Cooperative Driving of Connected Autonomous Vehicles using Responsibility Sensitive Safety Rules: A Control Barrier Functions Approach

MOHAMMAD KHAYATIAN, VECNA ROBOTICS
MOHAMMADREZA MEHRABIAN, SD SCHOOL OF MINES AND TECHNOLOGY
I-CHING TSENG, CHUNG-WEI LIN, NATIONAL TAIWAN UNIVERSITY
CALIN BELTA, UNIVERSITY OF MARYLAND, COLLEGE PARK
AVIRAL SHRIVASTAVA, ARIZONA STATE UNIVERSITY

Connected Autonomous Vehicles (CAVs) are expected to enable reliable, efficient, and intelligent transportation systems. Most motion planning algorithms for multi-agent systems implicitly assume that all vehicles/agents will execute the expected plan with a small error and evaluate their safety constraints based on this fact. This assumption, however, is hard to keep for CAVs since they may have to change their plan (e.g., to yield to another vehicle) or are forced to stop (e.g., A CAV may break down). While it is desired that a CAV never gets involved in an accident, it may be hit by other vehicles and sometimes, preventing the accident is impossible (e.g., getting hit from behind while waiting behind the red light). Responsibility-Sensitive Safety (RSS) is a set of safety rules that defines the objective of CAV to blame, instead of safety. Thus, instead of developing a CAV algorithm that will avoid any accident, it ensures that the ego vehicle will not be blamed for any accident it is a part of. Original RSS rules, however, are hard to evaluate for merge, intersection, and unstructured road scenarios, plus RSS rules do not prevent deadlock situations among vehicles. In this paper, we propose a new formulation for RSS rules that can be applied to any driving scenario. We integrate the proposed RSS rules with the CAV's motion planning algorithm to enable cooperative driving of CAVs. We use Control Barrier Functions to enforce safety constraints and compute the energy optimal trajectory for the ego CAV. Finally, to ensure liveness, our approach detects and resolves deadlocks in a decentralized manner. We have conducted different experiments to verify that the ego CAV does not cause an accident no matter when other CAVs slow down or stop. We also showcase our deadlock detection and resolution mechanism using our simulator. Finally, we compare the average velocity and fuel consumption of vehicles when they drive autonomously with the case that they are autonomous and connected.

CCS Concepts: • **Computer systems organization → Robotic autonomy**; **Robotics**.

ACM Reference Format:
Mohammad Khayatian, Vecna robotics, Mohammadreza Mehrabian, SD School of Mines and Technology, I-Ching Tseng, Chung-Wei Lin, National Taiwan University, Calin Belta, University of Maryland, College Park, Aviral Shrivastava, Arizona State University . 2024. Cooperative Driving of Connected Autonomous Vehicles using Responsibility Sensitive Safety Rules: A Control Barrier Functions Approach. 1, 1 (January 2024), 26 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

---

Author's address: Mohammad Khayatian, Vecna robotics
Mohammadreza Mehrabian, SD School of Mines and Technology
I-Ching Tseng, Chung-Wei Lin, National Taiwan University
Calin Belta, University of Maryland, College Park
Aviral Shrivastava, Arizona State University.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2024 Association for Computing Machinery.
XXXX-XXXX/2024/1-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Autonomous Vehicles (AVs) have the potential to make transportation safer by reducing the number of accidents that are caused due to human error. When AVs become connected (which are referred to as Connected Autonomous Vehicles (CAVs)), they can further improve road safety by sharing their information with each other such as position, velocity, future plans, etc. In addition, CAVs are projected to improve fuel consumption, travel time, and passenger comfort through cooperative driving.

Achieving cooperative behaviors among robots is widely studied under multi-agent motion planning in the robotics domain [1],[2]. Similarly, in the Intelligent Transportation Systems (ITS) domain, many techniques [3] are proposed where CAVs share their information with each other (through V2V) or the infrastructure (through V2I) to perform traffic management at intersections [4],[5],[6],[7], or merge points [8],[9],[10].

In general, existing motion planning algorithms and traffic management techniques consider a safety buffer around each vehicle to account for uncertainties in the localization and trajectory tracking. A trajectory is deemed to be safe if the safety buffer of a vehicle does not overlap with obstacles or other vehicles' safety buffer at any point in time. While reasonable, this definition may not provide absolute safety because it implicitly assumes that all vehicles will follow the expected plan –with small errors that are within the safety buffer. However, vehicles are operating in a very dynamic environment and may have to change their plan frequently. For example, consider a scenario when two vehicles are driving on a street, one behind the other. If the front vehicle suddenly stops for any unplanned reason (e.g. yielding to a jaywalker), then the rear vehicle may hit the front car.

Responsibility-Sensitive Safety (RSS) approach [11] from Mobileye+Intel addresses the safety issue from the legal/blame perspective and allows vehicles that have the right-of-the-way according to the rules of the road to change their plan. RSS proposes a set of safety rules such that if a vehicle abides by these rules, then it cannot be blamed for an accident. In the scenario mentioned above, RSS rules are used to determine the minimum distance at which the rear vehicle should follow the front one so that it will be able to stop without causing an accident even in the worst-case scenario. RSS uses a lane-based coordinate system to define lateral and longitudinal distances between vehicles depending on the driving scenario. The main shortcoming of RSS is that not all scenarios can be evaluated because longitudinal and lateral distances are not clearly defined for merges, intersections, as well as unstructured roads – where lane markings are not provided. The first contribution of this paper is to provide a trajectory-based definition for RSS rules, that works in all situations, including merges, intersections, and unstructured roads. Instead of a lane-based coordinate system, we use future trajectories of CAVs to represent vehicles' conflicts, which can be applied to any road geometries and situations. Inspired by the RSS legal/blame perspective, we develop a new set of safety rules for CAVs to guarantee that no accidents happen if CAVs abide by proposed RSS rules.

When CAVs interact with each other, they may face a deadlock situation where they yield to each other for an indefinite time. Researchers have proposed methods to detect and resolve deadlocks at intersections [6, 12, 13] and roundabouts [14]. In existing approaches, the intersection/roundabout area is divided into a grid of zones, and vehicles that intend to occupy the same zone are said to have a conflict. Then, the dependencies between CAVs (who should enter a conflict zone first and who enters second) are represented with a directed graph, and deadlocks are resolved by removing cycles in the graph. One of the limitations of existing approaches is that they use a fixed grid of zones to detect conflicts between vehicles and the size of each zone affects the efficiency of the conflict detection algorithm –since using coarse grids makes the schedule pessimistic and using fine grids increases the number of checks. Furthermore, in existing approaches, the dependency graph is computed individually by each CAV, which is extremely inefficient because the same computation is done redundantly and the overhead grows as the number of vehicles increases. As the second contribution of this paper, we propose an efficient and decentralized approach to detect and resolve deadlocks where each CAV determines only its own conflicts and shares that information with others. Since minimal information is shared to achieve consensus among CAVs, the network overhead is very low.

In our previous paper [15], we studied the motion planning and control of the ego CAV using a heuristic approach. As the third contribution of the paper, we use Control Barrier Functions (CBF) together with the Control Lyapunov Function (CLF) to achieve the optimal energy behavior while enforcing the safety constraints, speed limits, and limits of the control input. For the first time, we use Control Barrier Functions (CBF) to enforce RSS safety constraints for CAVs.

In summary, this article makes three contributions:

- We propose a new set of RSS safety rules that can be applied to any scenario and easily be evaluated

- We propose an efficient deadlock detection and resolution algorithm for CAVs

- We use CBF and CLF to compute the safe and optimal control input for each CAV

Results from conducting experiments on our realistic simulator –that considers vehicle dynamics and network delay– demonstrate that all CAVs remain safe even if one or more CAVs slow down or stop at any point in time. We evaluate the efficiency of our approach by comparing the average travel time of CAVs with a case that vehicles are autonomous but not connected. Finally, we showcase our deadlock resolution mechanism for an intersection scenario.

2 RELATED WORK

In the ITS domain, many researchers have proposed methods to cooperatively manage CAVs at intersections [3–7, 16–19], roundabouts [20], ramp-merging [8–10], performing cooperative lane changing [21, 22], forming platooning in highways [23, 24]. Such approaches can only be applied to a specific scenario and do not scale. There have been a number of cooperative approaches that are not scenario-based. In the method proposed by During *et al.* [25],[26], the ego CAV first determines a set of possible maneuvers that can resolve the conflict and then, selects the one that has the lowest cost. The cost is determined based on energy consumption, time of maneuver, and driving comfort. In another work, Chen *et al.* [27] proposed a cooperative driving algorithm where the driving information of neighboring CAVs is obtained and the desired velocity is predicted using a Recurrent Neural Network (RNN). The motion planner utilizes the predicted velocity and incorporates a fuzzy path-following controller. These approaches, however, do not consider cases where a CAV is unable to perform the desired maneuver/follow the assigned trajectory.

In the robotics domain, many researchers have focused on multi-agent motion planning algorithms problem [1],[2]. In general, cooperative motion planning algorithms can be categorized as distributed [28] and centralized [29]. In distributed approaches, each agent computes a path such that it avoids obstacles and other agents while in centralized approaches, a central planner (could be on each agent) computes the plan for all agents by exploring the whole design space. In general, distributed approaches are more popular as they require less computation and are more resilient to changes in the plan or uncertainty. Existing motion planning algorithms for multi-agent systems and traffic management approaches for CAVs provide safety proofs based on the assumption that all agents stick to their plan or error is small. In the real world, CAVs may have to slow down and stop due to unforeseen reasons e.g. a CAV may break down. As a result, existing techniques are not absolutely safe for CAVs.

In 2017, researchers from Mobileye proposed a set of rules called RSS [11], which determines the minimum distance that an AV must maintain from other vehicles in order to remain safe and not be blamed for an accident. RSS rules consider the worst-case scenario for other vehicles and the ego vehicle (during the response time) to provide safety guarantees. RSS rules have been used to develop a monitoring system [30] and are implemented in the Carla simulator. In [5], researchers have proposed to use surveillance cameras [31] to check for rogue cars at the intersection and provide safety considering the worst-case scenario similar to RSS.

The main issue with RSS is that it uses a lane-based coordinate system and safety rules are defined based on longitudinal (towards the lane) and lateral (perpendicular to the lane) distances, which is hard to evaluate for intersections, merges, or unstructured road scenarios where no road markings are present. In addition, RSS rules do not consider the interaction among other CAVs and therefore, cannot detect cases where a deadlock happens.

CBFs are used in the transportation domain to enforce safety and stability constraints [32, 32] where a Quadratic Programming (QP) problem is formulated based on the constraints on the control input. In another similar work, CBFs are used for Adaptive Cruise Control (ACC) problem [33]. However, in all of these approaches, system delay (response time of the ego vehicle) is ignored. In this paper, we use RSS safety rules that account for the vehicle's response time.

Researchers have proposed algorithms to detect and resolve deadlocks at intersections [6, 12], roundabouts [14] and network of intersections[13]. In such approaches, a set of pre-defined zones is used to represent the occupancy of CAVs. Next, a wait-for graph is created to represent the dependency between vehicles for entering conflict zones, and deadlocks are identified by detecting cycles in the graph. However, using fixed conflict zones to detect a conflict and perform deadlock resolution is either inefficient (for coarse zones) or compute-intensive (for fine zones). Furthermore, existing approaches do not consider vehicle dynamics when resolving a deadlock and assume that a deadlock is resolved in one shot. While in reality, it takes some time for CAVs to slow down/speed up and resolve a deadlock.

