Multi-Vehicle Coordination for Overtaking Maneuvers

Lejun Jiang
University of Pennsylvania
lejunj@seas.upenn.edu

Wesley Yee
University of Pennsylvania
wesyee@seas.upenn.edu

Advisor: Rahul Mangharam
University of Pennsylvania
rahulm@seas.upenn.edu

Abstract

In this study, our objective is to develop a scalable, lightweight framework capable of generating time-parameterized, coordinated overtaking maneuvers which obey vehicular dynamic constraints. Our proposed algorithm utilizes a central node responsible for path planning using RRT and for calculating velocity-tuned paths using a decoupled configuration space for collision avoidance. After processing the RRT* generated path using minimum jerk trajectory smoothing, the paths are sent to each vehicle and executed using the Pure Pursuit algorithm. Lastly, we demonstrate the coordinated overtaking maneuver with three vehicles using the F1/Tenth Gym simulator.*

1. Introduction

Cooperative driving automation (CDA) technologies enable mobility applications which are not achievable via automated driving (ADS)-automated vehicles operating independently of each other. These technologies do so by sharing information that can be used to increase safety, efficiency, and reliability of the transportation system, which may serve to accelerate the deployment of driving automation in on-road motor vehicles. The SAE J3216 CDA standard provides various classes of cooperation, ranging from Class A (Status-sharing) to Class D (Prescriptive), the latter of which enables a central authority to dictate the actions of all on-road agents. In a future world where Class D cooperation is enabled, one could imagine emergency scenarios in which first-response vehicles require all pedestrian vehicles to quickly merge to the road shoulder in a safe and efficient manner.

1.1. Path Planning for Overtaking

In this study, we specifically focus on overtaking, a maneuver which is simple enough to test our coordination algorithm and is a fundamental movement in actual traffic scenarios. Various approaches to path planning for vehicle overtaking have already been investigated in [5]. Such approaches include the use of parameterized sigmoid functions [3], state lattices [8], and planning using Voronoi diagrams [7]. However, many of these methods, while computationally efficient, present several key disadvantages, such as difficulties in dealing with evasive maneuvers during potential collision scenarios, discontinuous edges, and general inflexibility towards complex driving maneuvers. Thus, we adopt the use of the RRT* path planning algorithm due to its efficient computation time and applicability to evasive and increasingly complicated maneuvers.

1.2. Coordinated Path Planning

Coordinated path planning algorithms for mobile robotics systems in general have already been explored in the literature. For holonomic systems, [10] presents a complete algorithm for computing optimal goal assignment, paths, and time parameterizations along those paths for collision avoidance. As the robots modeled in [10] are holonomic, the search space for feasible paths is not constrained to those that are dynamically feasible for those of our vehicle model (defined in Methods).

One strategy for coordinated planning involves searching the joint configuration space constructed with all of the vehicles. However, it quickly becomes apparent that with each additional vehicle, the search space increases exponentially in size. While the resulting solution is guaranteed to be optimal, it is infeasible in practice with a large number of vehicles. Thus, we utilize the decoupled strategy as outlined in [6] and subsequently in [10], in which each vehicle is assigned a priority, and time parameterizations along each vehicle's planned path are calculated in order of priority.

2. Objective

We aim to combine the aforementioned research into a scalable, lightweight algorithm which is able to generate time-parameterized, coordinated overtaking maneuvers which obey vehicular dynamic constraints.

3. Methods

Figure 1 displays the structure of the framework visually, which functions as follows:

