# Real-Time Motion Planning With Dynamic Obstacle Avoidance for Autonomous Vehicles

Pinak Jani, and Krutarth Trivedi,
Master's in Robotics Engineering,
*Worcester Polytechnic Institute, Worcester, MA 01609 USA*

***Abstract*—The self-driving car requires a robust motion planner for navigating itself in the given structured environments such as roads or highways and unstructured environments such as parking lots. In structured environments, some constraints such as maintenance of high speed, the presence of curvature roads, and specific driving rules exist. On the other hand, in unstructured environments, there is no lane information to guide or constrain the actions of the vehicle. To navigate an autonomous vehicle in both these environments, global route planning with local planning for obstacle avoidance with static and dynamic obstacles is addressed.**

## I. Introduction

SINCE the last decade, autonomous vehicles are attracting considerable attention from academia, industry, and governments all over the world. Generally, the autonomous vehicle is structured in the following main layers: Perception, Sensor Fusion, Decision, and Control. When autonomous vehicles advance toward realistic road traffic, they are required to be capable of handling various complex maneuvers, such as lane-keeping, vehicle following, lane-changing, lane-merging, avoiding both static and dynamic objects, and interacting with other traffic participants while complying with the traffic rules. Developing an autonomous vehicle to have these functionalities requires the systematic integration of state-of-the-art technologies in perception, localization, decision making, motion planning, and control. As one of these core technologies, motion planning plays a vital role as a structured hierarchy that includes global route planning, behavioral planning, and local planning. The global route planner generates a global route that creates the collision-free way-points in the path to reach the destination point. The behavior planner decides if the vehicle needs to change or follow the lane, turn or drive straightly, and stop or go according to the planned route while considering the real-time traffic information. The local planner is responsible for checking collision detection, and dynamic obstacle-avoidance.

## II. Background

Motion Planning environment begins by development of Configuration Space, which is classified into the collision space, uncertain space and the free space. The collision space determines the obstacles in the configuration space, with the uncertain space giving the probabilities for collision. The ego, which is the controlled and sensors-equipped vehicle, then utilizes the free space to plan a road map. There are two basic approaches to planning a road map:

(1) Sampling based Approach

(2) Deterministic Approach

The Sampling-based approach takes samples of nodes from the substantial set of nodes in an environment and constructs the road map. It utilizes techniques such as the Probabilistic Road Map method (PRM) [1] and Rapidly Exploring Random Search Trees method (RRT or RRT*) [14] which can accommodate for the dynamics of the vehicle during the global road map generation. These algorithms sample points in the graph formed by prior knowledge.The deterministic approach utilizes techniques such as visibility graphs, Voronoi diagrams, and cell decomposition which enables representation of an environment in the form of a graph. Then using the A* [14] or Dijkstra algorithm [14], the search for the path with the minimum cost from the starting point to the target point is done.

In this project, we propose to utilize the Sampling based framework for generating a probability road map. This technique typically lacks the dynamic path planning capabilities and hence motion planning needs to be done at multiple levels. The global route planning level generates a global route that creates the collision-free way-points to reach the destination point. Once this path is planned, a local planner implements an algorithm which dynamically modifies the path planned by the the global route level planner if either any static and/or dynamic obstacle obstruct the path or lane-maneuvering has to be performed.

## III. Scope of Work and Goals

In terms of Autonomous Vehicles functionality, the main focus of the proposed project is on global route planning and local route planning using state-of-the art motion planning algorithms. The two problems being dealt with through this project are the static and dynamic obstacle avoidance for both, structured and unstructured environments.

Corresponding Authors: Pinak Jani(pjani@wpi.edu), and Krutarth Trivedi(ktrivedi@wpi.edu) Co-Authors: Wael Mohammed (wmohammed@wpi.edu), and Puru Upadhyay (pupadhyay@wpi.edu)
The proposed project is an intrinsic component of the work - "Real-time Motion Planning for Autonomous Vehicles" and some parts are shared with the co-authors who are proposing a project - "Real-Time Motion Planning With Lane Maneuvering for Autonomous Vehicles"