3 GENERIC FORMULATION OF RSS RULES

In this section, we introduce a trajectory-based formulation for RSS rules. The advantage of this approach is that the rules are generic and can be applied to all cases, including unstructured roads.

Given the future paths of CAVs are known, each CAV can determine the set of conflict zones C. A conflict zone, $C_i \subset C$ is defined as a contour that includes a subset of two CAVs' future paths (FP) where the distance between the future paths is less than a threshold, d_{th}. Since two CAVs may have more than one conflict, only consecutive edges that have a distance of less than d_{th} are considered to be a part of the same conflict zone. The midpoints of the edges are used to calculate the distance between two edges from two future paths. To specify the boundaries of a conflict zone, midpoints of first and last edges are used.

Based on the road geometry and rules of the road, every pair of CAVs can determine who has the advantage to enter the conflict zone first and who has the disadvantage. For simplicity, we assume the CAV with the earlier arrival time has the advantage. Without loss of generality, we assume that one of the CAVs has an advantage and the other one has a disadvantage. We understand that considering a more sophisticated rule, such as one based on traffic volume on different roads, would indeed impact the complexity of the planning. Despite this, we view the utilization of right-of-way determined by arrival time as a baseline or initial stride towards potential advancements in forthcoming research endeavors. This choice stands out for its inherent fairness and intuitive nature, setting the stage for iterative refinements in the pursuit of optimized solutions.

We represent the distance of the CAV with the advantage from the beginning of the conflict zone and from the end of the conflict zone with d^A_{begin} and d^A_{end}, respectively. Similarly, we represent the distance of the CAV with a disadvantage from the beginning of the conflict zone with d^D_{begin}. Figures 1, 2, and 3 show different scenario where the d^A_{begin}, d^A_{end} and d^D_{begin} are shown. We assume that Equation 1 represents the dynamics of each CAV. We assume the following vehicle dynamics for the CAV.

$$\begin{cases} \dot{x} = v\cos(\phi); \\ \dot{y} = v\sin(\phi); \\ \dot{\phi} = \frac{v}{L}\tan(\psi); \\ \dot{v} = a, \end{cases} \tag{1}$$

where x and y represent the position of the ego CAV in Cartesian coordinates, ϕ is the CAV's heading angle from the x-axis, v and a are linear velocity and acceleration of the CAV respectively, L is CAV's wheelbase distance and ψ is steering angle of front wheels with respect to the heading of the CAV. In order to make the model more realistic, we consider an upper bound and a lower bound on the acceleration and steering angle of a CAV as $a \in [a_{min}, a_{max}]$ and $\psi \in [\psi_{min}, \psi_{max}]$ where a_{max} and a_{min} are the maximum acceleration and deceleration and ψ_{max} and ψ_{min} are the maximum and minimum steering angles of the vehicle.

To simplify the longitudinal distance problem, the trajectory of each CAV is projected onto its path and represented with the double-integrator model.

$$\begin{cases} \dot{x} = v \\ \dot{v} = a \end{cases} \tag{2}$$

As a result, the stop distance of the CAV with advantage is calculated as:

$$d^A_{stop} = \frac{v_A^2}{2|a_{brake}|}, \tag{3}$$

where v_A is the velocity of the CAV with advantage. We assume that the worst-case response time of the CAV is ρ. Taking into account the response time, the worst-case stop distance of the CAV with a disadvantage is calculated as:

$$d^D_{stop} = v_D\rho + \frac{1}{2}a_{ACC}\rho^2 + \frac{(v_D + a_{ACC}\rho)^2}{|2a_{brake}|}, \tag{4}$$

where v_D is the velocity of the CAV with a disadvantage. The first two terms ($v_D\rho$ and $\frac{1}{2}a_{ACC}\rho^2$) indicate that the CAV with disadvantage may be accelerating in the worst-case scenario while waiting for broadcast information from other CAVs (e.g., the CAV with the advantage). If the distance of the CAV with an advantage from the end of the conflict zone is greater or equal to the stop distance of the CAV with an advantage ($d^A_{end} \geq d^A_{stop}$), there is a possibility that it may slow down and stop inside the conflict zone and block the CAV with the disadvantage. Otherwise, there is no conflict. Accordingly, we define the modified RSS rule as:

Definition 3.1. **General RSS Rule:** Given the entering order of CAVs to a conflict zone is known, the minimum safe distance to maintain from the conflict zone (d^D_{SAFE}) for the CAV with a disadvantage is:

$$d^D_{SAFE} = \begin{cases} d^D_{stop} - d^A_{scenario} + \Delta, & \text{if } d^A_{end} > d^A_{stop}; \\ 0, & \text{otherwise,} \end{cases} \tag{5}$$

and

$$d^D_{SAFE} > v_D\rho + \frac{1}{2}a_{ACC}\rho^2, \tag{6}$$

where $d^A_{scenario}$ is the scenario-dependent distance that the CAV with advantage travels inside the conflict zone, $\Delta = \frac{VL_A+VL_D}{2}$ and VL_A and VL_D are the vehicle length for the vehicle with advantage and disadvantage, respectively. Note that the safe distance deduced from Equation 5 might lean towards the conservative side due to our prioritization of safety in this work. Introducing a method with increased precision may result in a greater computational burden, potentially presenting new hurdles for real-time safety mechanisms. Investigating and enhancing a more accurate safe distance, while maintaining manageable computational demands in real-time scenarios, is earmarked for future exploration.

Since the distance values are calculated based on the center of CAVs, the term $\frac{VL_A+VL_D}{2}$ is added. To make sure the traveled distance during the response time of the CAV with disadvantage is not greater than the safe distance, the second Equation 6 should be satisfied too.

Lemma 3.2. *If the CAV with a disadvantage always maintains a distance of at least d^D_{SAFE} from its conflict zone, it will not hit the CAV with an advantage even if it changes its plan and decelerates at any point in time.*

If the distance of the CAV with the advantage from the end of the conflict zone is smaller than its stop distance, $d^A_{end} < d^A_{stop}$, it will stop outside of the conflict zone even if it decelerates at a rate of smaller than or equal to a_{brake}. If the distance of the CAV with the advantage from the end of the conflict zone is greater than its stop distance, $d^A_{end} > d^A_{stop}$, it may stop inside the conflict zone if it decelerates. In this case, the CAV with a disadvantage will be notified after ρ milliseconds in the worst-case scenario. If the CAV with disadvantage accelerates at a rate smaller than or equal to a_{ACC} during this time interval (ρ) and then decelerates at a rate of a_{brake}, its stop distance will be equal to d^D_{stop} (Equation 4) and it will not enter the conflict zone and no accident will happen. For scenarios where the scenario-dependent distance is not zero, $d^A_{scenario} > 0$ (same lane and merge), the paths of the CAVs overlap and if the CAV with advantage decelerates, it will allow the CAV with disadvantage to travel through the conflict zone by $d^A_{scenario}$ and still be safe. As a result, the required safe distance is $d^D_{stop} - d^A_{scenario}$.

Next, we study a few case studies and show how the safe RSS distance is calculated for each scenario.

3.1 Same Lane

Let us consider a scenario where two CAVs are driving in the same lane as depicted in Figure 1. The front CAV has the advantage since its arrival time at the conflict zone is smaller than the rear CAV. Since the paths of the front CAV overlap with the path of the rear CAV, $d^A_{scenario} = d^A_{stop}$, which means the front CAV travels d^A_{stop} meters inside the conflict zone before a complete stop and the rear CAV has d^A_{stop} meters more to stop. According to Equation 5, the required safe distance for the rear CAV (d^D_{SAFE}) to maintain from the conflict zone/front CAV is:

$$d^D_{SAFE} = d^D_{stop} - d^A_{stop} + \Delta,$$

where d^D_{stop} and d^A_{stop} are calculated according to Equations (3) and (4).

3.2 Intersection

Now, let us consider a scenario where two CAVs approach an intersection and their future path crosses inside the intersection area as depicted in Figure 2. We assume the arrival time of the green

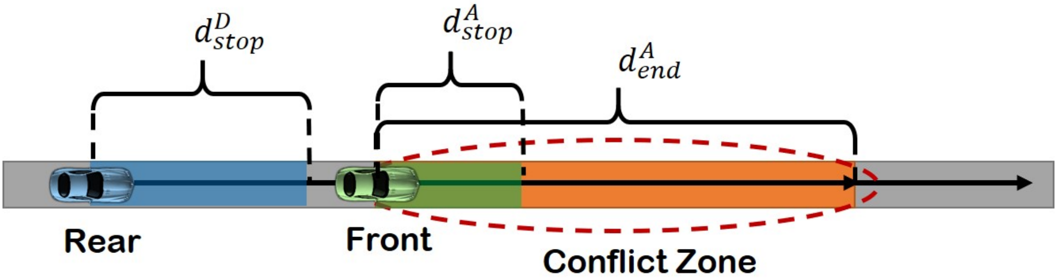


Fig. 1. An example of a same-lane scenario with two CAVs. The front CAV has the advantage and its distance from the conflict zone is zero. The conflict zone is highlighted in orange. Note that this is a simplified graph. The conflict zone can have crossroads nearby.

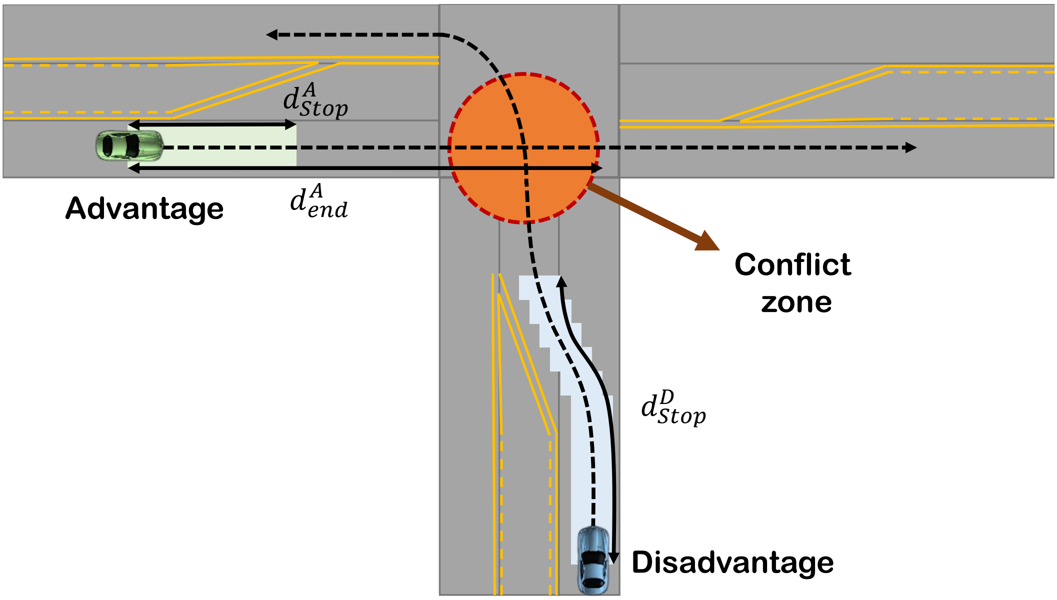


Fig. 2. A scenario with two CAVs approaching an intersection and their future path intersect. It is safe to enter the conflict zone after the other CAV leaves the conflict zone.

