2021 IEEE Intelligent Vehicles Symposium Workshops (IV Workshops)
July 11-17, 2021. Nagoya, Japan

2021 IEEE Intelligent Vehicles Symposium Workshops (IV Workshops) | 978-1-6654-7921-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/IVWorkshops54471.2021.9669253

OpenPlanner 2.0: The Portable Open Source Planner for Autonomous Driving Applications

Hatem Darweesh[1] , Eijiro Takeuchi[1,2] , and Kazuya Takeda[1,2]

***Abstract*— There are few open source autonomous driving planners that are general enough to be used directly, or which could be easily customized to suit a particular application. OpenPlanner 1.0 was introduced back in 2017 to fill this gap. It was developed and integrated with the open source autonomous driving framework Autoware. Since then, many improvements have been introduced, following the original design goals. In this paper, the basic design will be re-introduced along with the latest developed technologies. The new planner is called OpenPlanner 2.0 and includes several new techniques such as multiple HD road network map formats support, trajectory and behavior estimation, planning based HMI support, path generation using kinematics based motion simulation and lane change behavior. OpenPlanner 2.0 is already attracting attention from the autonomous driving research and development communities; universities and companies. Several projects are using and contributing to its code base. Some of these applications will be discussed in this paper as well. A comparison between OpenPlanner and other open source planners showing the aspects where it is superior is also presented.**

I. INTRODUCTION

Autonomous driving requires perception, localization, control and planning. Although there are currently many open-source resources available to researchers for perception, localization and control, there are few open-source planners that are general enough to be used directly, or which could be easily modified to suit a particular application. This is because planning is the core module that connects everything together and it is also application dependent. Planning is also one of the most difficult autonomous driving tasks, as it determines the vehicle's actions, and the safety of the vehicle, its passengers and other road users depend on its correct operation.

To achieve full planning, pipeline inputs such as agent position, detected objects, maps, traffic information and goal location are needed. The output will be a smooth, obstacle free trajectory. Previously developed planners have tended to be either proprietary or open-source. Open-source planners are usually difficult to use, are designed for a specific environment or platform, or lack sufficient documentation or tutorials.

Planning for autonomous driving applications is still a difficult problem however, due to the vast number of possible situations that can occur in dynamic driving environments. Thus, international collaboration is required. The following are the basic development objectives:

[1] Graduate School of Informatics, Nagoya University, Furo-cho, Chikusa-ku, Nagoya, 464-8603, Japan
[2] Tier IV, Inc., Nagoya, Japan

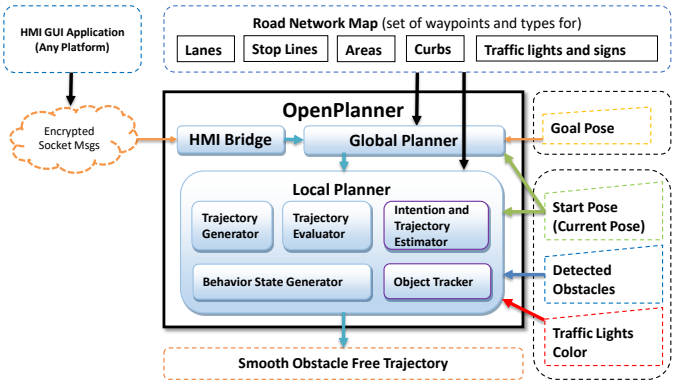


Fig. 1: Architecture of OpenPlanner 2.0

- Fosters international collaboration, so that different development teams can share feedback and solve problems using the same framework.
- Provides a complete autonomous driving software package, include all of the planning modules (global planner, local planner, intention prediction and behavior planner).
- Integrates the various planning modules so that they can be used as separate APIs or within a Robot Operating System (ROS).
- Broad platform support, so that the system will work with wide range of autonomous platforms.
- Broad map support, so that the system will support standard and open-source map formats.
- Broad environment support, so that the system will work indoors, outdoors, on a wide variety of structured streets and off-road.
- Incorporates intention awareness, so that the system is able to predict the movement of other vehicles and plan accordingly.

In OpenPlanner 1.0 [1] we introduced an integrated planner in the sense that it performs global, local and behavior planning. It is an open source planner for the robotics community to take advantage of, use, modify and build on. It was able to utilize HD road network maps [2][3] for both global planning and local planning. The global planning use dynamic programming [4][5] to plan a path from starting position to goal position. The local planner is divided into several modules: trajectory generation, trajectory evaluation and behavior selection (decision making). The planner was evaluated on several robotics platforms working in different

environmental conditions.