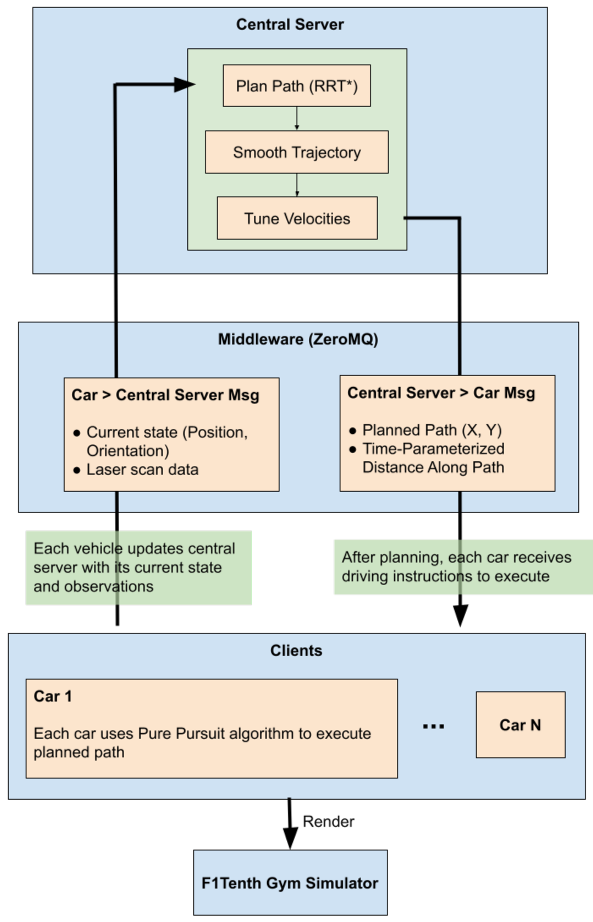

Central Server
Plan Path (RRT*)
Smooth Trajectory
Tune Velocities
Middleware (ZeroMQ)
Car > Central Server Msg
● Current state (Position, Orientation)
● Laser scan data
Central Server > Car Msg
● Planned Path (X, Y)
● Time-Parameterized Distance Along Path
Each vehicle updates central server with its current state and observations
After planning, each car receives driving instructions to execute
Clients
Car 1
Each car uses Pure Pursuit algorithm to execute planned path
...
Car N
Render
F1Tenth Gym Simulator

Figure 1: Diagram of multi-vehicle coordinated path planning framework.

1. Each vehicle $\kappa_i \in K = \{\kappa_1 \ldots \kappa_N\}$ initiates a request to the central server containing its initialized state (global position and orientation) along with its laser scan data.
2. Upon receiving all vehicles' initial states, the central server selects a goal point for each vehicle a distance l_i meters from current position of vehicle κ_i along the path of the track, respectively (this can be changed based on scenarios).
3. The central server utilizes laser scan data from each vehicle to construct a shared occupancy grid for use in path planning.
4. The central server utilizes the RRT* path planning algorithm to calculate the paths $\rho_i \in P = \{\rho_1, \ldots, \rho_N\}$ for each vehicle, respectively.
5. The central server generates smoothed trajectories $x_i \in X = \{x_1, \ldots, x_N\}$ using minimum jerk piecewise polynomials constrained by waypoints in $\rho_1 \ldots \rho_N$.
6. Based on user-determined priorities for each vehicle, the smoothed trajectories $x_1 \ldots x_N$ are tuned and time-parameterized as $\gamma_1 \ldots \gamma_N : \gamma \in [0, 1]$ to avoid collisions using integer programming optimization.
7. The smoothed trajectories $x_1 \ldots x_N$ and time-parameterized paths $\gamma_1 \ldots \gamma_N$ are sent to the respective vehicles $\kappa_1 \ldots \kappa_N$ as responses.
8. Each client vehicle κ_i generates steering inputs from x_i, γ_i using the Pure Pursuit controls algorithm [2], and target velocity inputs from the velocity tuning algorithm.
9. The F1/Tenth Gym Simulator executes the commanded inputs.

3.1. RRT* Path Planning

Our framework's implementation of the RRT* algorithm is based on the pseudocode outlined in [4]. However, we incorporate each vehicle's laser scan data to avoid any obstacles such as the walls or potential blockades in the center of the road. The coordinates coinciding with such obstacles are marked with a "1" in Figure 2, which indicates any path coinciding with that coordinate is deemed a collision. From the perspective of vehicle κ_i (in red), the opposing vehicle κ_j (in blue) is also viewed as an obstacle according to κ_i's laser scan data and is initially marked with a "1". However, if the shared occupancy grid also knows the position of κ_j, it can replace the coordinates marked with a "1" and coinciding with κ_j with "2", which indicates free space.

This distinction is important, since the velocity tuning algorithm enables vehicles $\kappa_1 \ldots \kappa_N$ to overlap planned paths $x_1 \ldots x_N$ in free space so long as they occupy the shared space at different points in time.

3.2. Vehicle Dynamic Model

The framework runs on the F1/Tenth Gym Simulator, which utilizes the Kinematic Single-Track (KS) dynamic model based on [1]. The KS dynamic model models the road vehicle similar to a bicycle, in which the front and rear axis are lumped into a single wheel, and does not consider tire slip.