CAV to be earlier than the blue CAV and therefore, it has the advantage. If the green CAV stops anywhere inside the conflict zone, it's not safe for the blue CAV to enter the conflict zone. Therefore, the scenario-dependent distance is zero, $d^A_{scenario} = 0$. As a result, we have:

$$d^D_{SAFE} = \begin{cases} d^D_{stop} + \Delta, & \text{if } d^A_{end} \geq d^A_{stop}; \\ 0, & \text{otherwise.} \end{cases}$$

If the distance of the green CAV from the end of the conflict zone is smaller than its stop distance, even in the worst case (if it decelerates at the maximum rate), it will stop outside the conflict zone and will not cause conflict for the blue CAV. In this case, there will be no conflicts and $d^D_{SAFE} = 0$.

3.3 Merge

Next, we consider a merge scenario where two CAVs merge into the same lane as it is shown in Figure 3. Without loss of generality, we assume one of the CAVs (green one) has the advantage and the other CAV has the disadvantage respectively. In this scenario, the scenario-dependent distance is

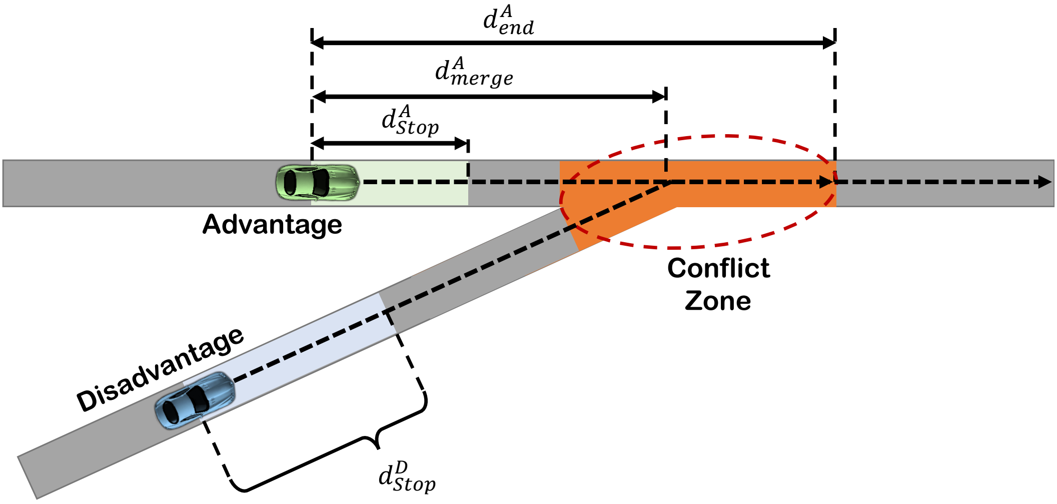


Fig. 3. A scenario where two CAVs are expected to be merged into the same lane. The CAV with an earlier arrival time has the advantage.

$$d^A_{scenario} = \begin{cases} 0, & \text{if } d^A_{stop} < d_{merge}; \\ d^A_{stop} - d^A_{merge}, & \text{if } d^A_{stop} \geq d_{merge}, \end{cases} \tag{7}$$

where d^A_{merge} is the distance of the CAV with advantage from the merging point, which is indicated in Figure 3. As a result, the blue CAV must maintain the following minimum distance from the conflict zone:

$$d^D_{SAFE} = d^D_{stop} - \min(0, d^A_{stop} - d^A_{merge}) + \Delta. \tag{8}$$

Note that once the blue CAV reaches the merge point, the $d^A_{scenario}$ is changed. The lateral case in the original RSS rules (two CAVs driving on adjacent lanes) can be modeled like a merging case. If any of the CAVs attempts to merge into the other CAV's lane, it is only allowed if the created conflict zone is far enough from the other CAV i.e. at least d_{max}.

4 COOPERATIVE DRIVING OF CAVS

In this section, we first present the algorithm that runs on each CAV assuming no deadlock situation happens. In the next section, we explain the deadlock resolution algorithm.

4.1 Graph Map

The graph map is a directed graph G(V, E) containing all the waypoints (V) for navigation and the way they are connected (E). The waypoints are placed at the center of the lane for normal roads and explicitly defined for merge and intersection areas. Figure 4 shows the map for three cases. Each edge has a weight w that indicates the travel time of that segment of the road. The weight of an edge is computed as $w = \frac{d}{v_{max}}$ where d is the distance between the source and sink node in

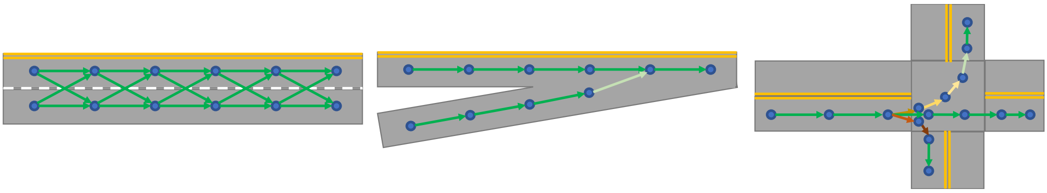


Fig. 4. Map graph for three road cases. The color of each edge corresponds to the weight of the edge (maximum velocity that is allowed).

2D and v_{max} is the maximum velocity that is allowed for that segment of the road. The edges are colored in Figure 4 to reflect the weight value. For instance, the maximum velocity is reduced for making a turn at the intersection.

4.2 Main Algorithm

Given the initial position and final destination of a CAV are known, the motionPlanner uses the world's map to determine the shortest route (R) that connects CAV's current position to the destination. We assume at least one feasible path exists that connects CAV's current location to its destination. The map, $M(N, E)$, is a directed graph where N is the set of nodes (waypoints) and E is the set of edges (connections between waypoints). Each edge has a weight, w, which indicates the minimum travel time for that segment of the road. In our algorithm, we assume the ego CAV's computation time and communication time are bounded by T.

In a periodic manner, each CAV broadcasts its ID, position, velocity, timestamp, and future path (FP), which is an array of x-y coordinates. We assume that all CAVs synchronize their clock using GPS so that timestamps are captured with clocks that have almost the same notion of time. When the CAV receives the information of other CAVs, it checks if their paths intersect or if the distance between their paths is less than a threshold. If so, the CAV computes a set of conflict zones (C). For each conflict zone, the CAV determines which vehicle has the advantage to enter the conflict zone

Algorithm 1 CAVs algorithm

```
while not reached the destination do
    FP = future_path()
    CAV_info = [x, y, v, ts, FP, ID]
    broadcast(CAV_info)
    others_info = listen_for_Info()
    for members of others_info do
        [C, PDG] = find_conflict(CAV_info, others_info)
    end for
    broadcast(PDG)
    others_PDG = listen_for_PDGs()
    CDG = construct_CDG(PDG, others_PDG)
    C = deadlock_resolution(C, CDG)
    if has a disadvantage then
        Map = update_weights(Map)
        [FP, velocity] = motion_planner($C$, Map)
    end if
    motionController(FP, Velocity)
end while
```

first based on who is expected to reach the conflict zone first. To detect possible deadlocks, the CAV computes a graph called Partial Dependency Graph (PDG), which represents the dependency among other CAVs and itself (who should yield to who over a conflict zone). Next, the CAV broadcasts the computed PDG, and after receiving other CAVs PDG, it constructs the Complete Dependency Graph (CDG) to detect and resolve possible deadlocks. Finally, if the CAV has a disadvantage over a conflict zone, it computes a safe velocity so that it always maintains a safe distance from that conflict zone. Based on the determined velocity, the weight of some of the edges is updated to reflect the presence of other CAVs and to make sure a safe distance is always maintained from the conflict zone. Then, the motion planner runs the shortest path algorithm again to check if a shorter path exists that does not cause a new conflict. Finally, the motion controller uses a subset of future waypoints and velocities of corresponding edges to determine the desired velocity and control inputs (steering angle and acceleration) for the CAV. Alg. 1 shows the pseudo-code of our algorithm that is executed on each CAV. To have a better understanding of our algorithm, we have depicted different components of our approach and their relationship in Figure 5. Next, we will focus on explaining the functionality of each component of the algorithm.

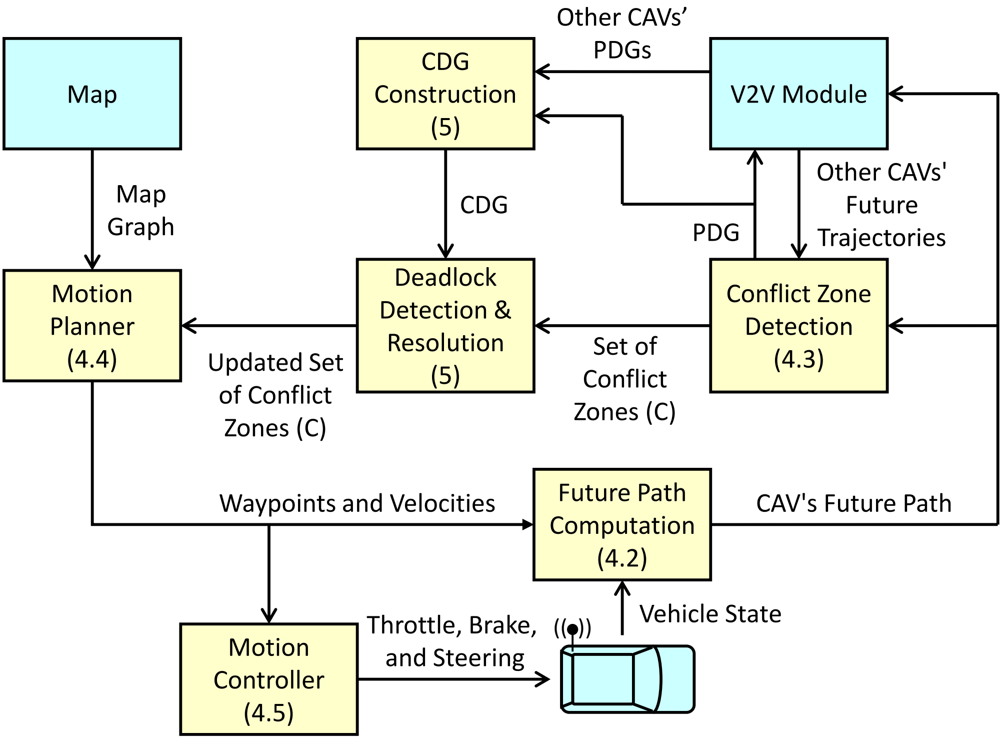

Map
Map Graph
Motion Planner (4.4)
CDG Construction (5)
Other CAVs' PDGs
V2V Module
CDG
PDG
Other CAVs' Future Trajectories
Deadlock Detection & Resolution (5)
Conflict Zone Detection (4.3)
Updated Set of Conflict Zones (C)
Set of Conflict Zones (C)
Waypoints and Velocities
Future Path Computation (4.2)
CAV's Future Path
Vehicle State
Motion Controller (4.5)
Throttle, Brake, and Steering

Fig. 5. Overview of our approach. Details of each component –except the V2X module and map– are explained later.