Since the release of OpenPlanner we have received many comments and collaboration proposals. Those discussions and practical demands have guided the development of the new algorithms. The utilization of the planner by many international research teams have made many diverse and innovative contributions by applying it to wide range of autonomous driving planning problems around the world. The following are few examples of projects that use OpenPlanner as thier main planning solution:

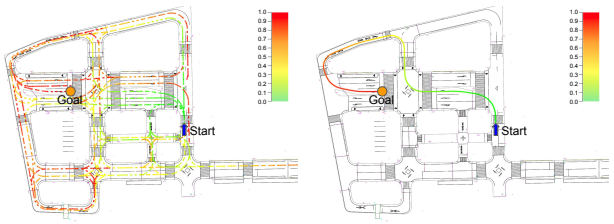
- The Roboat project, The objective of this project is to encourage the use of Amsterdam's canals for transportation, using a fleet of autonomous boats [6][7].
- Developing on campus autonomous golf cart, University of California San Diego, USA [8].
- A testing platform for an autonomous driving map editor, road network map editor developed at the University of Konkuk, Korea [9].
- Chosun University has achieved 5th place in the 2020 University Student Autonomous Driving Contest, Daegu, Korea [10].
- Level 4 fully electric autonomous bus framework developed by ADASTEC [11].

The implementation of OpenPlanner 2.0 is part of the Autoware.AI project [12]. Autoware.AI is an open-source, autonomous driving framework developed by Nagoya University, which is used by many researchers for autonomous driving research and development [13][14]. Autoware.AI is based on the Robot Operating System (ROS) described in [15]. The architecture of OpenPlanner 2.0 is illustrated in Figure 1. It includes a global planner that generates global reference paths using a vector (road network) map. It then includes a Human Machine Interface (HMI) bridge that connects between the HMI client and the global planner. The local planner then uses the global path to generate an obstacle-free, local trajectory from a set of sampled roll-outs. It uses various costs, such as collision, traffic rules, transition and distance from center, to select the optimal trajectory. An intention and trajectory estimator calculates the probabilities associated with other vehicles' intentions and trajectories, while the behavior generator uses predefined traffic rules and sensor data to function as a decision maker.

Section II will explain the global planning and the integration with different mapping standards. Section III will include the HMI bridge design. Section IV will discuss the path generation modifications. The intention and trajectory estimation will be introduced in Section V. The behavior generation including lane change will be explained in Section VI. Finally in Section VII a comparison between OpenPlanner and other open source planners will be discussed with final remarks.

II. Global Planning and HD Map Support

The global planner handles path routing. It takes the HD road network map, a start position and a goal position as input and then finds the shortest or lowest cost path using dynamic programming. Figure 2 shows an example of a global plan constructed in a complex road network map.



(a) (b)

Fig. 2: Searching the map for the shortest path. Line color indicates route cost. Cost is represented by distance, green color is closest distance to Start and red color is max distance reached at the Goal position.

OpenPlanner includes an internal HD road network map structure [16]. It works as an interface layer between any map format and the internal planning functionality. That enables it to utilize other standard open source map formats such as Lanelet2 [17] and OpenDRIVE [18]. Figure 3 shows the general map utilization design. Having custom map model for OpenPlanner is essential to remove any dependencies associated with the mapping special design. The internal map structure consists of: lane center lines, road network connections, stop lines, traffic lights, traffic signs, boundary areas, crossings, curbs, white lines, and road markings.

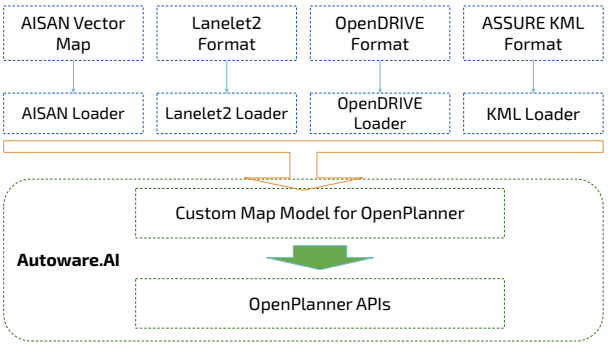


Fig. 3: Map loaders for different map standards.