The state variables are:

$$x_1 = s_x, x_2 = s_y, x_3 = \delta, x_4 = v, x_5 = \Psi$$

The input variables are:

$$u_1 = v_\delta, u_2 = a_{\texttt{long}}$$

Where s_x and s_y indicate the x and y positions in global coordinates, δ indicates the steering angle in the front

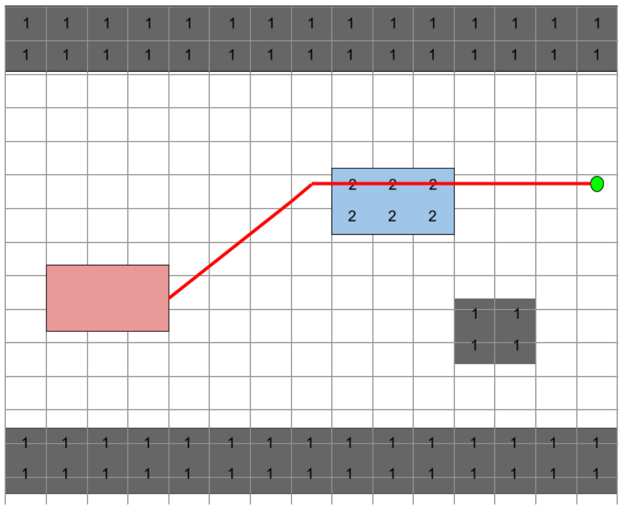


Figure 2: The ego vehicle (in red) updates the shared occupancy grid with its laser scan data. Coordinates of the grid marked with a ”2” are determined as free space for RRT* to plan a path through.

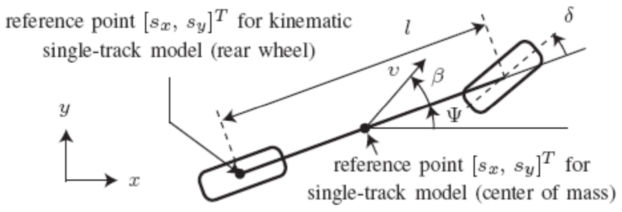

reference point $[s_x, s_y]^T$ for kinematic single-track model (rear wheel)
l
δ
v
β
Ψ
y
x
reference point $[s_x, s_y]^T$ for single-track model (center of mass)

Figure 3: Kinematic Single-Track (KS) vehicle dynamic model from [1].

wheel, v indicates the velocity in the x direction, Ψ indicates the yaw angle, v_δ indicates steering angle velocity of the front wheel, and $a_{\texttt{long}}$ indicates the longitudinal acceleration.

3.3. Trajectory Smoothing

Although paths generated by RRT* are not guaranteed to be dynamically feasible by the KS model, it is helpful to construct smoothed paths through waypoints $\lambda_i^1 \dots \lambda_i^M$ for a given path ρ_i, which eliminate discontinuities that are characteristic of planning in a discretized space. This is done by designing a multi-segment piecewise trajectory $x_i(t)$ such that $x_i(t^k) = \lambda_i^k$ and $x_i(t^{k+1}) = \lambda_i^{k+1}$.

$$t = \begin{bmatrix} t_i^0 & t_i^1 & \dots & t_i^M \end{bmatrix}^T$$

$$x_i(t) = \begin{bmatrix} x_i^0(t) & x_i^1(t) & \dots & x_i^M(t) \end{bmatrix}^T$$

where each polynomial segment is bounded as:

$$x_i(t) = \begin{cases} x_i^1(t), & t_i^0 \le t < t_i^1 \\ x_i^2(t), & t_i^1 \le t < t_i^2 \\ \dots \\ x_i^M(t), & t_i^{m-1} \le t < t_i^m \end{cases}$$

To generate the smoothed trajectory, the following minimum jerk optimization formulation is solved:

$$x_i^{*k}(t) = \arg\min_{x(t)} \int_0^T \mathcal{L}(\dddot{x}, \ddot{x}, \dot{x}, x, t)dt$$
$$\text{where } \mathcal{L} = (\dddot{x}^2)$$

This suggests that the solution for unconstrained motion with minimum cost takes the form of a quintic polynomial that is evaluated starting at $t = 0$:

$$x_i = c_i^{k,5}t^5 + c_i^{k,4}t^4 + c_i^{k,3}t^3 + c_i^{k,2}t^2 + c_i^{k,1}t + c_i^{k,0}$$