### A. Structured Environment

Since for a structured environment, autonomous vehicles are being made to follow lane-divided roads featuring unidirectional flows, some of the tasks that are to be done in this regard are as follows:
- Car following: Along with lane keeping, the ego must follow the vehicle in front of it with a safe gap when lane changing is not being done.
- Lane keeping: Maintaining sufficient gaps in all directions of the ego while being within the lane marks for safety.
- Lane changing: This is done under either directional or obstacle constraints. The motion planner must ensure that the space in the target lane is sufficient and that the speed is adequate to keep the ego in a safe state.
- Passing: The ego vehicle respects a lane keeping or car following decision while obstacles are in the adjacent lanes.
- Overtaking: This maneuver consists of a lane change, then passing a vehicle or an obstacle, and finally another lane change to return to the previous ego lane.
- Static and Dynamic obstacle avoidance: This particular component deals with the situation where the ego vehicle should avoid collision with static and/or moving obstacles. The ego vehicle will need to recalculate the detouring path and to steer itself toward a safe and efficient path in real time in order to reach the destination.

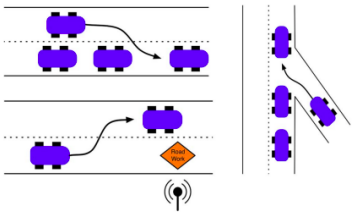


Fig. 1: Lane Change Maneuvering

### B. Unstructured Environment

For an unstructured environment, lane structure is not available. Hence, the most important task that is to be done in this regard is the navigation through the static and/or dynamic obstacle where the ego must reach a given goal while avoiding obstacles like curbs, parked cars and so on.

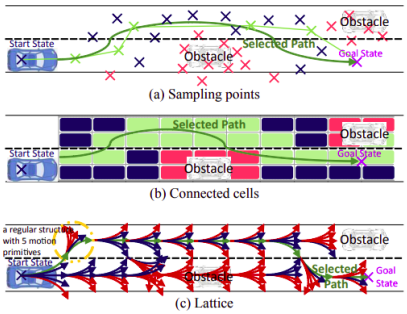



Fig. 2: Static Obstacle Avoidance

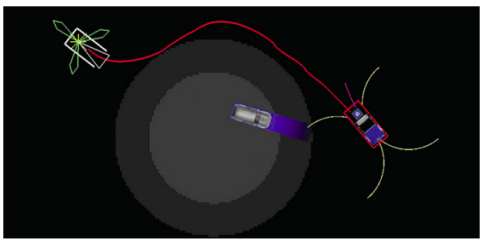


Fig. 3: Dynamic Obstacle Avoidance

## IV. Methodology

In this section, the framework of the proposed algorithm is presented.

### A. The Overall Structure

An overview of the software system architecture for the autonomous driving with obstacle avoidance is outlined in figure 3:

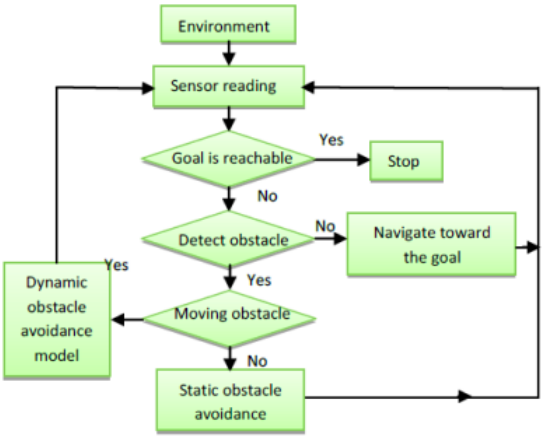



Fig. 4: Structure Diagram

### B. Motion Planning

The motion planning layer provides a plan to achieve a particular desired goal considering the current and predicted vehicle states. The desired goal could be anything like traversing a particular lane, avoiding a particular obstacle or following another vehicle. The motion planner executes these planning considering the behavioral constraints in the environment and provides optimal solution to achieve the desired goal. The motion planner executes these goals by generating a set of trajectories using trajectory generating algorithms.

*1) Global Planner:* The global planner calculates the shortest route from the initial position to the goal position using the given or generated map of the surrounding environment. In order to do this, we sample the environment with random points or (nodes of the graph), and then we draw edges between these nodes choose the edges which are collision free, lastly using the Dijkstra discrete Search algorithm [14] we find the shortest path using these sample nodes while considering the collision checker algorithm on each step.

