Journal of Engineering Science and Technology
Vol. 18, No. 5 (2023) 2388 - 2397

EFFICIENT AUTONOMOUS ROAD VEHICLES LOCAL PATH PLANNING STRATEGY IN DYNAMIC URBAN ENVIRONMENT USING RRT-ACS, BI-DIRECTIONAL RULE TEMPLATES, AND CONFIGURATION TIME-SPACE

MUHAMMAD ARIA RAJASA POHAN*, JANA UTAMA

Electrical Engineering Department, Universitas Komputer Indonesia,
Jl. Dipati Ukur 102-116, Bandung 40132, Indonesia
*Corresponding Author: muhammad.aria@email.unikom.ac.id

Abstract

This study aims to present an effective local path-planning strategy for Autonomous Vehicles (AVs) operating in a dynamic urban environment. The method combines bi-directional rule templates, configuration time-space, and RRT-ACS algorithm (hybridization of rapidly exploring random tree star with ant colony system). The use of RRT-ACS accelerates the convergence of feasible pathway planning. Path quality is improved using bi-directional rule templates derived from dynamic urban environment traffic scenarios and combining the RRT-ACS algorithm with configuration time-space. The resulting technique, RRT+BRT+CTS (RRT-ACS with two-way rule template and configuration space-time), emerged as a major research breakthrough. Several simulations of dynamic urban environmental conditions are used to validate the effectiveness of the RRT+BRT+CTS. Comparisons were made with two well-known path-planning algorithms used in dynamic urban environments: Fast RRT and Closed Loop RRT (CL-RRT). The simulation results demonstrate the superior performance of the proposed strategy for local path planning in dynamic environments regarding speed and efficiency for AVs. As a result, the proposed strategy is a suitable choice for addressing the complex challenges of Avs's urban navigation.

Keywords: Autonomous road vehicles, Bi-directional rule templates, Configuration time-space, Path planning, RRT-ACS algorithm.

1. Introduction

Autonomous Vehicles (AVs) can operate without a human driver [1]. AVs can have a big impact on human life [2]. Some of the contributions of AVs include autonomous taxis that are free from human driver crime [3], becoming a tourist attraction [4], increasing travel safety [5], reducing traffic jams [6], and reducing the need for parking spaces [7]. Some important processes that the AVs system must carry out are generating a global path consisting of waypoints, detecting objects around the vehicle, predicting the movement of dynamic objects, and generating local paths [8]. This research focuses on developing local path-planning algorithms in dynamic environments.

Many literature-based path-planning methods for dynamic environments exist. Rapidly exploring random tree (RRT) algorithms has attracted much attention in dynamic path-planning research. Previous research revealed another innovative approach using the MOD-RRT* algorithm, which involves path planning and replanning [9]. However, MOD-RRT* is only used in mobile robot navigation, not AVs. In previous research the Closed Loop RRT (CL-RRT) has been introduced. Another research introduces AV-specific path-planning strategy called Fast RRT [11]. Traffic-based rule templates and aggressive search tree expansion are introduced in Fast RRT. Fast RRT is faster than standard RRT. However, the above methods use the basic RRT* algorithm and have not been tested with more advanced RRT* improvement algorithms.

Many RRT* algorithm improvements have been developed. The RRT*-connect technique [12] optimizes pathways but requires a full state search. This is in line with previous research on informed RRT* approaches [13]. breakthrough. However, it encounters difficulties when a tiny route obstructs the target node. To overcome this restriction, one of the solutions is to use informed RRT*-connect technique [14]. Alternatively, another solution is to use RRT-ACS algorithm [12]. The RRT-ACS algorithm outperformed RRT*-connect, RRT*, informed RRT*-connect, and informed RRT* in Friedman's nonparametric test. Despite these advances, no prior research has used the RRT-ACS algorithm for AV path planning.

