was then calculated by finding the Euclidean distance from the center of the front axle to the closest point on the trajectory and then recorded. The steering angle, which was stored as a variable, was also recorded. After updating the position, if the vehicle was within 1 meter of the end goal, the simulation finished. The figure was produced using Microsoft PowerPoint.

autonomous vehicle research, provides multiple maps which are employed to simulate town environments (11). Namely, these maps had the following distinct features: straight sections, right-angled turns, and roundabouts (11). Therefore, the courses we chose involved traversing two right-angled turns, traversing a roundabout fully and traversing three-quarters of the way around a roundabout. The course maps with the bus presented for scale are provided **(Figure 2)**. Trajectories for the simulation were made up of straight segments and curved line segments; points on the trajectories were generated using linear interpolations and geometric Slerps.

While the simulation was running, values for the steering angle and cross track error (measured as the distance from the front axle to the nearest waypoint, illustrated in **Figure 1**) were recorded and then stored in a csv file at the end of the simulation. These values were then processed and analyzed

in Excel. Each control algorithm was tested on each course once. The experiment was not repeated as the results produced by the simulation were identical. The average cross track error was calculated by summing the cross track error at each iteration in the simulation and dividing the total cross track error by the number of samples taken. The code is publicly available; the GitHub page is provided in the appendix.

Control Algorithms Tested

The first control algorithm tested was the pure pursuit control algorithm. At each step, the algorithm finds a point on the trajectory which is at a lookahead distance *L* away from the vehicle. Then, the algorithm calculates the required steering angle for the front axle based on the lookahead distance using a formula (3). The formula used is illustrated in **Equation 1.1**. In the equation, *d* represents the steering angle, *L* represents the length between the two axles of the car, a is the heading from the front axle to the waypoint and l_d is the distance to the waypoint.

$$d = \arctan\left(\frac{2L\sin(a)}{l_d}\right) \quad \textbf{(1.1)}$$

The second main control algorithm used was the Stanley control algorithm. This algorithm calculates the heading error (the difference between the vehicle's heading and the heading of the current point on the trajectory) as well as a term to correct for the cross track error. These two terms are added together, and this value is set as the steering angle. If the cross track error is low, the cross track error correcting term will be small, and the control algorithm will adjust the steering angle so that the vehicle follows the trajectory. If the vehicle deviates from the trajectory, the cross track error correcting term is larger, and the vehicle turns towards the trajectory, reducing the cross track error (3, 6). **Equation 1.2** illustrates the Stanley control algorithm. In this equation, *k* is a gain parameter, *e* is the cross track error, *v* is the speed of the vehicle, Θ is the current heading of the vehicle, and Θ_t is the trajectory heading. A larger value for the *k* term means the vehicle takes longer to return to the trajectory, and a smaller *k* term means the vehicle returns to the trajectory faster.

$$d = \arctan\left(\frac{-ke}{v}\right) + (\theta - \theta_t) \quad \textbf{(1.2)}$$

The SPP controller was a combined Stanley and pure pursuit controller which would operate based on the current cross track error. If the cross track error was larger than the set threshold, the vehicle used the pure pursuit control algorithm to provide the steering angle. Otherwise, the Stanley control algorithm was used to provide the steering angle. The threshold value for the cross track error was set by trying out different values and finding which value gave the lowest average cross track error for the courses.

The SL controller was a modified Stanley controller with lookahead turning: the controller uses the heading of a point on the trajectory ahead of the vehicle's current position to set the current steering angle, using the same formula as the Stanley control algorithm. This algorithm was chosen as for longer vehicles, the larger distance between the axles would make turning harder, and so by turning earlier the vehicle would be able to turn easier.

For each of the control algorithms, gain parameters and the cross track error threshold (for the SPP controller) were chosen by running the simulation multiple times to find parameters which gave the lowest average cross track error. This is because analyzing the cross track error was easier than analyzing the steering angle to gain insight on which parameters were optimal.

APPENDICES

The code for the simulation is available at github.com/heemogoblin/trajectory-following-simulation/tree/master.

ACKNOWLEDGMENTS

We would like to thank Professor Howie Choset of Carnegie Mellon University for reading through our manuscript. We would also like to thank Ilya Makarov of the National University of Science and Technology for reading through our article and providing useful and detailed points for improvement. We would like to thank Artyom Boyarov's parents for their assistance during his study and research and for funding a course on autonomous vehicles, which encouraged him to undertake this research project.

Received: May 11, 2022
Accepted: July 13, 2022
Published: January 9, 2023

REFERENCES

1. Campbell, Mark, *et al.* "Autonomous Driving in Urban Environments: Approaches, Lessons and Challenges." *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences,* vol. 368, no. 1928, Oct. 2010, doi:10.1098/rsta.2010.0110.
2. Urmson, Chris, *et al.* "Autonomous Driving in Traffic: Boss and the Urban Challenge." *AI magazine,* vol. 30, no. 2, Jun. 2009, doi:10.1609/aimag.v30i2.2238.
3. Paden, Brian, *et al.* "A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles." *IEEE Transactions on Intelligent Vehicles,* vol. 1, no. 1, Mar. 2016, doi:10.1109/tiv.2016.2578706.
4. Mouratidis, Kostas, and Victoria Cobeña Serrano. "Autonomous Buses: Intentions to Use, Passenger Experiences, and Suggestions for Improvement." *Transportation Research Part F: Traffic Psychology and Behaviour,* vol. 76, Jan. 2021, doi:10.1016/j.trf.2020.12.007.
5. Bolzern, Paolo and Arturo Locatelli, "A comparative study of different solutions to the path-tracking problem for an articulated vehicle," *Proceedings of the International*

Conference on Control Applications, vol. 1, Sep. 2002, doi:10.1109/CCA.2002.1040224.
6. Hoffmann, M. Gabriel *et al.* "Autonomous Automobile Trajectory Tracking for Off-Road Driving: Controller Design, Experimental Validation and Racing." 2007 *American Control Conference,* Jul. 2007, doi:10.1109/ACC.2007.4282788.
7. Atsushi Sakai, *et al.* "PythonRobotics: a Python code collection of robotics algorithms.", Sep. 2018.
8. Dosovitskiy, Alexey *et al.* "CARLA: An Open Urban Driving Simulator". *Proceedings of the 1st Annual Conference on Robot Learning,* Nov. 2017.
9. El-Geneidy, Ahmed, and Nithya Vijayakumar. "The Effects of Articulated Buses on Dwell and Running Times." *Journal of Public Transportation,* vol. 14, no. 3, Sep. 2011, doi:10.5038/2375-0901.14.3.4.
10. Wikipedia contributors. "Speed Limits by Country." *Wikipedia.* en.wikipedia.org/wiki/Speed_limits_by_country. Accessed 24 Feb. 2022.
11. "Maps - CARLA Simulator." *CARLA Documentation.* carla.readthedocs.io/en/latest/core_map/. Accessed 10 Mar. 2022.