4.3 Future Path Computation

Each CAV broadcasts its ID, position (x, y), velocity (v), and the corresponding timestamp (ts) as well as its future path $((x_1, y_1), ..., (x_n, y_n))$. Assuming the CAV's motion controller is tuned to have a short settling time, the CAV will track its path with a negligible error. As a result, we represent the future position of the CAV with a subset of its expected route (R). Given $R \subset M(N, E)$ is the route of the CAV, the future path of the CAV, $FP \subset R$ is calculated as follows which consists of n points:

$$FP = \left\{ (x_i, y_i) \in R \,\middle|\, \left(\sum_{i=2}^{i=n} \sqrt{\Delta X_i^2 + \Delta Y_i^2} \right) < d_{max} \right\}, \tag{9}$$

where $\Delta X_i = x_i - x_{i-1}$ and $\Delta Y_i = y_i - y_{i-1}$ and d_{max} is the fixed length of the future path calculated as:

$$d_{max} = v_{max}(\rho + t_b). \tag{10}$$

ρ represents the worst-case end-to-end delay from one CAV capturing its information and broadcasting it, to another CAV's actuation based on the received information (see Figure 6) and t_b is the worst-case brake time which can be calculated as $t_b = \frac{v_{max}}{|a_{brake}|}$. Figure 6 shows the execution profile of our algorithm on two CAVs (i and j). Let us assume that CAVs i and j have a conflict and CAV i (top) has the advantage. If CAV i slows down due to any reason right after sensing and broadcasting its info, the CAV j will not be notified until receiving the next broadcast. As a result, the worst-case end-to-end delay (ρ) is bounded by $2T$ as depicted in Figure 6. By computing the d_{max} based on the worst-case info-sharing delay and worst-case braking time, we ensure that for the first time that two CAVs detect that they have a conflict, the CAV with the disadvantage have enough distance to safely stop without entering the conflict zone, even in the worst-case scenario.

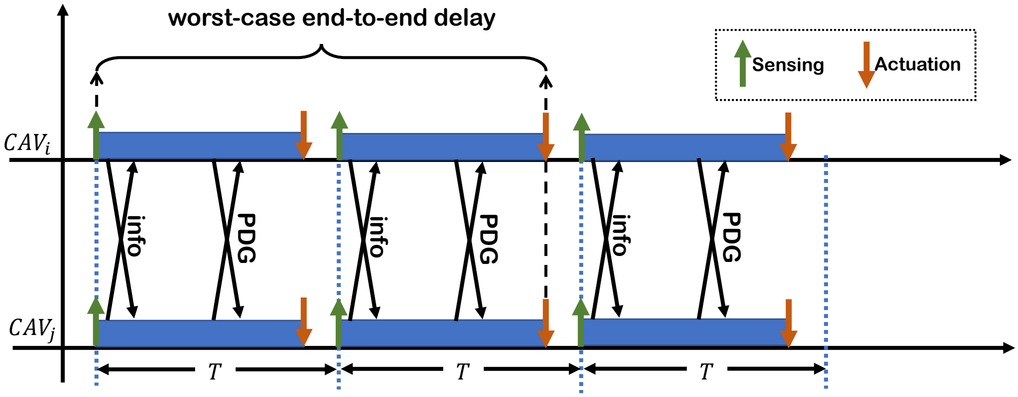


Fig. 6. CAVs perform computation and communication in a synchronized manner. The worst-case sensing to actuation delay corresponds to the case that CAV_i breaks down right after sensing.

4.4 Conflict Zone Detection and Advantage Determination

Despite existing approaches that use fixed conflict zones, we use CAV's expected trajectory to detect a conflict zone. As mentioned before, CAVs' future paths (FP) are used to represent their expected future position. First, CAVs compute the distance between the midpoint of edges on their path. All contiguous edges that have a distance less than d_{th} are considered to be a part of the same conflict zone. Two CAVs may have multiple conflicts on their path as depicted in Figure 7. Each conflict zone C_i is a data structure that includes waypoints that are inside the conflict zones, a distance of CAVs from the beginning and end of the conflict zone, their expected arrival time at the conflict zone (Equation 11) and the ID of the CAV that has the advantage. We compute the arrival

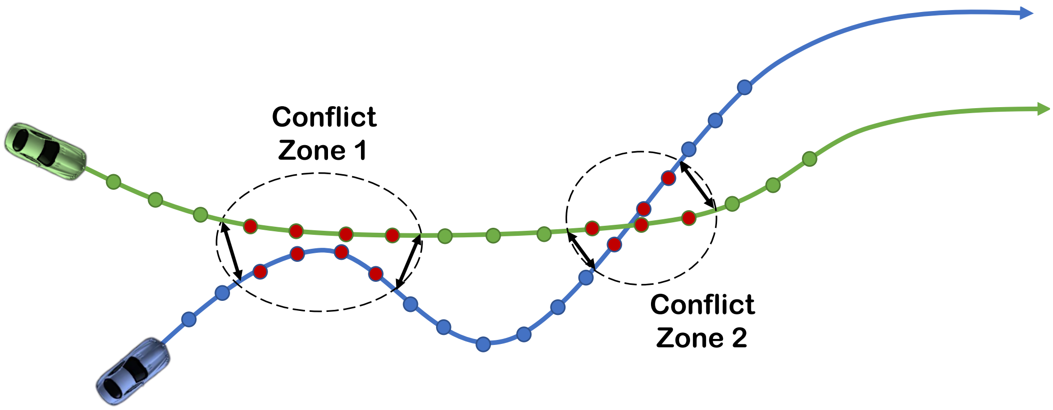


Fig. 7. An example of two CAVs with arbitrary paths and two conflict zones. The conflict zone includes parts of the CAV path (waypoints) where the distance between paths of CAVs is less than a threshold.

time assuming the CAV drives at a constant velocity.

$$TOA_i = \frac{d^i_{begin}}{v_i}, \tag{11}$$

where d^i_{begin} is the distance of the CAV i from the conflict zone and v_i is the velocity of the CAV i. Since the algorithm is executed periodically (every T ms), the value of TOA_i is updated as the velocity of the CAV changes. If a CAV is stopped inside a conflict zone, its arrival time is set to zero. By default, the CAV with the earliest arrival time will have the advantage unless it is changed to resolve a deadlock (explained in the next section) or the other CAV has a priority (e.g. opposite direction). If two CAVs have the same arrival time, the CAV with the lower ID will have the advantage to break the tie. In addition, if the difference between the arrival times of two CAVs is within the accuracy of the clock synchronization (± 10 nanoseconds for GPS), they use CAVs ID to determine who has the advantage.

4.5 Motion Planner and Controller

4.5.1 Map Update. To compute the set of Future Waypoints (FW) for the CAV, the planner updates the weights of the edges of the map graph and then computes the shortest path using the Dijkstra algorithm. E_I is the set of edges that are connected to waypoints that are *inside* a conflict zone (FW_I) and E_D is the set of edges that are connected to waypoints that are on the future path of the ego CAV and are *within* the d_{SAFE} distance of the conflict zone (FW_W).

$$E_I = \{\forall e \in E | sink(e) \in FW_I\}, \tag{12}$$

where $sink(e) \in FW_I$ represents edges that their sink is in the set of waypoints that are inside the conflict zone (FW_I).

$$E_D = \{\forall e \in E | sink(e) \in FW_W\}, \tag{13}$$

where $FW_W = \{v \in FW | dist(v, C) < d_{SAFE}\}$ is the set of waypoints that are on the future path of the CAV (FW) and are within the d_{SAFE} distance of the conflict zone (C).

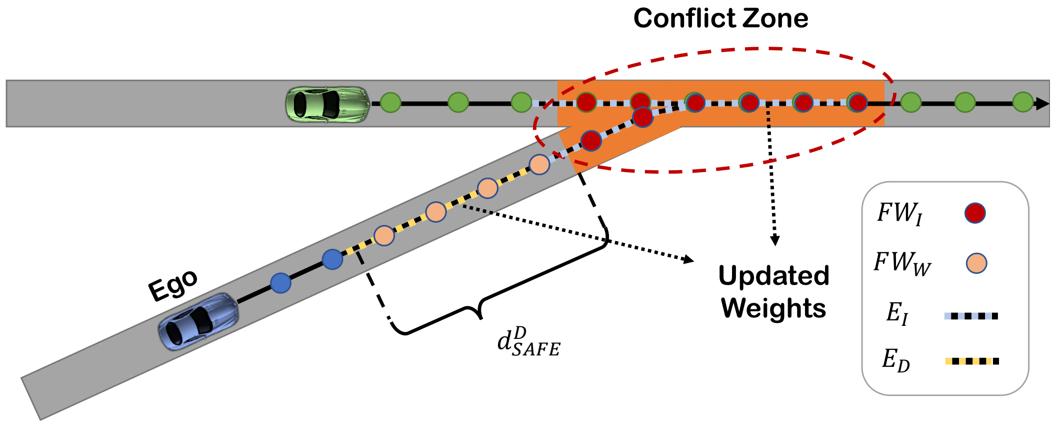


Fig. 8. Weights of the edges on the path of the other CAV and edges on the path of the ego CAV are updated to account for the presence of other CAVs as well as the conflict zone and the required safe distance.

To find the weight for each segment of the road that has a distance of d_C from the conflict zone, the maximum safe velocity is computed using Equation 14.

$$v_{SAFE} = \frac{-(2\rho a_{ACC} + 2|a_{brake}|) + \sqrt{\Delta_S}}{2}, \tag{14}$$

where $\Delta_S = 4(a^2_{brake} + 2a_{ACC}\rho a_{brake} - a_{ACC}\rho^2 a_{brake} - 2d_C|a_{brake}|)$. Equation 14 is determined by solving Equation 4 for v_D when the distance from the conflict zone is d_C. Then the weight for each edge is computed as:

$$w_i = \frac{l}{v^i_{SAFE}}, \tag{15}$$

where l is the length of the edge. For the edges that are inside the conflict zone, the minimum V_{SAFE} among all segments is used, i.e., the case where $d_C = 0$.

One benefit of this approach is that the ego CAV automatically explores other paths (if any) that are not involved in the conflict zone. Figure 9 shows a takeover scenario where the ego CAV (Blue) has initially a conflict with its front vehicle (Green). After updating the map and computing the shortest path, the ego CAV finds a new path that is conflict-free.

Remark *The main benefit of using a graph map for navigation is reducing the problem from 2D navigation to 1D navigation. By integrating the shortest path algorithm (for waypoints) with longitudinal control, the vehicle can find the optimal trajectory.*

As a result of this modification, fewer constraints are defined. For example, we don't have to define separate constraints for driving within the road boundary Furthermore, constraints are simpler to enforce, e.g., we only have to check the longitudinal distance instead of overlap checking for two rectangles. Therefore, the optimization problem is easier to solve and the solution is computed in a shorter time. To this end, we separately compute the control input for heading control and velocity control.