It then follows that constraints must be defined in order to solve for each segment's coefficients. The boundary conditions for the first and last segment are first defined as

Position:

$$x_i^0(t_i^0) = \lambda_i^0 \tag{1}$$
$$x_i^M(t_i^M) = \lambda_i^M \tag{2}$$

Velocity:

$$\dot{x}_i^0(t_i^0) = 0 \tag{3}$$
$$\dot{x}_i^0(t_i^0) = 0 \tag{4}$$

Acceleration:

$$\ddot{x}_i^0(t_i^0) = 0 \tag{5}$$
$$\ddot{x}_i^0(t_i^0) = 0 \tag{6}$$

Intermediary positional conditions are defined as:

$$x_i^k(t_i^k) = \lambda_i^k \tag{7}$$
$$x_i^{k+1}(0) = \lambda_i^k \tag{8}$$

Finally, continuity conditions are defined as:

$$\dot{x}_i^k(t_i^k) = \dot{x}_i^{k+1}(0) \tag{9}$$
$$\ddot{x}_i^k(t_i^k) = \ddot{x}_i^{k+1}(0) \tag{10}$$
$$\dddot{x}_i^k(t_i^k) = \dddot{x}_i^{k+1}(0) \tag{11}$$
$$\ddddot{x}_i^k(t_i^k) = \ddddot{x}_i^{k+1}(0) \tag{12}$$

Thus, for a path with M segments, there are a total of $6M$ unknown coefficients. Using the aforementioned constraints and corresponding derivatives of the quintic polynomial, the unknown coefficients are easily solved for. The positions along the smoothed trajectory are thus given by evaluating the corresponding polynomial at the time t_i^k.

3.4. Velocity Tuning

After obtaining $x_1 \ldots x_N$, the constant speed smoothed trajectories for all vehicles, the velocities are tuned based on user defined priority, where κ_k is of a higher priority than κ_{k+1}. The velocities are then tuned as described in Algorithm 1.

Car 1, κ_1, does not require velocity tuning as it is given highest priority. However, lower priority vehicles κ_k must proceed to plan in the time-parameterized space γ_k, which is defined as percent along the smoothed path x_k assuming constant velocity, as seen in Figure 5. If a higher priority vehicle is within $0.5m$ from current vehicle κ_k, the percentage along its trajectory at the instant in time is noted as being in collision with a higher priority vehicle and marked with a blue dot.

The points in this space in collision with other vehicles are to be avoided, so the vehicle must yield to higher priority vehicles in accordance with the optimization formulation described in the section titled "Time-Parameterized Velocity Optimization".

Algorithm 1: Velocity Tuning Algorithm

```
for $x_i \in X$ do
    Initialize collision points, $\Omega_i \leftarrow \varnothing$;
    for $t_j \in max(t_j^M)/\Delta t$ do
        Compute $\Theta_j$, set of positions of higher
          priority vehicles at time $t_j$;
        for $t_k \in max(t_k^M)/\Delta t$ do
            Compute $\theta_k$, position of current vehicle
              at time $t_k$;
            if $|\theta_k - \theta_j| < L\, for\, any\, \theta_j \in \Theta_j$ then
                Compute $s_k \leftarrow \gamma_i(t_k)$, time
                  parameterized value along path $x_i$;
                $\Omega_i$.insert$((\mathtt{t_j}, \mathtt{s_k}))$;
            end
        end
    end
    Optimize velocity path using $\Omega_i$
end
```

3.5. Time-Parameterized Velocity Optimization

In order to avoid any collision while obeying the dynamical constraints, an algorithm has to be designed to solve for the desired input of each car. Our simulation only takes velocity and steering as input to the car, so the target velocity at each time moment has to be determined. In our optimization algorithm, we minimize the time it takes for each car to finish their trajectory, while incorporating the collision avoidance and non-holonomic dynamics into the constraints.