III. HMI Bridge

A Human machine interface client is important for customized autonomous driving applications such as autonomous taxi, autonomous bus, delivery robots, and many more. The ability to receive and understand commands from a human user or a third party system to change the planning control sequence is essential for any planner. OpenPlanner 2.0 includes such high level messaging system.

As shown in the main architecture in Figure 1 and the bridge design in Figure 4, the HMI client is connected through the HMI bridge to the Global Planner module. This design choice is to minimize the communication load to the local planner. Integrating the HMI state machine to

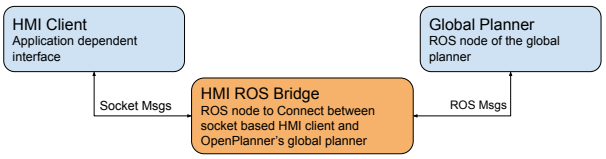


Fig. 4: HMI-OpenPlanner bridge architecture.

the local planner will increase the planning state machine complexity dramatically. We were able to achieve the main required HMI behaviors through the Global Planning stage. The state machine in Figure 5 shows example states that could be changed by default using the current HMI support implementation.

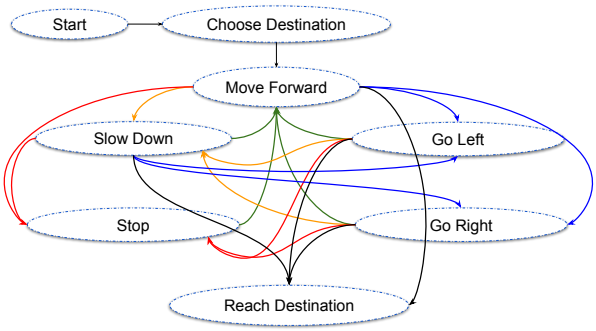


Fig. 5: The basic supported state machine for HMI commands.

IV. Path Generation

First, global path (reference path) is created using the global planner as explained in section II. Then efficient methods of local planning which generate multiple rollouts, starting from the center of a vehicle and running parallel to a reference path, have also been introduced [5][19]. These roll-outs are then linearly sampled and optimized [4].

A. Original Approach

Similar roll-outs sampling based approach has been used in OpenPlanner 1.0 [1]. Figure 6 shows the process of roll-outs sampling from reference path. For path smoothing (optimization step) we use Conjugate gradient (CG) [20].

This approach provided a reasonable performance for mobile robots and low speed autonomous driving with acceptable error margin. To go faster or control a big vehicle such as trucks and buses, more precise path generation is needed. We summarize the outstanding issues with the current path generation approach as follows:

- It does not consider any vehicle parameters (kinematics or dynamics)
- Trajectory smoothing is highly dependent on the waypoints sampling distance
- Generated trajectory is impossible to follow exactly using waypoint following controllers

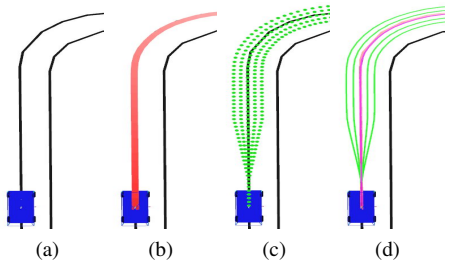


Fig. 6: Steps for generating local trajectories: (a) original map, (b) path section extracted from global path, (c) sampling, (d) smoothing using conjugate gradient.

- Generating smoother trajectories leads to cutting corners especially in the tight corners
- There are big difference between the generated trajectories and the actual vehicle motion path, trajectory evaluation is not accurate

B. Forward/Backward Kinematics based Simulation

To overcome these issues, a new path optimization technique is introduced to replace the CG step. The new approach uses the vehicle kinematics model in forward/backward motion simulation. The optimization targets the curvature of the local trajectory, and the cost function includes the vehicle kinematics. The forward simulation consists of two main steps. First step is the motion step where we use the kinematic model of the vehicle including the steering delay. The second step is the control step where we use P controller to find control signals for each moving step. The backward simulation is currently achieved by switching start/goal points and simulate the vehicle moving on the opposite direction. Flow chart of the developed algorithm is shown in Figure 7.