4.5.2 Heading Controller. The motion controller uses the set of FW to calculate the reference heading angle θ_{ref} of the CAV. For the desired heading angle (θ_{ref}), the motion controller selects a look-ahead point similar to the pure pursuit algorithm [34] and calculates the bearing angle from

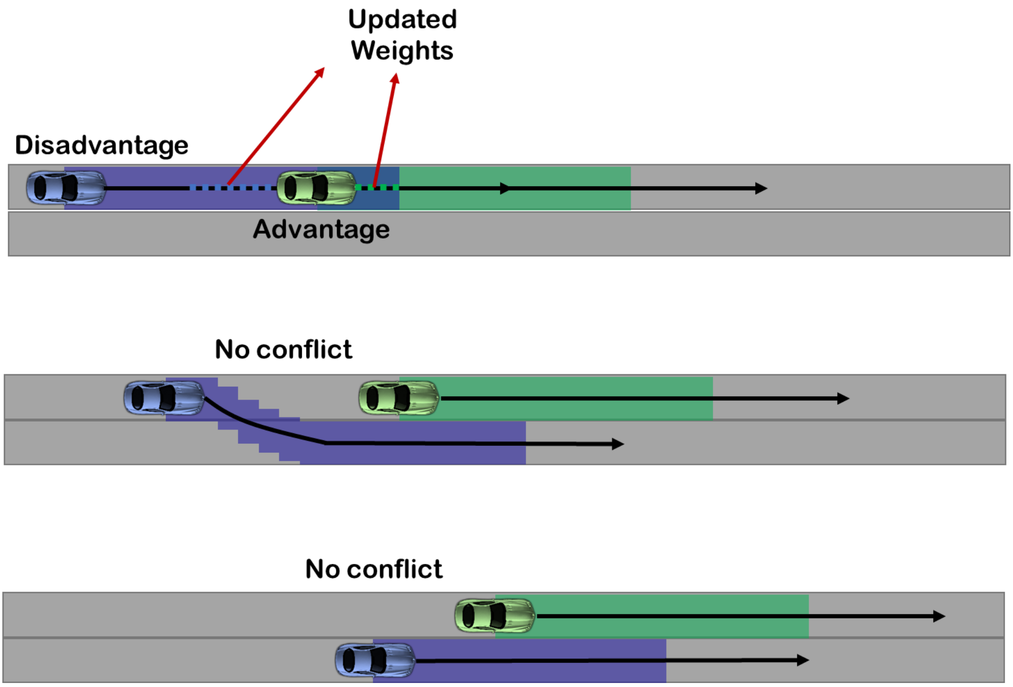


Fig. 9. Top) The initial situation with a conflict zone. Mid) Ego CAV selects another path that is conflict-free. Bottom) Ego CAV continues with the new path.

its current location (x, y) to the look-ahead point:

$$\theta_{ref} = atan2(x - x_l, y - y_l), \tag{16}$$

where x_l and y_l correspond to the x-y coordinate of the look-ahead point. Next, a Proportional Integral Derivative (PID) controller is utilized to calculate the steering angle of the CAV:

$$\psi = k_P e_\theta + k_I \int e_\theta + k_D \dot{e}_\theta, \tag{17}$$

where k_P, k_I, k_D are constant (controller gains) that are tuned to achieve a fast response with no/small overshoot (short settling time), and $e_\theta = \theta_{ref} - \theta$.

4.5.3 *Velocity Controller.*

For the velocity control, we use the Control Barrier Function (CBF) and Control Lyapunov Function (CLF) to enforce the desired behavior.

We rewrite vehicle dynamics (Eq. (2))as:

$$\dot{X} = f(X) + g(X)u, \tag{18}$$

where $X = [x\ y]^T$ is the state vector. We define a minimum energy cost function (J) as:

$$J = \int_{t_0}^{t_f} u^2 dt, \tag{19}$$

where t_0 ad t_f indicate the initial and final times. The first objective is that each CAV follows its desired velocity (v_D). If we select a Lyapunov candidate as:

$$V(X) = (v - v_D)^2. \tag{20}$$

Then, the control input should satisfy:

$$\dot{V}(X) + \epsilon V(X) \leq 0, \tag{21}$$

or

$$L_f V(X) + L_g V(X) u + \epsilon V(X) \leq 0, \tag{22}$$

where L_f and L_g represent the Lie derivatives of $V(X)$ with respect to the vector fields f and g. We relax this constraint by using a relaxation variable r_V and rewrite the above constraint as:

$$L_f V(X) + L_g V(X) u + \epsilon V(X) \leq r_V. \tag{23}$$

Later, we rewrite the cost function by adding r_V and trying to minimize it. Substituting system dynamics into the above constraint, we have:

$$C_0 := 2u(v - v_D) + \epsilon(v - v_D)^2 \leq r_V. \tag{24}$$

For the first safety constraint (Eq. 5), we define a barrier function as:

$$b_1(X) := x_c - x - d^D_{SAFE} > 0, \tag{25}$$

where x_c is the location of the conflict zone projected on the path of the ego CAV. Since d_{SAFE} is a function of CAV's velocity (v), the relative degree of the barrier function is one. After substituting d^D_{SAFE}, we can re-write the barrier function ($b_1(X)$ as:

$$x_c - x - v\rho - 0.5a_{max}\rho^2 - \frac{(v + a_{max}\rho)^2}{2a_{min}} + d_{scenario} - \Delta. \tag{26}$$

As a result, the input (u) should satisfy the following constraint to enforce Equation 25:

$$L_f b_1(X) + L_g b_1(X) u + b_1(X) \geq 0. \tag{27}$$

We represent this constraint as C_1:

$$C_1 := -v - u\left(\rho + \frac{v+a_{max}\rho}{a_{min}}\right) + x_c - x - v\rho - 0.5a_{max}\rho^2 - \frac{(v+a_{max}\rho)^2}{2a_{min}} + d_{scenario} - \Delta \geq 0. \tag{28}$$

For the second safety rule (Eq. 6), we define another barrier function as:

$$b_2(X) := x_c - x - v\rho - 0.5a_{max}\rho^2 > 0, \tag{29}$$

and the control input should satisfy:

$$C_2 := -v - u\rho + x_c - x - v\rho - 0.5a_{max}\rho^2 \geq 0. \tag{30}$$

Since there are constraints on the Vehicle's speed, we define two more barrier functions b_1 and b_2 as:

$$\begin{cases} b_3(X) := v_{max} - v \geq 0; \\ b_4(X) := v - v_{min} \geq 0. \end{cases} \tag{31}$$

Therefore, the control input (u) should satisfy C_2 and C_3 constraints:

$$\begin{cases} C_3 := -u + v_{max} - v \geq 0; \\ C_4 := u + v - v_{min} \geq 0. \end{cases} \tag{32}$$

4.5.4 Reformulation to A Quadratic Programming Problem. To use Quadratic Programming (QP), we discretize the problem where the time is discretized at every Δt and the input is assumed to be constant during each interval. Then, we can formulate the Equation 19 as a QP problem:

$$U^* = \arg\min \frac{1}{2}u^2 + \frac{1}{2}pr_v^2, \tag{33}$$

subjected to:

$$\begin{cases} 2u(v - v_D) + \epsilon(v - v_D)^2 \leq r_v; \\ -v - u(\rho + \frac{v+a_{max}\rho}{a_{min}}) + x_c - x - v\rho - 0.5a_{max}\rho^2 - \frac{(v+a_{max}\rho)^2}{2a_{min}} + d_{scenario} - \Delta \geq 0; \\ -v - u\rho + x_c - x - v\rho - 0.5a_{max}\rho^2 \geq 0; \\ -u + v_{max} - v \geq 0; \\ u + v - v_{min} \geq 0; \\ u_{min} < u < u_{max}, \end{cases} \tag{34}$$

where $p > 0$ is the weight for CLF. We can rewrite the constraints as:

$$AU \leq b, \tag{35}$$

where $U = [u \ r_v]^T$ and

$$A = \begin{bmatrix} 2(v - v_D) & -1 \\ \rho + \frac{v+a_{max}\rho}{a_{min}} & 0 \\ \rho & 0 \\ 1 & 0 \\ -1 & 0 \end{bmatrix}, \tag{36}$$

and

$$b = \begin{bmatrix} -\epsilon(v - v_D)^2 \\ -v(\rho + 1) + x_c - x - 0.5a_{max}\rho^2 - \frac{(v+a_{max}\rho)^2}{2a_{min}} + \\ d_{scenario} - \Delta \\ x_c - x - v(\rho + 1) - 0.5a_{max}\rho^2 \\ v_{max} - v \\ v - v_{min} \end{bmatrix}. \tag{37}$$

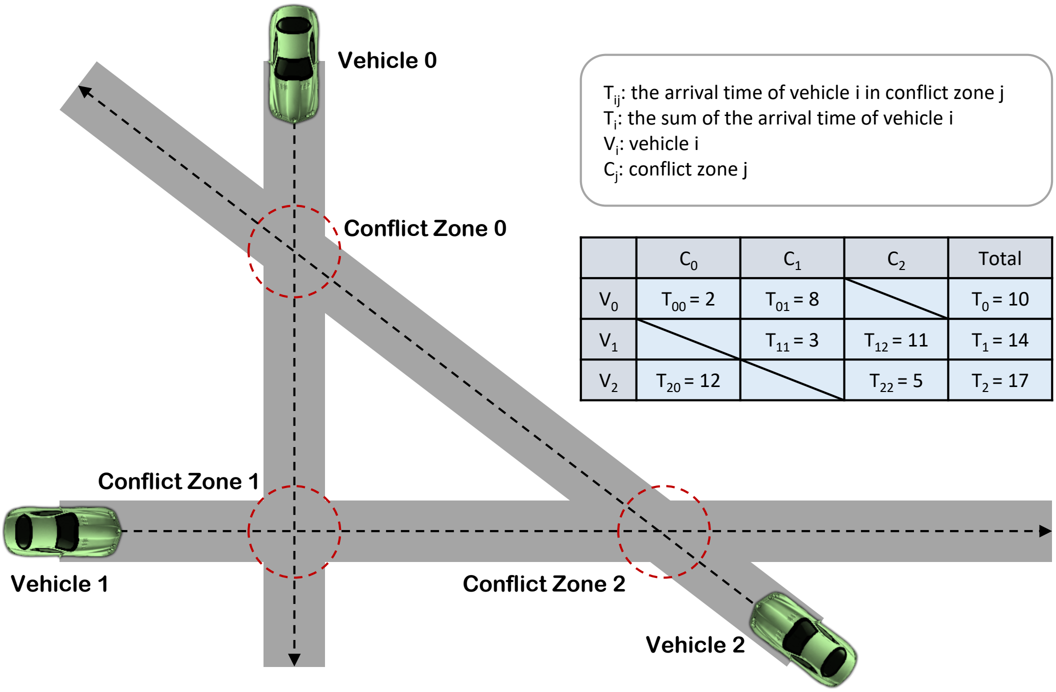
After solving the above QP, the optimal control input (u^*) is determined and applied to the CAV.

5 DEADLOCK DETECTION AND RESOLUTION

There can be scenarios where CAVs are involved in a deadlock, i.e., they wait for each other indefinitely. To avoid deadlock, we propose a deadlock detection and resolution algorithm that benefits from shared information of CAVs.

5.1 Deadlock Detection

In order to detect a deadlock, all CAVs create a directed graph called the *dependency graph*. The nodes of the dependency graph are vehicle IDs and edges of the graph indicate that if a CAV is yielding to another CAV over a conflict zone. There will be a directed edge from node V_i to node V_j if CAV V_i is yielding to the CAV V_j over a conflict zone. Since a CAV determines only the conflicts (dependencies) between itself and other CAVs –and not the conflicts between other CAVs, the constructed dependency graph is not complete. We refer to the dependency graph of each CAV as the “partial dependency graph” or PDG. To compute the complete graph, each CAV broadcasts its PDG to inform others about its interactions with other CAVs. From the received PDGs of other CAVs and the PDG of the ego CAV, the complete dependency graph (CDG) is constructed. To build the CDG, the ego CAV matches the IDs of nodes of its PDG with the IDs of nodes of received PDGs and updates its PDG by adding new nodes and edges (if necessary). Figure 11 shows a scenario with 5 CAVs and their corresponding PDGs as well as the final CDG.