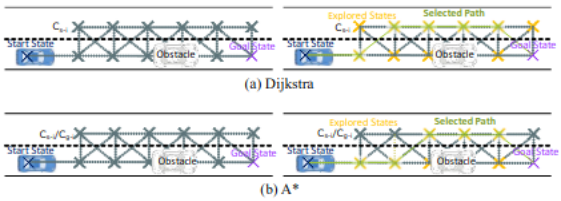



Fig. 5: Illustrations of the processes of (a) Dijkstra, (b) A*

*2) Local Planner:* Once the way-points are decided by the global planner, the next layer of the motion planner is the local planner. The local planner predicts the future states based upon the control inputs to an ego vehicle to perform out the global map, and simultaneously takes care of close-in obstacle avoidance, even if it was not provided by the global plan. In order to achieve this task, we develop a state lattice planner using Hybrid A* discrete search algorithm [15]. This planner functions in accordance with an obstacle map of all the surrounding obstacles created by the collision checker algorithm. If the obstacle is close to the proximity of an ego vehicle or if the predicted future trajectory of an ego vehicle lies in the close proximity of the obstacle then that trajectory is encoded as a hard constraint and fed to the local planner to strictly avoid. In such a case, the local planner finds an optimal path to reach to the next possible way-points on the global map.

*3) Collision Checker:* In order to avoid static and/or dynamic obstacles during both, global and local trajectories exploration, we develop a collision checker algorithm based on convex hull [16]. To make the collision-free way-points search more efficient in terms of obstacle avoidance, we pad all the obstacles with our ego vehicle's dimensions using the Minkowski Sum method [14].

### C. Structured Planning

The structured planning algorithm is responsible for navigating an ego vehicle in the structured environment such as in on-road driving scenarios. The main objective of this algorithm is to drive an ego vehicle in the desired lane avoiding obstacles, keeping a keep-safe distance from the surrounding cars, and if the need arises, plan a safe and collision-free trajectory to overtake the front vehicle. This involves use of a high-speed multi-layer motion planner having the vehicle dynamics, Ackerman steering [13] and dynamic feasibility.

### D. Unstructured Planning

This includes planning in case of unstructured motion goals that arise when the vehicle encounters anomalies in the environment which may be a collision with obstacles or a complex maneuver during the off-road driving scenario. Since, the environment is unstructured, in this case the movement of vehicle is far less constrained compared to on-Road planning scenario. The main aim of this algorithm is to navigate an ego vehicle in given environment while considering static and dynamic obstacle avoidance. This planning also prevents the vehicle from reaching undesirable pose or areas considering the vehicle dynamics and thereby allowing local maneuvering and dynamic obstacle avoidance.

## V. Implementation

In our implementation, We have used pygame [11] and Carla [12] as our primary working environments. However, with the increasing complexity of adding dependencies and slow performance in Carla, later we used pygame extensively for most of the dynamic obstacle avoidance tasks. The development of the environment and programming is done using Python3.8. The collaboration for the reported work is done using LaTeX and Overleaf, and the source code is shared on GitHub.

*1) Implementation in Carla:* For implementing a global route planner in CARLA [12], we first spawned the default world map along with an ego vehicle and instantiated it with the start location and the goal location. The next step was to get the way-points from the environment and store it locally in a list. These way-points will essentially function as grid locations in the given grid environment, and it would make the task easier for our search algorithms. For global route planner, we have used discrete graph based search algorithms - Dijkstra. In addition to this, we have also added vehicle constraints for our ego vehicle that uses Ackerman steering [13].

*2) Implementation in pygame Environment:* We have developed different environments for structured and unstructured scenarios using pygame [11]. In the case of a structured environment, we developed a 3-lane highway scenario with static and dynamic obstacles, and an Ego vehicle is simulated using Non-holonomic Constraints. Since, the current environment is based upon a highway scenario we assumed that the car moves with a constant speed on a straight road with static obstacles. Along with that, we also developed an unstructured environment with random static obstacles and dynamic obstacles making the planning more challenging. The implementation starts with generating the the map with static and dynamic obstacles and then spawning the ego vehicle. Then the global path planner searches for a path in the map to avoid the static obstacles and reach the goal position. Upon finding this path the ego vehicles start the action and to navigate the global path using the local planner mentioned earlier. The local planner navigates using the Ackerman Steering [13] through the way-points of global path by avoiding dynamic obstacles.