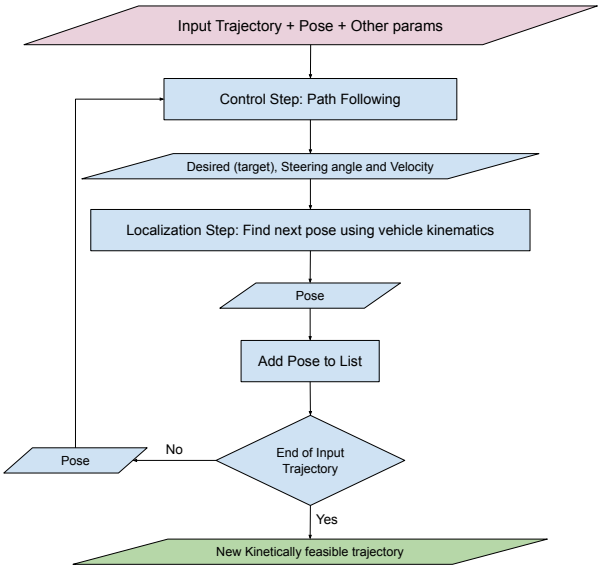


Fig. 7: Forward simulation algorithm flowchart.

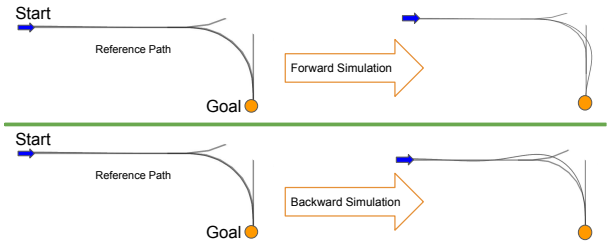


Fig. 8: Example illustrates the final generated path for both forward and backward simulation.

Figure 8 shows the result of using forward and backward simulation to generate path for a tight 90 degree right turn. The introduced approach could also replace the kinematics optimization step for free space path planning algorithms such as RRT* [21] and Hybrid A* [22].

The new method has several advantages, such as:

- Outputs smooth path similar to the optimization techniques without the low performance downside.
- Less parameters to tune compared to other trajectory optimization techniques.
- Once tuned, it will work within the kinematics limits of the vehicle.
- Could plan backward, so it will find the path which can lead to exact target position and orientation.

V. Intention and Trajectory Estimation

Estimating the other vehicle intentions (behaviors) and trajectories is a crucial step for obstacle avoidance and decision making for the autonomous vehicle. In OpenPlanner 1.0 no future prediction is considered. All detected obstacle were treated as static objects which prevented the planner from performing smoothly at high speeds. It also failed to avoid objects that don't share same line of sight with the ego vehicle.

Solutions for this problem were proposed in [23][24], but there are problems with these solutions, such as a lack of generalization for both driving scenarios and sensing models, as well as trade-offs between accuracy and performance. Multiple research methods had inspired us in the development of the new approach such as the use of Bayesian networks and HMMs [25][26] to estimate future states from observed data. Additionally, the algorithm makes efficient use of a multi-cue particle filter similar to [27][28].

The new approach is introduced in previous research [29]. It utilizes a behavior planner working in passive mode, wrapped in a multi-cue particle filter for uncertainty modeling. Using this technique we were able to estimate the probabilities of intentions and trajectories of surrounding vehicles. The architecture of the estimation algorithm is shown in Figure 9. The algorithm consists of four main steps; trajectory extraction, particle sampling, measurement update (weight calculation) and re-sampling.

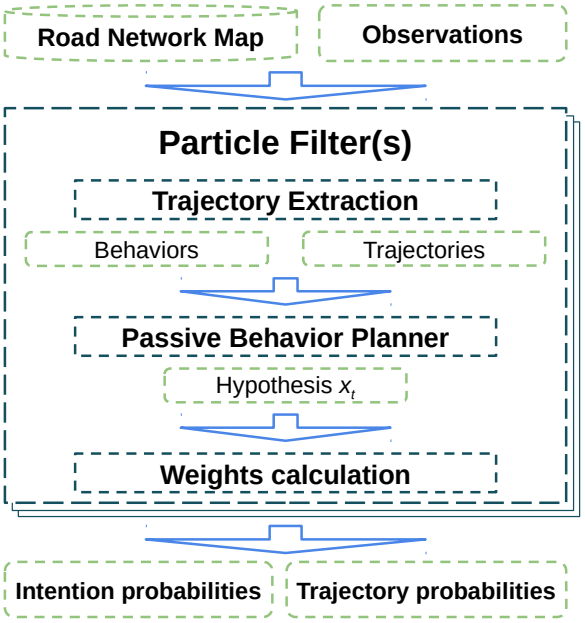


Fig. 9: Proposed intention and trajectory estimation system architecture.