This study proposes an effective technique for local path planning designed for AV navigating in dynamic urban environments. Our solution integrates the RRT-ACS algorithm, bidirectional rule templates, and configuration time-space into the path planning framework. The following is the novelty of this study.

i) We propose using the bi-directional rule templates. These templates differ from previously proposed [11], which only uses a one-directional rule template. These templates also differ from another previous research [15], where they used it for the mobile robot case. In this study, we used the rule templates for the AV case with different constraints than the mobile robot.

ii) We propose a path-planning implementation strategy to solve complex road environment cases in a space-time configuration. Although previous research has integrated path planning and configuration time-space [11], their experiment is limited to a straight line with three vehicles. In contrast, the experiments in this study will be carried out on various road cases.

iii) We propose employing the RRT-ACS to speed up the search tree expansion to achieve the goal state. RRT-ACS accelerates the search tree expansion compared to RRT* [15]. However, they implemented RRT-ACS in mobile robotics. According to the authors, no previous studies have used the RRT-ACS algorithm for on-road AV path planning.

Various simulations of dynamic urban scenarios are used to demonstrate the RRT+BRT+CTS efficiency. Two well-known path-planning algorithms for dynamic urban environments, Fast RRT and Closed Loop RRT (CL-RRT), are compared to RRT+BRT+CTS. The results show that the proposed technique is faster and more efficient, especially for dynamic on-road autonomous driving. Thus, the proposed method suits AVs' navigation in complex urban environments.

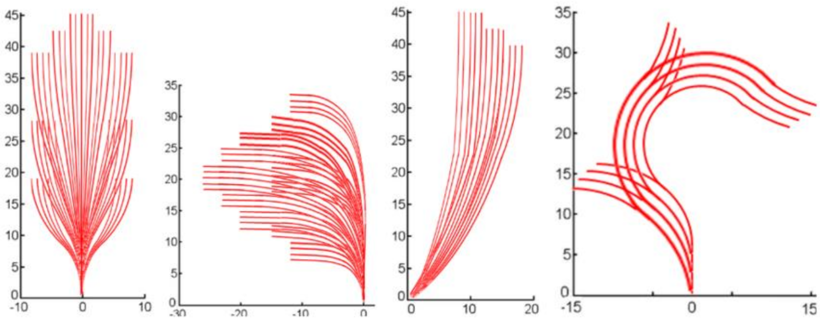
2.Proposed Algorithm: RRT+BRT+CTS

The core of our approach is using the RRT-ACS algorithm, bi-directional rule templates, and configuration time-space in the path planning process. We introduce a set of bi-directional rule templates in Section 2.1. Section 2.2 will thoroughly discuss the path-planning process in a configurable time-space framework suitable for dynamic urban environments. And finally, the complete proposed algorithm RRT+BRT+CTS is detailed in Section 2.3.

2.1. Bi-directional rule templates

Path for AVs has manoeuvre characteristics that follow the geometric structure of environments. Common manoeuvres for AV are going straight, turning right/left, lane merging/switching, roundabouts, and U-turns. Therefore, template rules for AV can be generated based on those categories. The rule templates consist of several templates based on AV manoeuvre categories.

Previous research has proposed using rule templates in path planning but only proposed four maneuvers (going straight, turning right, turning left, and U-turns) [11]. In this study, as proposed by the National Highway Traffic Safety Administration, we add two more templates: lane merging and roundabouts, as shown in Fig. 1.



(a) (b) (c) (d)

Fig. 1. Example rule templates for (a) straight forward, (b) left turns, (c) lane merging, and (d) roundabouts.

In this study, we also proposed bi-directional rule templates, as illustrated in Fig. 2. In the bi-directional templates method, two rule templates are used, which act as search trees. One template tree started from the starting node (red tree in Fig. 3), and another started from the goal node (blue tree in Fig. 2). Both trees aggressively seek a connection. If these two trees establish a connection, the process is complete. If these trees cannot connect, the RRT-ACS algorithm will be implemented to speed up these trees' growth to connect. A smooth and feasible path can be generated if these two trees are connected, as shown in the green curve in Fig. 2.