## VI. Results

We plan to evaluate the proposed motion planner by creating multiple test cases for both structured and unstructured environments. The link to the video demonstrating the working implementations and results is attached below: https://drive.google.com/drive/folders/1aRK2Fd6aWWbsFN9GD2YhZe7p4_Y79ir3?usp=sharing

Fig. 6: Simulation worlds in Carla

### A. Carla Results

The complete setup of the simulation environment is done in CARLA [12]. A suitable world map is selected and a non-holonomic vehicle is spawned at an initial position. A final position is selected and a global path planner is used to generate way-points from the initial to the final position. By using the VehiclePIDController, the ego vehicle is made to follow the way-points with constant speed and stop at the final way-point.

### B. pygame Environments Results

In order to validate the robustness of the developed algorithms, we performed exhaustive experiments for the following test cases in pygame [11]:
1) Navigation in an unstructured environment with randomly placed static and dynamic obstacles
2) Lane maneuvering with static obstacle

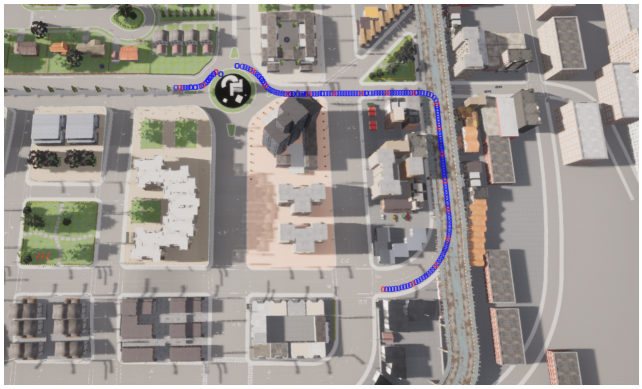


Fig. 7: Optimal paths generated using Dijkstra's Planner in CARLA

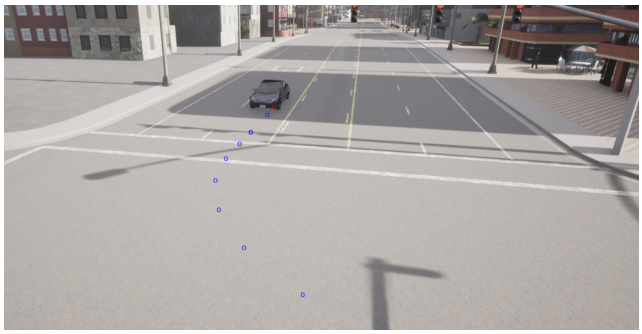


Fig. 8: Spawned vehicle following way-points in CARLA

3) Lane maneuvering with both static and dynamic obstacle
4) Navigation in an unstructured environment replicating the parking space.

Here, in all the results, the green blocks represent the static obstacles and the black blocks represent the dynamic obstacles, and the ego vehicle is represented as the blue block. The way-points and the global route are denoted as blue dots and the green lines connecting them respectively. Figure 11 presents the very first test case in which all the static obstacles are randomly initialized and the dynamic obstacles are spawned in real-time during the run-time simulation. Figure 12 shows the test case of a structured environment where an ego vehicle plans an optimal global path to reach the goal state avoiding static obstacles. Figure 13 explains the overtaking scenarios for on-road planning. Lastly, an environment replicating the parking space is shown in figure 14, where an ego vehicle maneuvers itself avoiding both parked cars and moving cars.

## VII. Challenges

Following are the challenges that we ran into while implementing this project.

*1) Setting up of CARLA:* Since none of the team members have had prior experience in CARLA, its software setup was a big challenge that kept throwing errors for a long time. During the installation process, some files were not present and needed to be downloaded separately. Various system path errors for including modules were also faced and resolved.