A. Trajectory Extraction

In this step, the algorithm extracts possible driving trajectories from the road network map and then add two additional trajectories to represent branching right and branching left. The left side of Figure 10 shows the center lines of the road network map, while the right side of the figure shows the four extracted trajectories; the two branching trajectories, one for going straight and another one for turning left.

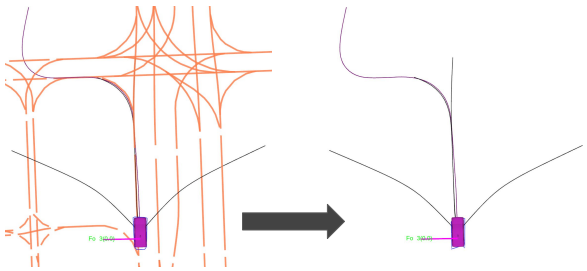


Fig. 10: Extracting all possible vehicle trajectories from a road network map. After all of the existing paths are extracted, the additional branching paths (right) are added.

B. Particle Sampling

In this step, it samples new particles for new behaviors and trajectories, thus when a new trajectory appears in a scene, new particles need to be generated. Figure 11 shows the state network representing the intentions of other vehicles and the transitions between them. Extracting the trajectory from the previous step helps us to represent the initial probability distribution needed to achieve particle sampling. When a new

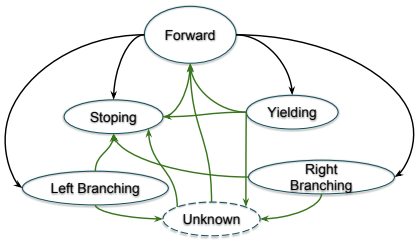


Fig. 11: Intention states to be estimated.

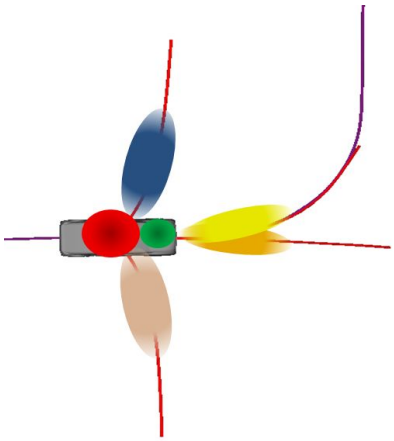


Fig. 12: Prior sampling distribution, manually designed with fixed parameters for the particle filter's motion model initial condition.

trajectory appears, a new set of particles are sampled from the initial prior distribution in Figure 12.

C. Weight Calculation

In this step, the sampled particles are filtered by calculating how far they are from the original distribution and sensing cues. The weight for each particle is calculated against the sensing data, such as position, direction, velocity, acceleration and turn signal information. It is important to take sensing noise into consideration; that is why sensing data is not used directly and Gaussian distribution-based noise is added at each step.

D. Re-Sampling

In this step, particles with small weights are removed and are replaced with new samples drawn from particles with the highest confidence level. This sampling step relies on the motion model that estimates the path the observed vehicle is expected to follow, which in this study is the behavior planner. It estimates the motion and intentions of other vehicles according to the observed circumstances.

VI. Lane Change Behavior

Lane change is one of the most important autonomous driving behaviors, though it was not fully supported in OpenPlanner 1.0. It was only supported in the global planning step, once the lane change point is estimated the local planner had to strictly follow the reference plan. In case of any object is blocking the target lane, the planner will just stop and wait for the target lane to become free again. Which could cause traffic deadlock. In the new planner, we implemented the support of lane change inside the local planning stages.

Figure 13 shows how this solution is handled. First, the global planner generates reference plan which includes global paths, lane change points and backup reference paths. The global planner is responsible for assigning lower cost to the target lane's path. Second, the trajectory generator creates smooth and kinematically feasible trajectories for all possible paths. Third, the trajectory evaluator tries to select best lane and best roll-out trajectory inside that lane. Since the cost of the target lane is always lower than other lanes, the local planner will always push the vehicle to switch lanes whenever possible. Finally, the behavior selector will monitor the current lane and target lane, if it can't reach target lane on time, it will send a re-planning signal to the global planner. At the re-plan step, the global planner either extends the backup path or find another route to the goal.