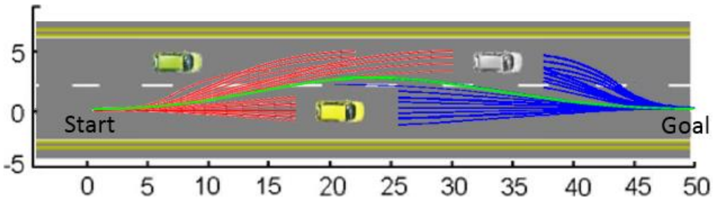


Fig. 2. A proposed path planning using bi-directional rule templates.

2.2. Configuration time-space for path planning

Using the configuration time-space framework for the path-planning process can enhance the path-planning results in dynamic environments. While conventional AV path planning uses 2-D spatial maps, incorporating configuration time-space adds time as an extra dimension. The static and dynamic obstacles can, therefore, be mapped. Consequently, the algorithm can adjuster each state's arrival time to the new node to avoid collisions with obstacles.

Figure 3 shows an example of implementing path planning in a configuration time-space. The yellow area shows the dynamic obstacles moving forward. The red and blue curves show the bi-directional rule templates. The final trajectory is represented in the green curve.

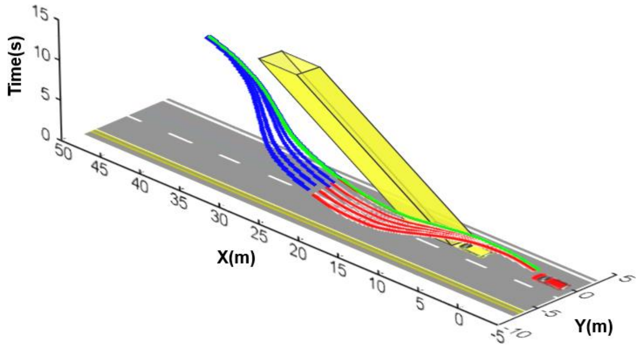


Fig. 3. A proposed path planning in the configuration time-space.

2.3. RRT+BRT+CTS algorithm

The RRT+BRT+CTS algorithm is shown in Fig. 4. First, the system will detect any dynamic objects around the AV. Path planning in the configuration time-space will occur if dynamic objects exist. However, if there are no dynamic objects, the path planning will be carried out without configuration time-space (lines 1-8). This procedure will reduce the computational overhead.

After that, two rule templates suitable for the environment around the AV are loaded (lines 3-4). If any nodes within these templates encounter collisions, they are pruned, leaving only collision-free nodes in the root structure (lines 9-10). The procedure is complete when these two trees are connected (lines 11-14). When a connection cannot be made, the RRT-ACS algorithm comes into play to speed up tree growth (lines 15-35).

Algorithm 1 : Proposed local path planning for AV

```
1.  env_inf ← get environmental_information
2.  if DynamicObstacles = true :
3.      T_A ← LoadTemplateOnCTR(q_start, TemplateSet, env_inf)
4.      T_B ← LoadTemplateOnCTR(q_goal, TemplateSet, env_inf)
5.  else
6.      T_A ← LoadTemplateOn2DMap(q_start, TemplateSet, env_inf)
7.      T_B ← LoadTemplateOn2DMap(q_goal, TemplateSet, env_inf)
8.  end
9.  T_A ← Trim Tree(T_A)
10. T_B ← Trim Tree(T_B)
11. if CanConnected(T_A, T_B) :
12.     X_soln ← BestPath(T_A, T_B)
13.     return X_soln
14. end
15. while TerminationConditionNotMet :
16.     for each iteration from k = 1 to m:
17.         while s = 0:
18.             q ← randvar[0,1]
19.             if q ≤ q_0 and τ ≠ ∅
20.                 q_rnd ← ACSSampling(τ, map)
21.             else
22.                 q_rnd ← RRTSampling(k, map)
23.             end
24.             T_A ← GrowTreeBasedOnRRT(q_rnd, T_A)
25.             T_B ← GrowTreeBasedOnRRT(q_rnd, T_B)
26.             if CanConnected(T_A, T_B):
27.                 X_bs ← UpdateBestPath(T_A, T_B)
28.                 s ←1
29.             end
30.         end
31.     end
32.     X_bs ← LocalSearch(X_bs, map)
33.     τ ← UpdatePheromone(τ, X_bs)
34. end
35. return X_soln ← X_bs
```

Fig. 4. Proposed method: RRT+BRT+CTS.