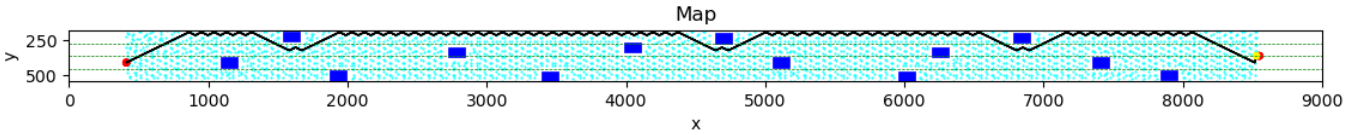



Fig. 9: Dijkstra Planner in a basic Pygame environment

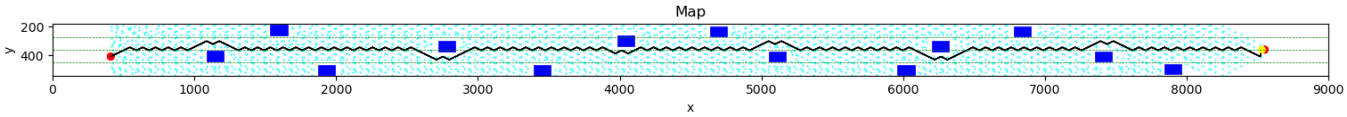



Fig. 10: A* Planner in a basic Pygame environment

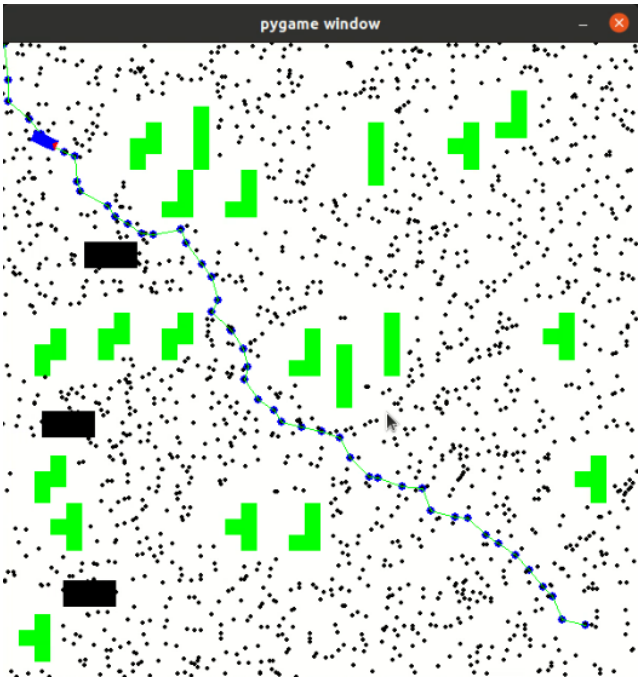



Fig. 11: Motion Planning in a randomly initialized unstructured Environment in Pygame

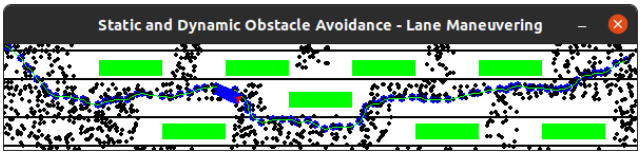



Fig. 12: Motion Planning in an on-road scenarios with static obstacles in Pygame

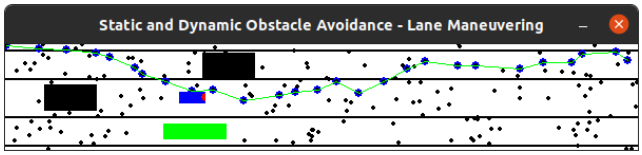



Fig. 13: Motion Planning in an on-road scenarios with static and dynamic obstacles in Pygame