T_{ij}: the arrival time of vehicle i in conflict zone j
T_i: the sum of the arrival time of vehicle i
V_i: vehicle i
C_j: conflict zone j

	C_0	C_1	C_2	Total
V_0	$T_{00} = 2$	$T_{01} = 8$		$T_0 = 10$
V_1		$T_{11} = 3$	$T_{12} = 11$	$T_1 = 14$
V_2	$T_{20} = 12$		$T_{22} = 5$	$T_2 = 17$

Fig. 10. Illustration of deadlock resolution using the cost score of the sum of arrival times. In this scenario, three vehicles navigate through three distinct conflict zones. When utilizing the individual arrival times at each conflict zone as the cost function, a potential deadlock scenario arises. For instance, vehicle 0 must yield to Vehicle 1 in conflict zone 1, vehicle 1 must yield to Vehicle 2 in conflict zone 2, and Vehicle 2 must yield to Vehicle 0 in conflict zone 0. In order to mitigate this issue, we propose a solution involving the summation of arrival times across all conflict zones to compute the cost score for each vehicle.

5.2 Deadlock Resolution

After constructing the CDG, each CAV checks if the CDG has a cycle. We use the Depth-First Search (DFS) algorithm to detect cycles. If a cycle is detected, each CAV calculates a cost score for all CAVs that are involved in the cycle based on their average time of arrival at their conflict zones. For instance, if CAV j has m conflicts, its cost score is calculated as:

$$S_j = \frac{\sum_{i=1}^{m} TOA_i^j}{m},$$

where TOA_i^j is the time of arrival of the CAV j at its ith conflict zone. To provide clarity regarding the rationale behind selecting the sum of arrival times over the individual arrival times at each conflict zone as the metric for determining the cost score, we present a demonstrative example in Figure 10. As shown in Figure 10, simply using the individual arrival time at each distinct conflict zone may potentially lead to a deadlock scenario. Thus, we calculate the cost score by aggregating the sum of the arrival times across all conflict zones. Then, all CAVs select the CAV with the least

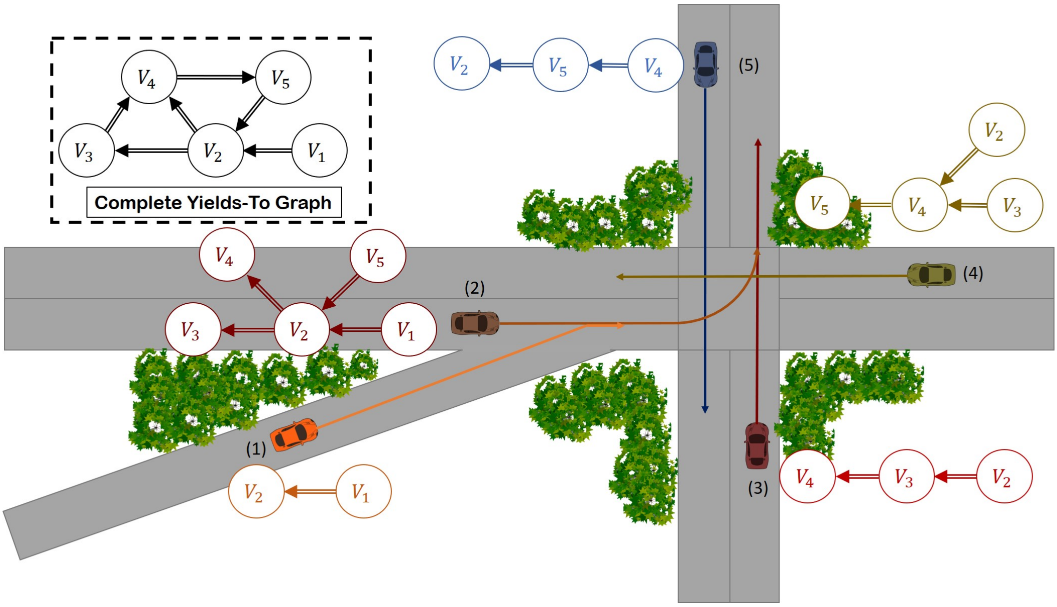


Fig. 11. Each CAV determines and broadcasts its PDG. After receiving other CAVs' PDGs, CAVs construct the CDG and can resolve deadlocks.

average time of arrival to have the advantage over all of its conflict zones. We refer to this CAV as the leader. Once the leader is determined, the direction of all outgoing edges from the leader's node is reversed. If after the calculation two CAVs have the same cost score, the CAV with the lower ID number will be selected as the leader to ensure consensus among CAVs. Since there can be more than one cycle in a graph, the deadlock resolution process is repeated until all cycles are removed. Figure 12 shows a scenario where there are two cycles that are removed in two iterations.

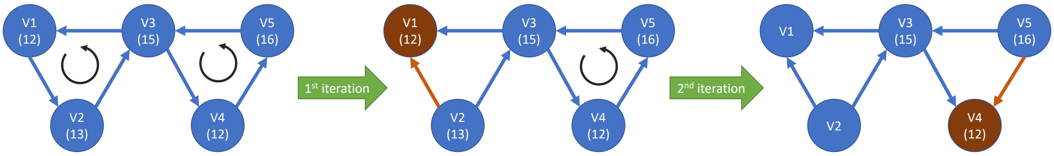


Fig. 12. The CDG for a scenario where there are two cycles. Orange nodes indicate the leader in each iteration and orange edges show the reversed dependencies.

Orange nodes indicate the selected leader after the detection of the cycles and orange edges show the reversed dependencies among CAVs after the leader selection.

Lemma *If the CDG has no cycles, then there is no deadlock involving the ego CAV.*

Proof *Once the CDG is modified to be acyclic, there is no path (set of sequential edges) starting at node* V_{ego} *that eventually loops back to node* V_{ego} *again, which means the ego CAV never yields to other CAVs that are yielding to the ego CAV and therefore, there is no deadlock involving the ego CAV.*

It takes some time to resolve a deadlock due to the vehicle's dynamic –CAVs cannot change their velocity and expected arrival time instantly. As a result, CAVs may face the same deadline again when they compute the CDG after T. However, we show that the result of the deadlock resolution will be the same (the same CAV will be selected as the leader) until the deadlock is resolved. Since the leader has the lowest average time of arrival in the first iteration, it does not yield to any other CAV while other CAVs involved in the deadlock slow down to yield to at least one CAV. Therefore, the average time of arrival of the leader will remain less than other CAVs in the second iteration and so on.

The main reason that the deadlock detection algorithm is executed repeatedly even after detection of a deadlock is because the leader may change its plan e.g., slow down and stop, and therefore, it is necessary to select another leader.

6 EXPERIMENTAL RESULT

We evaluated the scalability of our algorithm on a simulator that is developed in Matlab. We created a tool in Python to automatically extract a desired map from the OpenStreetMap[1] (OSM format) and then generate the world map graph for it. Once the map is generated, a driving scenario is created by randomly selecting the initial position and velocity as well as the destination for n CAVs. In Figure 13, a randomly generated map from openStreetMap, its corresponding map graph, and a random scenario with 20 CAVs are depicted. We used the differential equations represented in (1) to model the vehicle's behavior in 2D. CAVs communication delay is modeled by queuing the broadcast packets. The size of each vehicle is 5x2 m, the lane width is 5 m and the distance between waypoints on the map is 0.5 m. Gains of the controller for the heading controller are $K_P = 5$ and $K_D = 0.1$. Other parameters of the vehicle are listed in Table 1.

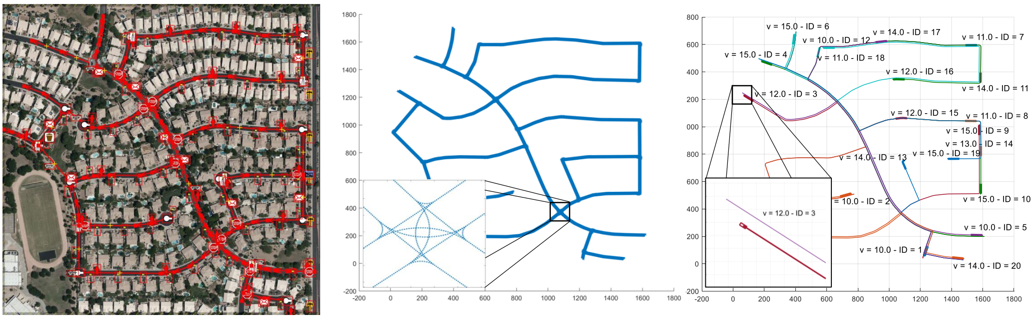

1800
1600
1400
1200
1000
800
600
400
200
0
-200
-200
0
200
400
600
800
1000
1200
1400
1600
1800
v = 15.0 - ID = 6
v = 10.0 - ID = 12
v = 14.0 - ID = 17
v = 11.0 - ID = 7
v = 15.0 - ID = 4
v = 11.0 - ID = 18
v = 12.0 - ID = 16
v = 14.0 - ID = 11
v = 12.0 - ID = 3
v = 12.0 - ID = 15
v = 11.0 - ID = 8
v = 15.0 - ID = 9
v = 13.0 - ID = 14
v = 15.0 - ID = 19
v = 14.0 - ID = 13
= 10.0 - ID = 2
v = 15.0 - ID = 10
v = 10.0 - ID = 5
v = 10.0 - ID = 1
v = 14.0 - ID = 20
v = 12.0 - ID = 3

Fig. 13. A snapshot of a map retrieved from the OpenStreetMap (left), its corresponding directed graph in MATLAB (middle), and a scenario with randomly spawned vehicles on the map (Right).

v_{min}	v_{max}	a_{min}	a_{max}	ψ_{max}	T	ρ
$0\frac{m}{s}$	$23\frac{m}{s}$	$-8\frac{m}{s^2}$	$5\frac{m}{s^2}$	$\frac{\pi}{3}rad$	$0.1s$	$0.2s$

Table 1. Parameters of the CAVs for simulation.

6.1 Safety Evaluation

To demonstrate the safety of the proposed algorithm, we created a merge and an intersection scenario where two CAVs have a conflict on their future path as depicted in Figure 14.