3. Results and Discussion

To assess the performance of the RRT+BRT+CTS algorithm, we executed it within five dynamic environment simulations. These scenarios replicated previous researches testing conditions [11, 16, 17]. The simulations were run on a 3.4 GHz Core i3 Central Processing Unit with 4 GB RAM. The compilation environment employed was LabVIEW 7.1.

The initial scenario is depicted in Fig. 5. The ego vehicle (depicted in red) is in the process of overtaking the lead vehicle (yellow) when a faster vehicle (grey) approaches from the ego vehicle's side. Subsequently, the ego vehicle waits, allowing the faster vehicle to pass the lead vehicle. After the adjacent lane is clear, the ego vehicle overtakes the lead vehicle, following the trajectory outlined by the red and blue curves. The red curve indicates the ego vehicle's path generated using the first rule template (initiating from the starting node), whereas the blue curve indicates the path generated by the second rule template (initiating from the goal node).

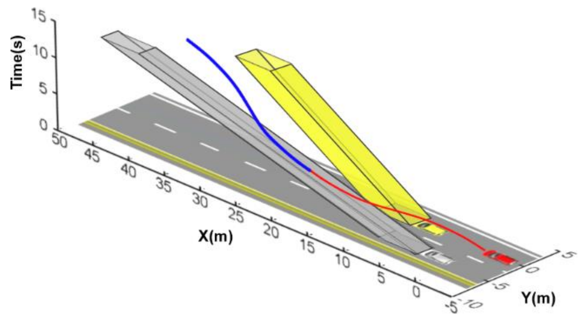


Fig. 5. First test scenario: The ego vehicle (red vehicle) is about to overtake the front vehicle (yellow vehicle) when a faster-moving vehicle (grey vehicle) appears beside the ego vehicle.

The next scenario is illustrated in Fig. 6. In this second scenario, the ego vehicle endeavors to switch lanes to an area occupied by multiple vehicles. The ego vehicle waits for numerous faster-moving vehicles in the target lane to pass before changing lanes. A slightly upward-sloping red line indicates slower movement during this delay. When there is enough space in the intended path to manoeuvre and occupy, the ego vehicle changes lanes, as shown by the nearly parallel blue curve to the horizontal axis - indicating that the car is moving fast.

In the third scenario, the ego's vehicle moves straight into the lane merging area. At the time, a vehicle from the side was moving around to do lane merging on the ego vehicle lane. Then, the ego vehicle adjusts its speed to allow the side vehicles to do lane merging. Then, the ego vehicle moves forward behind those vehicles. This scenario is shown in Fig. 7.

The fourth scenario is shown in Fig. 8. In the fourth scenario, the ego vehicle wants to go through a four-way intersection when another vehicle comes from another lane. Because that other vehicle is moving faster, the ego's vehicle adjusts speed to allow it to pass the intersection (the green line with an upright position means that the vehicle is stopped). Once it is safe, the ego's vehicle moves forward to pass through the intersection. In this fourth scenario, the complete path of the vehicle consists of three colours: red, green and blue. The bi-directional rule templates algorithm generates the red and blue paths. The RRT-ACS algorithm generates the green line because the two existing rule templates cannot connect.