We model the non-holonomic dynamics of the car with Eq. 13 according to [9]. From the equation we can see that the underlying longitudinal dynamics is essentially a P-controller.

$$v_{k+1} = \begin{cases} \frac{2a_{max}}{v_{max}}(v_{des} - v_k)\Delta t + v_k \; if \; v_{des} \geq v_k \\ \frac{2a_{max}}{-v_{min}}(v_{des} - v_k)\Delta t + v_k \; if \; v_{des} < v_k \end{cases} \tag{13}$$

where $a_{max} = 9.51m/s^2$, $v_{max} = 20.0m/s$, $v_{min} = -5.0m/s$, $\Delta t = 0.01s$, and k represents each discrete time step. We can observe that it is essentially a hybrid system, we incorporate it by using a mixed integer linear programming optimization algorithm (we discretize time into n time steps with time interval $\Delta\tau$):

- Variable:
  - $v_{des}[i]$, the desired velocity at time moment i
  - $v[i]$, actual velocity at time moment i
  - $s[i]$, percentage along path at time moment i
  - $y[i] \in \{0, 1\}$, used for incorporating collision avoidance
  - $\delta[i] \in \{0, 1\}$, used for incorporating hybrid system
  - $z[i] = \delta[i] * (v_{des}[i] - v[i])$
- Objective: maximize $\sum_{i=1}^{n} s[i]$
- Subject to:
  - Initial condition: $s[0] == 0$, $v[0] == 0$
  - Terminal constraint: $s[i] \leq 1$
  - Velocity constraint: $v[i] \leq v_{max}$
  - Kinematic constraint: $s[i+1] == s[i] + v[i]\Delta\tau$
  - Dynamical constraint (hybrid system): $v[i+1] == v[i] + 2a_{max}\Delta\tau(\frac{1}{-v_{min}}(v_{des}[i] - v[i]) - (\frac{1}{-v_{min}} - \frac{1}{v_{max}})z[i])$
  - Collision avoidance (using either-or constraints):
    * $s[i] - min(\Omega[i]) \leq My[i]$
    * $max(\Omega[i]) - s[i] \leq M(1 - y[i])$, M is large enough ($M = 100$)

    $\Omega[i]$ comes from Algorithm 1, we assume it is connected, and in practice it is true as seem from the example in Figure 5. Even if it is not, our method can be generalized to disconnected sets by adding more either-or constraints.
  - Enforcing $\delta[i] == 1 \Leftrightarrow v_{des}[i] - v[i] < 0$:
    * $-m'\delta[i] \leq v_{des}[i] - v[i] - m'$, m' is small enough ($m' = -v_{max}$)

    - $-(M' + 0.0001)\delta[i] \leq -(v_{des}[i] - v[i]) - 0.0001$, M' is large enough ($M' = v_{max}$)
  - Enforcing $z[i] == \delta[i](v_{des}[i] - v[i])$:
    - $z[i] \leq M'\delta[i]$
    - $z[i] \geq m'\delta[i]$
    - $z[i] \leq v_{des}[i] - v[i] - m'(1 - \delta[i])$
    - $z[i] \geq v_{des}[i] - v[i] - M'(1 - \delta[i])$

The resulting desired velocity at each time moment is fed into each vehicle for their on-board execution, whereas steering is determined by the desired waypoints and Pure Pursuit algorithm as described before.

4. Results

Simulations were conducted using the F1/Tenth Gym Simulator running on a Linux virtual machine. The following input parameters were used:

Car No.	Velocity (m/s)	Start Pos.	Orientation
1	5.0	(-5.0, 0.0)	0 deg
2	4.0	(-3.5, 0.0)	0 deg
3	3.0	(-2.0, 0.0)	0 deg

In order to enable overtaking, the velocity of higher priority vehicles was set to be highest and decreased in order of priority. Figure 4 displays the results of the simulation at various instances in time. The smoothed trajectories for each car x_1, x_2, x_3 are displayed as purple curves. The current lookahead point used for the Pure Pursuit algorithm is indicated by a green square, while the goal point for each vehicle is indicated by a red square.

At time $t = 0$, all cars begin at rest, with Car 1 starting its acceleration. At time $t = 1.0sec$, Car 1 has almost fully accelerated to its target velocity of $5.0m/s$, with Car 2 starting its acceleration. At time $t = 2.0sec$, Car 2 has reached its target velocity of $4.0m/s$, with Car 3 nearing its target velocity of $3.0m/s$. At time $t = 3.0sec$, all cars have reached their target velocities and are proceeding along their respective planned paths.

As seen in Figure 5a, by planning in the time-parameterized space, Car 2 is able to observe collision points with Car 1 as a section of blue dots and yields for about 1.0 seconds before proceeding. A similar story is observed in Figure 5b, where Car 3 observes collision points with both Car 1 and 2, and yields for about 1.25 seconds before accelerating.