To verify that CAVs are always safe, we force the CAV with the advantage to suddenly decelerate at different times. We show that no accident will happen regardless of the deceleration time and

---

[1] https://www.openstreetmap.org/

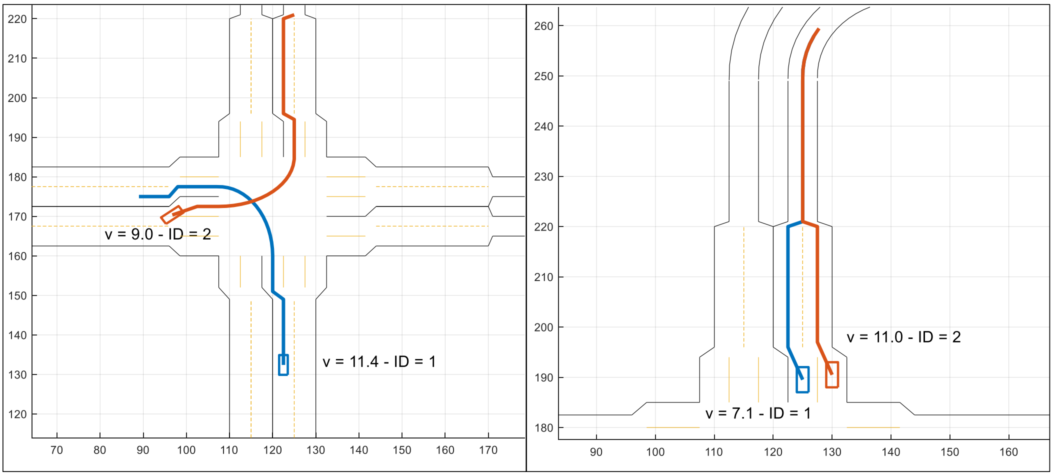
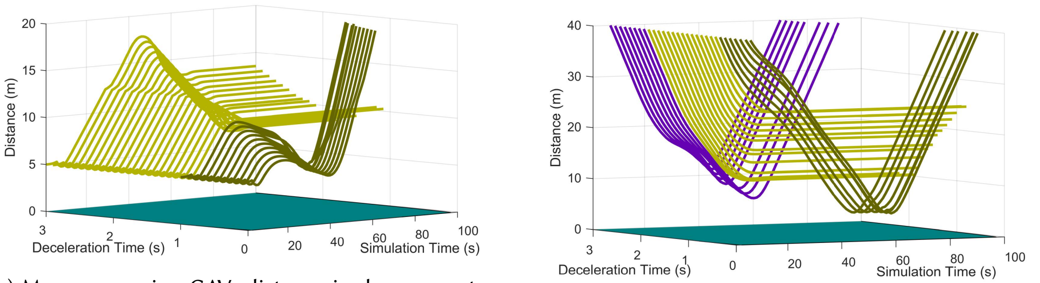


Fig. 14. An Intersection and a merge scenario are created. The CAV with advantage suddenly decelerates and stops.

CAVs maintain a minimum safe distance of 5 meters. Using brute-force testing, the deceleration time of the CAV with the advantage varies in a 30-second interval with a 0.1 s increment that includes critical times that stop inside the conflict zone. Figures 15b and 15a show the distance between CAVs for the intersection and merge scenarios, respectively.



(a) Merge scenario - CAVs distance is always greater than a threshold regardless of the deceleration time of the CAV with the advantage –the CAV with the advantage may stop before entering the conflict zone (dark green) or inside the conflict zone (yellow). The scenario will be like the same lane following after entering the merge.

(b) Intersection scenario - CAVs distance is always greater than a threshold regardless of the deceleration time of the CAV with the advantage –the CAV with the advantage may stop before entering the conflict zone (dark green), inside the conflict zone (yellow), or after the conflict zone (blue).

Fig. 15. Brute-force evaluation of an intersection and a merge scenario.

In the intersection scenario, the CAV with the advantage may stop before, inside, or after the conflict zone where distances between CAVs are depicted in dark green, yellow, and blue colors, respectively. For cases where the CAV with advantage stops before or after the conflict zone, the CAV with disadvantage continues while in cases where the CAV with advantage stops inside the conflict zone, the CAV with disadvantage slows down and stops (depicted in yellow). In the merge scenario, the conflict zone moves with the CAV with the advantage after it reaches the merging

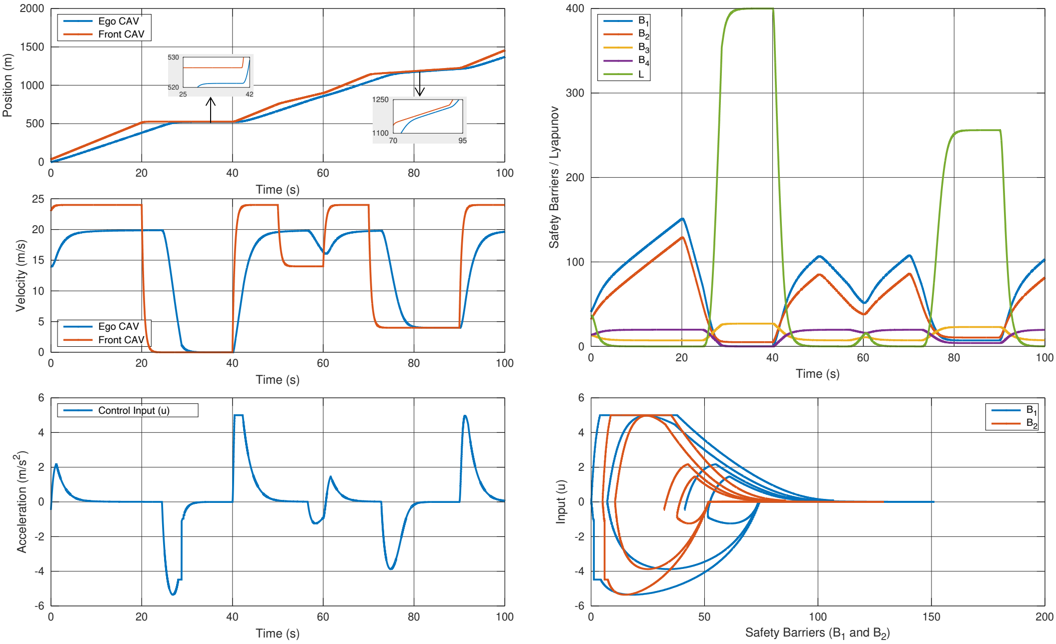


Fig. 16. Results with ideal sensors. Left top) Position and Velocity of CAVs. Top right) Values of the barrier functions b1, b2, b3, and b4 as well as the Lyapunov function L. Bottom left) control input from solving the QP. Bottom right) Changing trend of the barrier function vs input with time.

point. As a result, the CAV with the advantage either stops before the conflict zone or inside it. For cases where the CAV with the advantage stops before the merging point, the CAV with the disadvantage continues and enters the merge (depicted in dark green) and for the rest of the cases, the CAV with the disadvantage slows down and stops (depicted in yellow).

6.1.1 Evaluation of the Control Barrier Approach with Ideal Sensors. Our approach not only computes the optimal acceleration for the CAV but also ensures maintaining the desired velocity when possible and ensures safety. To demonstrate this, we consider a stop-and-go scenario where the ego CAV is following another CAV. Please be aware that in this section, we operate under the assumption that the sensor is perfect and free from any sensing errors. The initial velocity of the ego CAV and the front CAV are 14 m/s and 23 m/s, respectively. The distance between two CAVs is 35 m. Maximum and minimum speed limits are 27 m/s and 0 m/s, respectively. The maximum acceleration is 5 m/s^2, and the minimum acceleration is -8 m/s^2. The selected Δt is 0.01 and ϵ is 1. The response time of the ego CAV (ρ) is 200 ms, and the desired reference velocity is set to $v_d = 24$ m/s. Since this is a same-lane scenario, the $d_{scenario}$ is zero. Our stop-and-go scenario has two stops and a slowdown between two stops. Figure 16 top left shows the position and velocity of CAVs. The zoom areas show that the two CAVs keep a safe distance when the front CAV stops.

Figure 16 bottom left shows the control input computed after solving the QP problem (Section 4). Figure 16 top right shows the value of the barrier functions as defined in Section 4 and the Lyapunov function. We can observe that all barrier functions are always positive, indicating that vehicle behavior is safe. Around time $t = 25$ and $t = 75$, the distance between CAVs reduces, and the values of the first and second barrier functions (safety requirements) and the fourth barrier function (minimum speed limit requirement) get closer to zero. On the other hand, the value of the Lyapunov

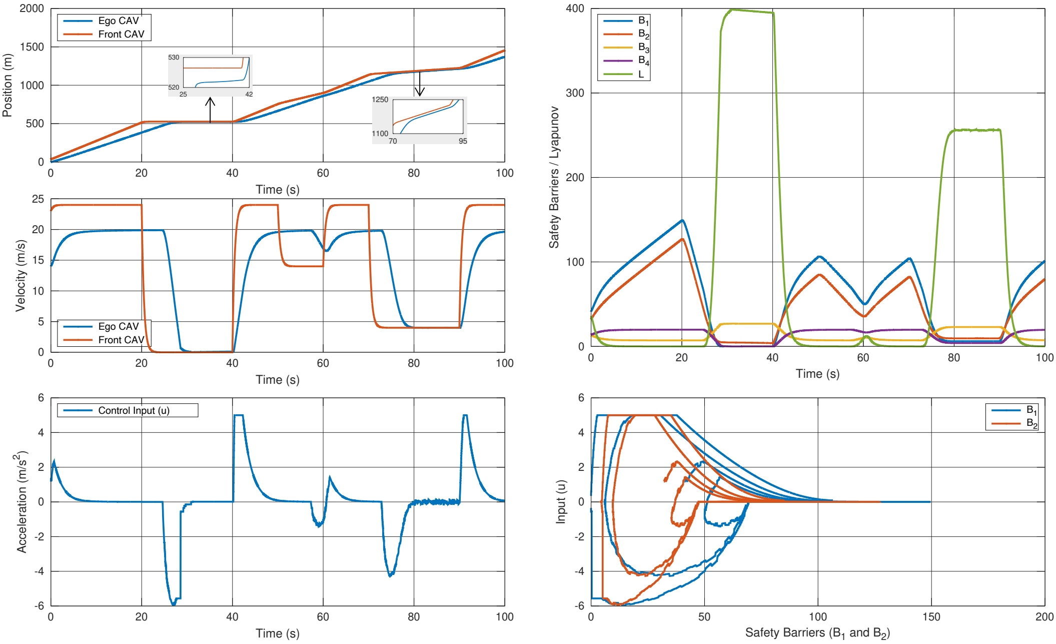


Fig. 17. Results with sensing errors. Left top) Position and Velocity of CAVs. Top right) Values of the barrier functions b1, b2, b3, and b4 as well as the Lyapunov function L. Bottom left) control input from solving the QP. Bottom right) Changing trend of the barrier function vs input with time.

function increases, meaning that the desired reference velocity (v_d) cannot be tracked. The bottom right figure of Figure 16 shows the value of the first barrier function (b_1) and the second barrier function (b_2) versus control input (u).

6.1.2 Evaluation of the Control Barrier Approach with Sensing Errors. The previous section (Section 6.1.1) demonstrates that the proposed control barrier approach enables CAVs to maintain a safe distance when equipped with ideal sensors and a detection algorithm. Nevertheless, accounting for sensing errors is imperative to uphold safety guarantees in the face of real-world uncertainties. Specifically, we account for sensing errors related to the perceived location of the front CAV. The sensed location of the front CAV ($\widehat{x}_2$) is expressed as follows:

$$\widehat{x}_2 = x_2 + k \cdot r \cdot (x_2 - x_1), \tag{38}$$

where x_1 denotes the location of the ego CAV, x_2 is the actual location of the front CAV, r is a uniformly distributed random variable in the range $[-1, 1]$, and k represents the maximum sensing error constant ($k > 0$). We set k to 0.1, indicating that the maximum sensing error of the front vehicle is 10 cm per meter. Note that the maximum sensing error gain (k) is assumed to be known in order to provide safety guarantees in the presence of uncertainty.

Since the sensed location is not precise, the formulation will account for the worst-case scenario when determining the safe distance. The worst-case scenario occurs when $r = 1$ (front car is assumed to be farther than actual):

$$\widehat{x}_2^{max} = x_2 + k \cdot (x_2 - x_1).$$

As a result, Equation 25 and 26 should be re-computed with the new value for the location of the front CAV assuming that the front CAV is closer than the actual (r = -1):

$$x_2^{min} = x_2 - k \cdot (x_2 - x_1).$$

A comparison between the results from Figure 16 and Figure 17 shows that Figure 17 closely resembles the result from Figure 16, albeit with some minor fluctuations. In control barrier functions of Figure 17, b_1 and b_2 remain positive throughout the scenario indicating that ego CAV always maintains a safe distance from its front vehicle despite the presence of sensor uncertainties.