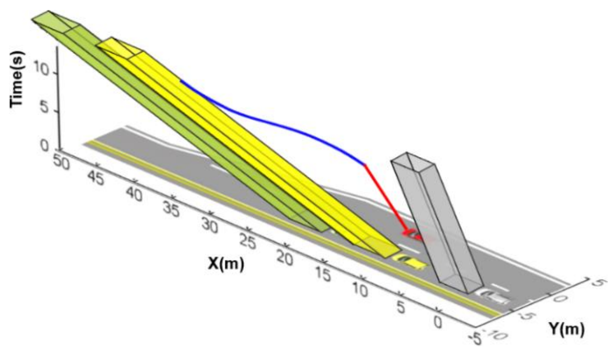


Fig. 6. Second test scenario: The ego vehicle wants to switch lanes where several vehicles occupy the target lane.

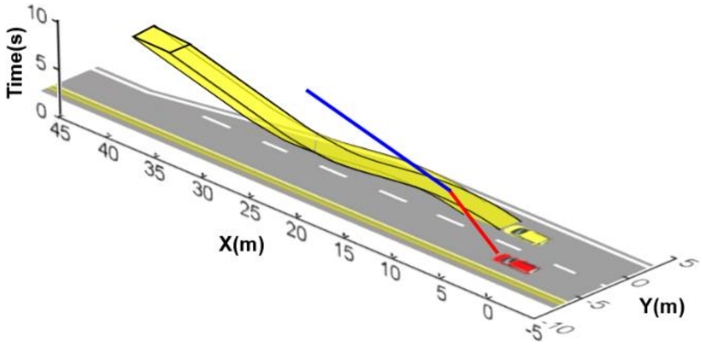


Fig. 7. Third test scenario: The ego's vehicle moved directly into the lane merging area when a side vehicle was merging.

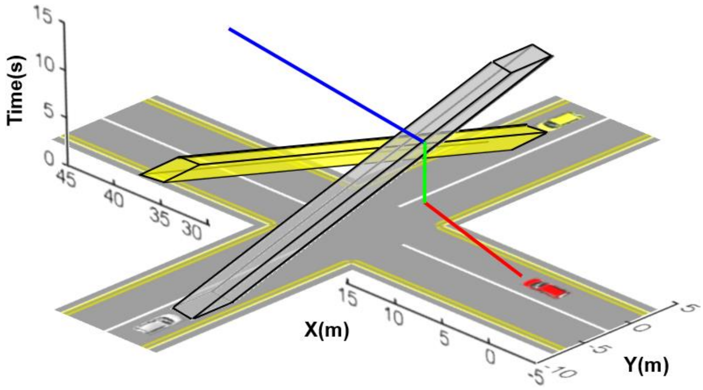


Fig. 8. Fourth test scenario: The ego vehicle wants to go through a four-way intersection when another vehicle comes from another lane.

Table 1 compares the planning times required for the RRT+BRT+CTS, Fast RRT [11], CL-RRT [10], and RRT* algorithms in the first to fourth scenarios. Table 2 shows that the RRT+BRT+CTS average time to carry out the planning process is 98.6% faster than the RRT* algorithm. RRT-ACS planning time was 59.4% faster than the RRT* algorithm in achieving the optimal path [15]. Bi-directional rule templates allow the RRT+BRT+CTS algorithm to plan a path without starting from an empty root, making it faster than the RRT-ACS algorithm. The RRT+BRT+CTS algorithm can also produce an optimal path faster than the Fast RRT because the RRT+BRT+CTS algorithm uses bi-directional rule templates, while the Fast RRT employs one directional rule template.

Table 2 shows that the Fast RRT and CL-RRT algorithms reduce planning time by 93% and 52.3%, respectively, compared to the RRT* algorithm. The results match the performance of the previous research [11], which indicate that the Fast RRT and CL-RRT algorithms have speed improvements of 91.4% and 51.4%, respectively, compared to the RRT* algorithm. Table 2 shows that the RRT+BRT+CTS algorithm outperforms in terms of computation time in a dynamic environment than other algorithms.

This study can be used as a reference for current issue in the vehicle with its automation and urban environment, as discussed in previous studies [18-24].

Table 1. Comparison of each algorithm's planning time.