Over 10 runs, the coordinated planning portion of the framework required on average 15.77 seconds to execute, with a standard deviation of 1.61 seconds.

5. Conclusion

In this study, a scalable, lightweight framework capable of generating time-parameterized, coordinated overtaking maneuvers which obey vehicular dynamic constraints was constructed and analyzed. More work is required in several key areas to effectively deploy this system to real-world scenarios:

- The time required to execute the coordinated planning portion of the framework is quite high for a small system of three vehicles, and additional effort will be made to reduce the execution time.
- Currently, the state of the vehicles is known as feedback from the F1/Tenth Simulator, so localization techniques such as EKF and SLAM will need to be incorporated.
- Currently, we only consider emergency scenarios where priorities of vehicles can be easily determined. In other scenarios where priorities can't be easily manually assigned, an algorithm such as the Hungarian Algorithm could be added to select priority based on a heuristic (i.e. vehicle with shortest path length).
- Additional testing will be required on physical F1/Tenth vehicles, as distributed communication and hardware constraints demands increased robustness of the software.

6. Acknowledgements

We would like to thank Professor Rahul Mangharam and members of the mLab including Johannes Betz and Hongrui Zheng for their expert advice on this project.

References

[1] Matthias Althoff, Markus Koschi, and Stefanie Manzinger. Commonroad: Composable benchmarks for motion planning on roads. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 719–726, 2017. 2, 3

[2] R. C. Coulter. Implementation of the pure pursuit path tracking algorithm. 1992. 2

[3] Xujiang Huang, Wenzhe Zhang, and Pu Li. A path planning method for vehicle overtaking maneuver using sigmoid functions. *IFAC-PapersOnLine*, 52(8):422–427, 2019. 1

[4] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *CoRR*, abs/1105.1186, 2011. 2

[5] Christos Katrakazas, Mohammed Quddus, Wen-Hua Chen, and Lipika Deka. Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions. *Transportation Research Part C: Emerging Technologies*, 60:416–442, 2015. 1

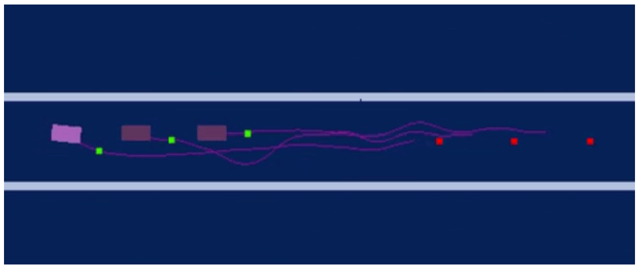
[6] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, USA, 2006. 1

[7] Unghui Lee, Sangyol Yoon, HyunChul Shim, Pascal Vasseur, and Cédric Demonceaux. Local path planning in a complex environment for self-driving car. *The 4th Annual IEEE International Conference on Cyber Technology in Automation, Control and Intelligent*, Hong Kong, June 4-7, 2014. 1

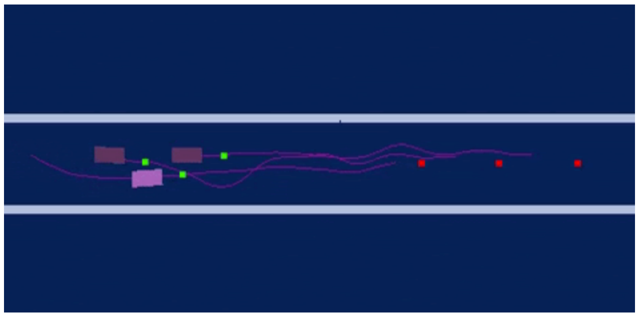
[8] Matthew McNaughton, Chris Urmson, John M. Dolan, and Jin-Woo Lee. Motion planning for autonomous driving with a conformal spatiotemporal lattice. *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13, 2011. 1

[9] Matthew O'Kelly, Hongrui Zheng, Dhruv Karthik, and Rahul Mangharam. F1tenth: An open-source evaluation environment for continuous control and reinforcement learning. In *NeurIPS 2019 Competition and Demonstration Track*, pages 77–89. PMLR, 2020. 4