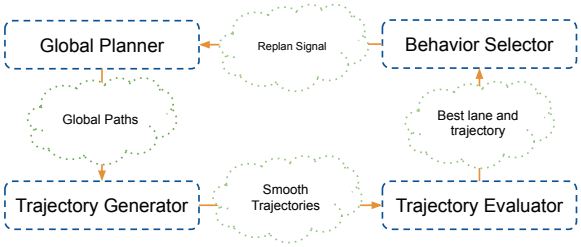


Fig. 13: Integrating lane change process to the local planning loop.

VII. Conclusions

A good planner for autonomous driving should be complete, including all essential planning components, easy to use with simple design, tutorials and support should be available. Also, It must be portable to be able to support multiple autonomous platforms running in different environments. It should support a wide range of standard open source maps. Finally it should provide intuitive parameters; enabling the users to choose between high performance and high accuracy depending on the computing platform they use.

OpenPlanner 1.0 and the new version OpenPlanner 2.0 provide such a recipe for a successful planner for autonomous agents in general. Table 14 shows a comparison between OpenPlanner and several open source planners.

Currently, the OpenPlanner team is working through the Autoware-OpenPlanner working group, under the umbrella of Nagoya University and the Autoware Foundation to ensure the continuation of development. Other objectives of this working group are:

- Autoware.AI and OpenPlanner community support
- Discussing the state of the art research in the planning field and help researcher test their hypothesis then include the successful techniques into Autoware.AI.

Planner / Feature	Open Motion Planning Library (OMPL)	ROS Navigation Stack	Open Robotics Design & Control Open-RDC	Mobile Robot Programming Toolkit MRPT	Apollo AD Planner	OpenPlanner 2.0
Platform Support	Robots	Robots	Robots	Robots	Autonomous Vehicles	Robots + Autonomous Vehicles
Environment Support	Indoor	Indoor, Side-alk	Indoor, Sidewalk	Indoor	Pre-driven structured roads	Indoor, outdoor, public road, highway
Library APIs	Independent	-	-	Independent	-	Independent
ROS Support	Yes	Yes	Yes	Via bridge	-	Yes
Integration	-	-	-	-	Apollo	Autoware.AI
Map Support	Cost map + point cloud	Cost map	Cost map	Cost map + point cloud	Customized OpenDRIVE	Vector Map, KML, OpenDRIVE, Lanelet2
Global Planning	Yes	-	-	-	Yes	Yes
Local Planning	Yes	Yes	Yes	Yes	Yes	Yes
Behavior Planning	Custom	-	-	-	Yes	Yes

Fig. 14: Comparison of OpenPlanner with leading open-source planners.

- Move the latest OpenPlanner development into the Autoware.Auto project [30].

Addtional work is planned to be included into OpenPlanner such as:

- Use Markov Decision Processes (MDP) instead of deterministic trajectory evaluation.
- Add new behavior sates such as overtake and yield.

ACKNOWLEDGMENT

This research was supported by the OPERA project of the Japan Science and Technology Agency and by Nagoya University.

REFERENCES

[1] H. Darweesh, E. Takeuchi, K. Takeda, Y. Ninomiya, A. Sujiwo, L. Y. Morales, N. Akai, T. Tomizawa, and S. Kato, "Open source integrated planner for autonomous navigation in highly dynamic environments," *Journal of Robotics and Mechatronics*, vol. 29, no. 4, pp. 668–684, 2017.

[2] P. Czerwionka, M. Wang, and F. Wiesel, "Optimized route network graph as map reference for autonomous cars operating on german autobahn," in *The 5th International Conference on Automation, Robotics and Applications*. IEEE, 2011, pp. 78–83.

[3] M. Haklay and P. Weber, "Openstreetmap: User-generated street maps," *IEEE Pervasive Computing*, vol. 7, no. 4, pp. 12–18, 2008.

[4] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, *et al.*, "Junior: The stanford entry in the urban challenge," *Journal of field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.

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

[6] W. Wang, B. Gheneti, L. Mateos, F. Duarte, C. Ratti, and D. Rus, "Roboat: An autonomous surface vehicle for urban waterways," *IEEE Robotics and Automation Letters (submitted)*, 2019.