Scenario	Computation time (in milliseconds) RRT+BRT+ CTS	Fast RRT	CL-RRT	RRT*
First Scenario:				
1. Best time	1.4	15.2	60.8	158.5
2. Mean time	2.3	24.4	88.7	191.6
3. Worst time	3.9	39.4	505.1	508.0
Second Scenario:				
1. Best time	4.8	21.0	135.6	247.8
2. Mean time	7.8	33.1	293.1	659.8
3. Worst time	11.8	48.0	535.9	993.1
Third Scenario:				
1. Best time	3.6	16.1	205.8	426.0
2. Mean time	8.3	36.4	266.6	574.3
3. Worst time	15.0	63.2	631.1	585.5
Fourth Scenario:				
1. Best time	12.1	26.4	435.8	675.1
2. Mean time	20.0	41.2	564.6	1055.5
3. Worst time	31.5	59.1	1125.2	1292.0

Table 2. Comparison of each algorithm's average time relative to the RRT* algorithm.

Scenario	RRT+BRT+ CTS	Fast RRT	CL-RRT
First	98.8%	87.3%	53.7%
Second	98.8%	95.0%	55.6%
Third	98.6%	93.7%	53.6%
Fourth	98.1%	96.1%	46.5%
Average	**98.6%**	**93.0%**	**52.3%**

4. Conclusion

This research introduces local path planning in an environment with dynamic obstacles using the RRT+BRT+CTS strategy. Several simulations of dynamic urban conditions are used to validate the efficiency of the RRT+BRT+CTS strategy. Fast RRT and CL-RRT, well-known path-planning algorithms used in dynamic urban contexts, were compared to RRT+BRT+CTS. The simulation findings reveal that the suggested strategy outperforms speed and efficiency when applied to local path planning in an environment with dynamic obstacles. As a result, the proposed technique is an option for AVs to navigate complex urban situations.

Acknowledgement

The Ministry of Education, Culture, Research, and Technology Republic of Indonesia funded this research under the contract between LLDIKTI4 and Universitas Komputer Indonesia number 010/SP2H/RT-JAMAK/LL4/2023 under the Applied Research scheme for fiscal year 2023.

References

1. Daily, M.; Medasani, S.; Behringer, R.; and Trivedi, M. (2017). Self-driving cars. *Computer*, 50(12), 18-23.

2. Cohen, S.A.; and Hopkins, D. (2019). Autonomous vehicles and the future of urban tourism. *Annals of Tourism Research*, 74(1), 33-42.
3. Ribeiro, M.A.; Gursoy, D.; and Chi, O.H. (2022). Customer acceptance of autonomous vehicles in travel and tourism. *Journal of Travel Research*, 61(3), 620-636.
4. Tranter, K. (2016). The challenges of autonomous motor vehicles for Queensland road and criminal laws. *QUT Law Review*, 16(2), 59-81.
5. Nascimento, A.M.; Vismari, L.F.; Molina, C.B.S.T.; Cugnasca, P.S.; Camargo, J.B.; de Almeida, J.R.; and Hata, A.Y. (2019). A systematic literature review about the impact of artificial intelligence on autonomous vehicle safety. *IEEE Transactions on Intelligent Transportation Systems*, 21(12), 4928-4946.
6. Hu, X.; Zheng, Z.; Chen, D.; and Sun, J. (2023). Autonomous vehicle's impact on traffic: Empirical evidence from waymo open dataset and implications from modelling. *IEEE Transactions on Intelligent Transportation Systems*, 24(6), 6711-6724.
7. Bischoff, J.; Maciejewski, M.; Schlenther, T.; and Nagel, K. (2018). Autonomous vehicles and their impact on parking search. *IEEE Intelligent Transportation Systems Magazine*, 11(4), 19-27.
8. Pendleton, S.D.; Andersen, H.; Du, X.; Shen, X.; Meghjani, M.; Eng, Y.H.; and Ang-Jr, M.H. (2017). Perception, planning, control, and coordination for autonomous vehicles. *Machines*, 5(1), 1-54.
9. Qi, J.; Yang, H.; and Sun, H. (2020). MOD-RRT*: A sampling-based algorithm for robot path planning in dynamic environment. *IEEE Transactions on Industrial Electronics*, 68(8), 7244-7251.
10. Manav, A.C.; and Lazoglu, I. (2021). A novel cascade path planning algorithm for autonomous truck-trailer parking. *IEEE Transactions on Intelligent Transportation Systems*, 23(7), 6821-6835.
11. Ma, L.; Xue, J.; Kawabata, K.; Zhu, J.; Ma, C., and Zheng, N. (2015). Efficient sampling-based motion planning for on-road autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 16(4), 1961-1976.
12. Pohan, M.A.R.; Trilaksono, B.R.; Santosa, S.P.; and Rohman, A.S. (2021). Path planning algorithm using the hybridization of the rapidly-exploring random tree and ant colony systems. *IEEE Access*, 9(1), 153599-153615.
13. Gammell, J.D.; Barfoot, T.D.; and Srinivasa, S.S. (2018). Informed sampling for asymptotically optimal path planning. *IEEE Transactions on Robotics*, 34(4), 966-984.
14. Mashayekhi, R.; Idris, M.Y.I.; Anisi, M.H.; Ahmedy, I.; and Ali, I. (2020). Informed RRT*-connect: An asymptotically optimal single-query path planning method. *IEEE Access*, 8(1), 19842-19852.
15. Pohan, M.A.R.; and Utama, J. (2023). Efficient sampling-based for mobile robot path planning in a dynamic environment based on the rapidly-exploring random tree and a rule-template sets. *International Journal of Engineering*, 36(4), 797-806.
16. Rasekhipour, Y.; Khajepour, A.; Chen, S.K.; and Litkouhi, B. (2016). A potential field-based model predictive path-planning controller for