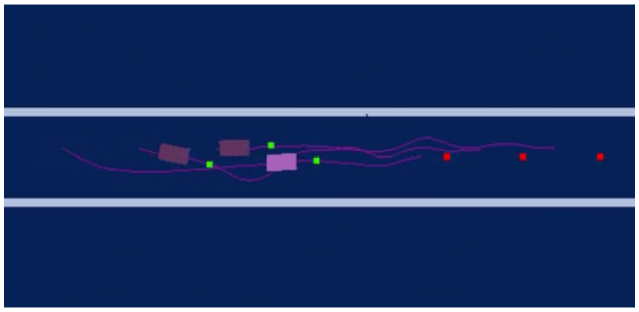
[10] Matthew Turpin, Nathan Michael, and Vijay Kumar. Concurrent assignment and planning of trajectories for large teams of interchangeable robots. *IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, May 6-10, 2013. 1



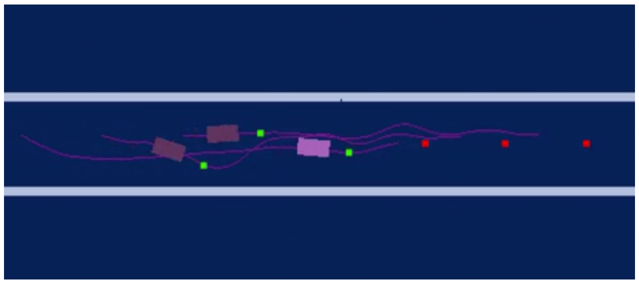
(a) $t = 0sec$



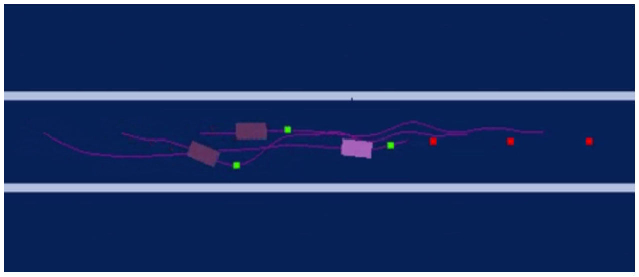
(b) $t = 1.0sec$



(c) $t = 2.0sec$

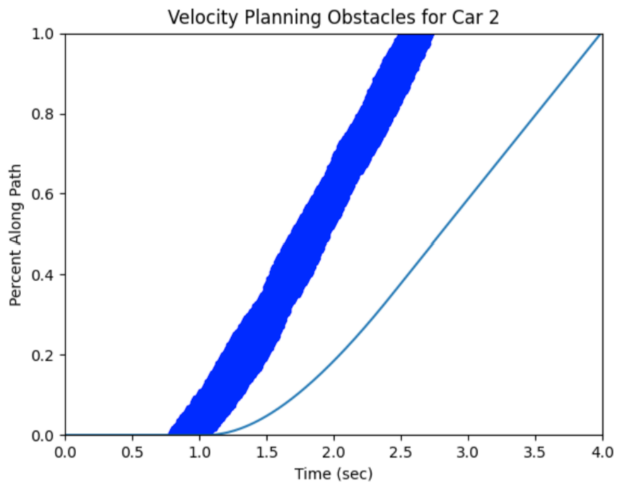


(d) $t = 3.0sec$

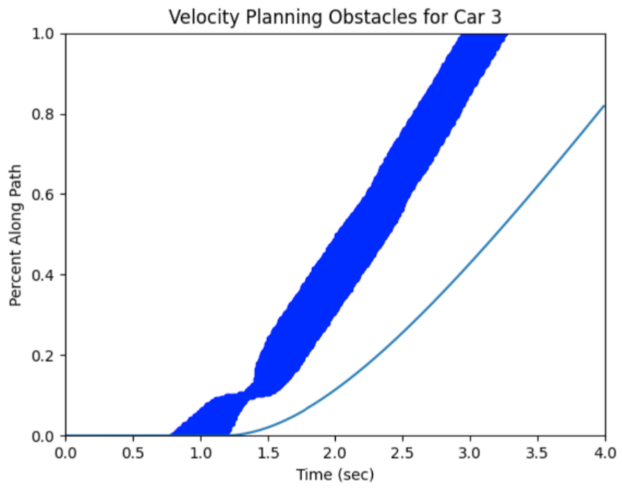


(e) $t = 4.0sec$

Figure 4: Three vehicle overtaking maneuver



(a) Car 1 is displayed in blue as an obstacle which Car 2 must yield to



(b) Car 2 and 3 are displayed in blue as obstacles which Car 3 must yield to

Figure 5: Observed Obstacles in Time-Parameterized Space for Cars 2 and 3