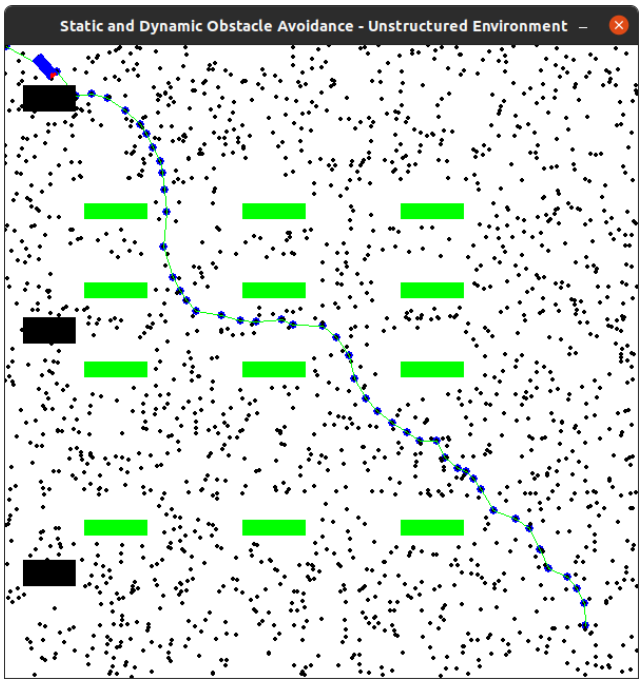



Fig. 14: Motion Planning in a parking space in Pygame

Some amount of time was needed to become familiar with the CARLA code base.

*2) Generation of way-points:* After obtaining the initial and final position of the vehicle, we faced some issues in obtaining the needed way-points between the two. The particular cause of the error was the usage of a different type of data than the earlier initial and final points we selected. This was resolved by converting the selected points to the needed data type.

*3) Movement of Vehicle:* On obtaining the way-points, the movement of the ego vehicle along these way-points was to be done. In Carla, VehiclePIDController was found and used to move the ego vehicle. The various parameters it needed were studied and its PID values were tuned to obtain desired motion.

*4) Vehicle not stopping:* In Carla, Once the vehicle started its motion along the way-points, it was failing to stop at the final way-point. In order to overcome this issue, the control signal was again applied for the last way-point with zero throttle.

*5) Errors in the ego vehicle action:* In Pygame, Although the global path is generated but upon execution of action using the vehicle kinematics the ego vehicle tends to collide into obstacles due to mismatch in the real-time simulation of ego vehicle.

*6) Obstacle Avoidance:* We tried several methods in implementing the obstacle avoidance algorithm some of them were computationally very expensive and considering fast planning requirement we finally came up with a convex hull based algorithm using Shapely library in Python.

*7) Dynamic Obstacle Environment:* Developing an environment with dynamic obstacles was the main challenge as we did not have an environment available for even a basic implementation. We planned to find or develop a dynamic environment using CARLA but were unsuccessful in doing so thus we switched to Pygame and developed the environment from scratch.

*8) Lattice Planner for Local Planning:* We faced some issues in implementing the Lattice Planner for local planning of the ego vehicle to avoid unforeseen dynamic obstacles in the way-points and traverse the most efficient obstacle-free path. These issues were some minor logic errors that were solved upon debugging.

## VIII. Conclusion

Based on the performed experiments and obtained results, we conclude that by using a single multi-level planner with a sampling-based global planner and a state lattice local planner, we could achieve efficient path planning in case of both structured and unstructured scenarios. It was interesting to observe that using PRM as the global planner, we could also achieve multi-query planning with better accuracy, especially for the parking scenarios and a complex on-road maneuvering is also possible to achieve effectively using the state lattice as the local planner with a convex hull intersection checking algorithm.

## IX. Appendix

### A. Krutarth Trivedi

Krutarth Trivedi is fairly experienced in C++ and Python. His experience with data structure and algorithms has helped the team understand the software Framework better, which enabled the team make progress in implementing the programs. Krutarth's work involved initialization of a world map environment based on which the source code for creating the global route planner was developed, and later, helped with the test and validation of the search algorithms in pygame.

### B. Pinak Jani

Pinak Jani has experience with pygame using Python. His experience helped the team to understand how to create a pygame environment for the highway scenario along with static obstacles. Pinak's work involved working on vehicle dynamics and implementing the same for an ego vehicle in CARLA. He also worked on initialization of a pygame environment and development of search algorithms like A* and Dijkstra, and later, he helped with the test and validation of the global route planner in CARLA.