[7] M. . AMS. (2019) Roboat project. https://roboat.org. [Online; accessed December-2019].

[8] D. Paz, P.-J. Lai, S. Harish, H. Zhang, N. Chan, C. Hu, S. Binnani, and H. I. Christensen, "Lessons learned from deploying autonomous vehicles at uc san diego," in *Field and Service Robotics*. Springer, 2021, pp. 427–441.

[9] W. N. Tun, S. Kim, J.-W. Lee, and H. Darweesh, "Open-source tool of vector map for path planning in autoware autonomous driving software," in *2019 IEEE International Conference on Big Data and Smart Computing (BigComp)*. IEEE, 2019, pp. 1–3.

[10] I. Korean Ministry of Trade and Energy. (2020) 2020 University Student Autonomous Driving Contest. http://autonomouscar.or.kr/. [Online; accessed May-2021].

[11] ADASTEC. (2021) Level 4 Autonomy Platform. https://www.adastec.com/. [Online; accessed May-2021].

[12] T. A. Foundation. (2020) Autoware.AI. https://www.autoware.ai/. [Online; accessed May-2021].

[13] S. Kato, E. Takeuchi, Y. Ishiguro, Y. Ninomiya, K. Takeda, and T. Hamada, "An open approach to autonomous vehicles," *IEEE Micro*, vol. 35, no. 6, pp. 60–68, 2015.

[14] S. Kato, S. Tokunaga, Y. Maruyama, S. Maeda, M. Hirabayashi, Y. Kitsukawa, A. Monrroy, T. Ando, Y. Fujii, and T. Azumi, "Autoware on board: Enabling autonomous vehicles with embedded systems," in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2018, pp. 287–296.

[15] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3. Kobe, Japan, 2009, p. 5.

[16] ZATiTECH. (2020) OpenPlanner Internal Map Format Documenation. https://github.com/hatem-darweesh/assuremappingtools/tree/master/docs. [Online; accessed May-2021].

[17] F. Poggenhans, J.-H. Pauls, J. Janosovits, S. Orf, M. Naumann, F. Kuhnt, and M. Mayr, "Lanelet2: A high-definition map framework for the future of automated driving," in *Proc. IEEE Intell. Trans. Syst. Conf.*, Hawaii, USA, November 2018.

[18] VIRES. (2019) OpenDRIVE Project. http://www.opendrive.org/. [Online; accessed December-2019].

[19] X. Li, Z. Sun, A. Kurt, and Q. Zhu, "A sampling-based local trajectory planner for autonomous driving along a reference path," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*. IEEE, 2014, pp. 376–381.

[20] Y. Saad, *Iterative methods for sparse linear systems*. siam, 2003, vol. 82.

[21] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[22] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *The International Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, 2010.

[23] S. Brechtel, T. Gindele, and R. Dillmann, "Probabilistic decision-making under uncertainty for autonomous driving using continuous pomdps," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2014, pp. 392–399.

[24] P. Liu, A. Kurt, *et al.*, "Trajectory prediction of a lane changing vehicle based on driver behavior estimation and classification," in *17th international IEEE conference on intelligent transportation systems (ITSC)*. IEEE, 2014, pp. 942–947.

[25] Z. Ghahramani, "An introduction to hidden markov models and bayesian networks," in *Hidden Markov models: applications in computer vision*. World Scientific, 2001, pp. 9–41.

[26] H. Bai, S. Cai, N. Ye, D. Hsu, and W. S. Lee, "Intention-aware online pomdp planning for autonomous driving in a crowd," in *2015 ieee international conference on robotics and automation (icra)*. IEEE, 2015, pp. 454–460.

[27] S. Vacek, S. Bergmann, U. Mohr, and R. Dillmann, "Rule-based tracking of multiple lanes using particle filters," in *2006 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*. IEEE, 2006, pp. 203–208.

[28] C. Shen, A. Van den Hengel, and A. Dick, "Probabilistic multiple cue integration for particle filter based tracking," *Australian Pattern Recognition Society*, 2003.

[29] H. Darweesh, E. Takeuchi, and K. Takeda, "Estimating the probabilities of surrounding vehicles intentions and trajectories using a behavior planner," *International journal of automotive engineering*, vol. 10, no. 4, pp. 299–308, 2019.

[30] T. A. Foundation. (2020) Autoware.Auto. https://www.autoware.auto/. [Online; accessed May-2021].