autonomous road vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 18(5), 1255-1267.

17. Wu, X.; Nayak, A.; and Eskandarian, A. (2021). Motion planning of autonomous vehicles under dynamic traffic environment in intersections using probabilistic rapidly exploring random tree. *SAE International Journal of Connected and Automated Vehicles*, 4(12), 383-399.
18. Bhosale, T.; Attar, A.; Warang, P.; and Patil, R. (2022). Object detection for autonomous guided vehicle. *ASEAN Journal of Science and Engineering*, 2(2), 209-216.
19. Setiyo, M.; Widodo, N.; Purnomo, B.C.; Munahar, S.; Rahmawan, M.A.; and Luthfi, A. (2019). Harvesting cooling effect on LPG-fueled vehicles for mini cooler: A lab-scale investigation. *Indonesian Journal of Science and Technology*, 4(1), 39-47.
20. Eftekhari, S.; and Al-Obaidi, A.S.M. (2019). Investigation of a cruising fixed wing mini unmanned aerial vehicle performance optimization. *Indonesian Journal of Science and Technology*, 4(2), 280-293.
21. Husain, S.S.; Kadhim, M.Q.; Al-Obaidi, A.S.M.; Hasan, A.F.; Humaidi, A.J.; and Al Husaeni, D.N. (2023). Design of robust control for vehicle steer-by-wire system. *Indonesian Journal of Science and Technology*, 8(2), 197-216
22. Ratsame, P.; Koedsaksit, M.; Jitsoontron, S.; Jaisamak, U.; and Puengsungewan, S. (2021). Vehicle-tracking mobile application without a GPS sensor. *Indonesian Journal of Educational Research and Technology*, 1(2), 11-14.
23. Husain, S.S.; Kadhim, M.Q.; Al-Obaidi, A.S.M.; Hasan, A.F.; Humaidi, A.J.; and Al Husaeni, D.N. (2023). Design of robust control for vehicle steer-by-wire system. *Indonesian Journal of Science and Technology*, 8(2), 197-216
24. Aziz, M.; Huda, M.; Nandiyanto, A.B.D.; and Abdullah, A.G. (2018). Opportunity of frequency regulation using electric vehicles in Denmark. *Journal of Engineering, Science and Technology*, 13(6), 1700-1712.