## References


[1] Xiaohui Li, Zhenping Sun, Dongpu Cao, Zhen He, and Qi Zhu, *Real-Time Trajectory Planning for Autonomous Urban Driving: Framework, Algorithms, and Verifications*. IEEE/ASME TRANSACTIONS ON MECHATRONICS, Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=arnumber=7303933

[2] Laurène Claussmann, Marc Revilloud, Dominique Gruyer, Sébastien Glaser, *A Review of Motion Planning for Highway Autonomous Driving*, IEEE Transactions on Intelligent Transportation Systems, IEEE, 2019, 21, pp 1826-1848. ff10.1109/TITS.2019.2913998ff. ffhal-02923507f

[3] *https://courses.cs.washington.edu/courses/cse571*

[4] Farah Kamil, *A Review on Motion Planning and Obstacle Avoidance Approaches in Dynamic Environments*, Advances in Robotics and Automation · January 2015

[5] Xiaohui Li, Zhenping Sun, Dongpu Cao, Zhen He, and Qi Zhu, *Real-Time Trajectory Planning for Autonomous Urban Driving: Framework, Algorithms and Verifications*, IEEE/ASME TRANSACTIONS ON MECHATRONICS, VOL. 21, NO. 2, APRIL 2016

[6] Zhiqiang Jian, Songyi Zhang1, Shitao Chen1, Xin Lv1, Nanning Zheng, *Real-Time Trajectory Planning for Autonomous Urban Driving: Framework, Algorithms and Verifications*, 2019 IEEE Intelligent Vehicles Symposium (IV) Paris, France. June 9-12, 2019

[7] David Bevly, Xiaolong Cao, Mikhail Gordon, Guchan Ozbilgin, Student Member, IEEE, David Kari, Brently Nelson, Jonathan Woodruff, Matthew Barth, Fellow, IEEE, Chase Murray, Arda Kurt, Member, IEEE, Keith Redmill, Senior Member, IEEE, and Umit Ozguner, Fellow, IEEE, *Lane Change and Merge Maneuvers for Connected and Automated Vehicles: A Survey*, IEEE TRANSACTIONS ON INTELLIGENT VEHICLES, VOL. 1, NO. 1, MARCH 2016

[8] Jang-Ho Cho, Dong-Sung Pae, Myo-Taeg Lim,and Tae-Koo Kang,*A Real-Time Obstacle Avoidance Method for Autonomous Vehicles Using an Obstacle-Dependent Gaussian Potential Field*, Journal of Advanced Transportation, vol. 2018, Article ID 5041401, 15 pages, 2018. https://doi.org/10.1155/2018/5041401

[9] Sascha Kolski, Dave Ferguson,Mario Bellino, and Roland Siegwart, *Autonomous Driving in Structured and Unstructured Environments*

[10] Dahhmani/motion-planning-for-self-driving-cars: ENPM661 planning for Autonomous Robots *https://github.com/dahhmani/Motion-Planning-for-Self-Driving-Cars, journal=GitHub, author=Dahhmani*

[11] Pygame *https://www.pygame.org/wiki/GettingStarted*

[12] Carla Simulator *https://carla.org/*

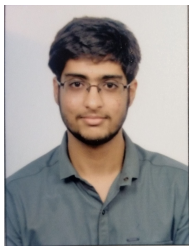
[13] Ackermann Steering *https://datagenetics.com/blog/december12016/index.html*

[14] S. M. LaValle, Planning Algorithms, New York: Cambridge University Press, 2006. *http://lavalle.pl/planning/*

[15] Sedighi, S., Nguyen, D.V. and Kuhnert, K.D., 2019, April. Guided hybrid A-star path planning algorithm for valet parking applications. In 2019 5th international conference on control, automation and robotics (ICCAR) (pp. 570-575). IEEE. *https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8813752*

[16] Barber, C.B., Dobkin, D.P. and Huhdanpaa, H., 1996. The quickhull algorithm for convex hulls. ACM Transactions on Mathematical Software (TOMS), 22(4), pp.469-483. *https://dl.acm.org/doi/abs/10.1145/235815.235821*

## X. Biography Section



**Pinak Jani** received the Bachelor's degree in Electronics and Communication Engineering from KJ Somaiya College Of Engineering, Maharashtra, India in 2021 and currently pursuing Master's in Robotics at Worcester Polytechnic Institute, MA, USA.

His research interests include Robot Vision, Control and Motion Planning for Autonomous Vehicles.

**Krutarth Trivedi** received the Bachelor's degree in Electronics and Communication Engineering from Gujarat Technological University, Gujarat, India in 2018 and currently pursuing Master's in Robotics at Worcester Polytechnic Institute, MA, USA.

His research interests include Perception, Sensor Fusion, Motion Planning, and Intelligent Control for Autonomous Vehicles.