6.2 Deadlock Resolution Demonstration

We created a deadlock situation at the intersection to evaluate our deadlock detection and resolution approach in Figure 18. The right part of Figure 18 shows the CCG for the scenario. We fixed the paths of CAVs to make a left turn at the intersection while having the same distance from the intersection and the same velocity. We simulated CAVs' behavior with and without our deadlock detection. Figure 19 shows the velocities of CAVs for both cases. In the case that no deadlock resolution is done, CAVs slow down to yield to other CAVs and eventually stop and will wait forever. For the case with deadlock resolution, CAVs slow down at first but speed up when their conflict zone is cleared. We can observe that after 7s, all CAVs reach their desired velocity (10m/s) while in the case with no deadlock detection, their velocity converges to zero.

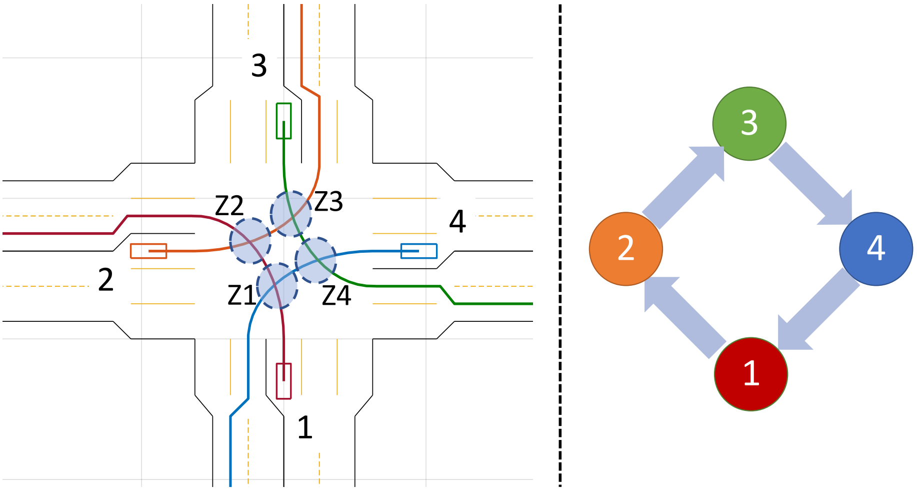


Fig. 18. A deadlock scenario where 4 CAVs approach the intersection with the same velocity (left) and the corresponding CDG (right).

6.3 Efficiency Evaluation

To evaluate the efficiency of our approach, we compared the performance of our approach with the case that vehicles are autonomous but not connected. For the non-connected case, the intersections are managed by stop signs, and all other conflicts among CAVs are handled by the AV's perception system e.g. adaptive cruise control (ACC) system. We extracted a map from the OpenStreetMap (Figure 13) and simulated three scenarios, i) light traffic with 5 vehicles, ii) moderate traffic with 10 vehicles, and iii) heavy traffic with 20 vehicles being present at the same time. When a vehicle exits the map boundary, a new vehicle is spawned. We measured the average velocities of CAVs and

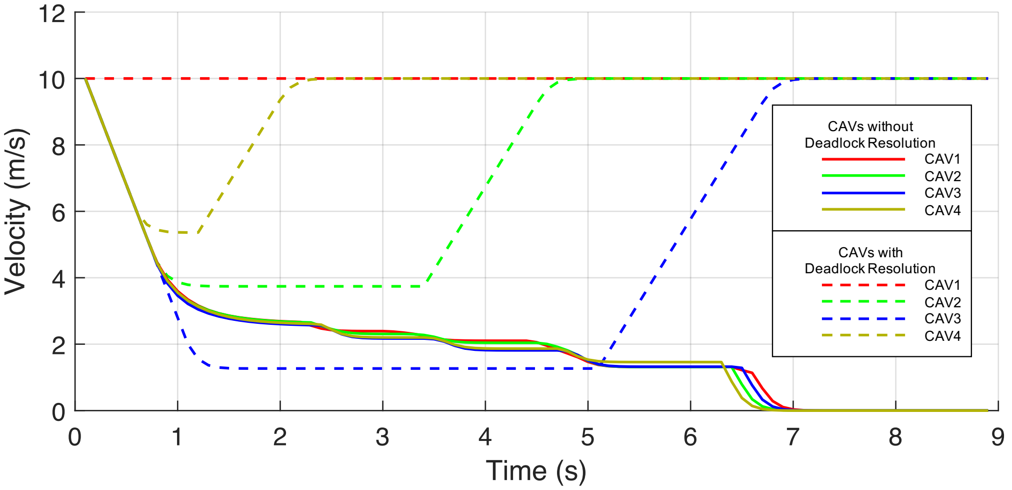


Fig. 19. Velocity profiles of CAVs with and without deadlock resolution for the scenario in Figure 18

reported them in Table 2. We also computed the fuel consumption of CAVs using the following model [35] and reported them in Table 2:

$$f = \begin{cases} 0, & \text{if } P_T > 0; \\ \frac{f_i}{3600} + \beta_1 P_T + \beta_2 a P_I, & \text{otherwise,} \end{cases} \tag{39}$$

where $P_T = \min(P_{max}, P_C + P_I)$ is the total tractive power (kW), $P_C = b_1 v + b_2 v^3$ is the cruise component of total power (kW), $P_I = \frac{mav}{1000}$ is the inertia component of the total power (kW), $f_i = 888.8mL/h$ is the instantaneous fuel consumption rate (mL/s), P_{max} is the maximum engine power (kW), m is the vehicle mass, a and v are the instantaneous acceleration and velocity, b_1 is rolling resistant factor (kN), and b_2 is the aerodynamic drag factor (kN/(m/s)^2), β_1 and β_2 are the efficiency factors for non-accelerating and accelerating cases.

	Light Traffic		Moderate Traffic		Heavy Traffic	
	AVs	CAVs	AVs	CAVs	AVs	CAVs
Avg Vel.	10.5	11.5	10.9	11.8	11.2	11.9
Avg Fuel Con.	1.27	0.49	1.08	0.47	1.01	0.48

Table 2. Comparing the average velocity (m/s) and fuel consumption (mL/s) of vehicles when they navigate autonomously (non-connected) and cooperatively (connected).

With the help of shared information, CAVs not only drive at higher velocities, they drive smoother than non-connected cases because they slow down and stop less frequently and therefore, their fuel consumption is less than the connected case.

7 CONCLUSION AND FUTURE WORKS

In this paper, a new definition is introduced for the RSS rules that can be applied to any scenario, and CAVs' safety is ensured by considering the worst-case scenario. Next, we presented a cooperative driving algorithm for CAVs based on proposed RSS rules. Our algorithm can also detect and resolve deadlocks in a distributed manner. The correctness of our approach is verified by conducting experiments using our simulator. Future works include taking into account various fault models (e.g. the network delay is larger than T or a CAV is unable to communicate, a CAV is being compromised and lies about its position and/or its future trajectory, etc.) with the help of infrastructure (e.g. installed cameras). While this work does not extensively explore communication protocols, it is important to recognize that effective communication significantly contributes to the seamless coordination of actions among CAVs. Therefore, enabling efficient and reliable communication among CAVs is also a promising avenue for future research.

8 ACKNOWLEDGEMENT

This work was partially supported by funding from NIST Award 70NANB19H144, Semiconductor Research Corporation (SRC) project 3154, and National Science Foundation (NFS) grants CNS 1525855 and CPS 1645578. This work was also partially supported by MOE in Taiwan under Grant Number NTU-112V2003-1 and NSTC in Taiwan under Grant Numbers NSTC-112-2636-E-002-010 and NSTC-112-2221-E-002-168-MY3.

REFERENCES

[1] MG Mohanan and Ambuja Salgoankar. A survey of robotic motion planning in dynamic environments. *Robotics and Autonomous Systems*, 100:171–185, 2018.

[2] Federico Rossi, Saptarshi Bandyopadhyay, Michael Wolf, and Marco Pavone. Review of multi-agent algorithms for collective behavior: a structural taxonomy. *IFAC-PapersOnLine*, 51(12):112–117, 2018.

[3] Mohammad Khayatian, Mohammadreza Mehrabian, Edward Andert, Rachel Dedinsky, Sarthake Choudhary, Yingyan Lou, and Aviral Shirvastava. A survey on intersection management of connected autonomous vehicles. *ACM Transactions on Cyber-Physical Systems*, 4(4):1–27, 2020.

[4] Bowen Zheng, Chung-Wei Lin, Shinichi Shiraishi, and Qi Zhu. Design and analysis of delay-tolerant intelligent intersection management. *ACM Transactions on Cyber-Physical Systems*, 4(1):1–27, 2019.

[5] Mohammad Khayatian, Rachel Dedinsky, Sarthake Choudhary, Mohammadreza Mehrabian, and Aviral Shrivastava. R 2 im-robust and resilient intersection management of connected autonomous vehicles. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE, 2020.

[6] Yi-Ting Lin, Hsiang Hsu, Shang-Chien Lin, Chung-Wei Lin, Iris Hui-Ru Jiang, and Changliu Liu. Graph-based modeling, scheduling, and verification for intersection management of intelligent vehicles. *ACM Transactions on Embedded Computing Systems (TECS)*, 18(5s):1–21, 2019.

[7] Mohammad Khayatian, Yingyan Lou, Mohammadreza Mehrabian, and Aviral Shirvastava. Crossroads+ a time-aware approach for intersection management of connected autonomous vehicles. *ACM Transactions on Cyber-Physical Systems*, 4(2):1–28, 2019.

[8] Xiao-Yun Lu and J Karl Hedrick. Longitudinal control algorithm for automated vehicle merging. *International Journal of Control*, 76(2):193–202, 2003.

[9] Jackeline Rios-Torres and Andreas A Malikopoulos. Automated and cooperative vehicle merging at highway on-ramps. *Transactions on Intelligent Transportation Systems*, 18(4):780–789, 2016.

[10] Shunsuke Aoki and Ragunathan Rajkumar. A merging protocol for self-driving vehicles. In *2017 ACM/IEEE 8th International Conference on Cyber-Physical Systems (ICCPS)*, pages 219–228. IEEE, 2017.

[11] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. On a formal model of safe and scalable self-driving cars. *arXiv preprint arXiv:1708.06374*, 2017.

[12] Changliu Liu, Chung-Wei Lin, Shinichi Shiraishi, and Masayoshi Tomizuka. Distributed conflict resolution for connected autonomous vehicles. *IEEE Transactions on Intelligent Vehicles*, 3(1):18–29, 2017.

[13] Florent Perronnet, Jocelyn Buisson, Alexandre Lombard, Abdeljalil Abbas-Turki, Mourad Ahmane, and Abdellah El Moudni. Deadlock prevention of self-driving vehicles in a network of intersections. *IEEE Transactions on Intelligent Transportation Systems*, 20(11):4219–4233, 2019.

[14] Reza Azimi, Gaurav Bhatia, Ragunathan Raj Rajkumar, and Priyantha Mudalige. Stip: Spatio-temporal intersection protocols for autonomous vehicles. In *ICCPS'14: ACM/IEEE 5th International Conference on Cyber-Physical Systems (with CPS Week 2014)*, pages 1–12. IEEE Computer Society, 2014.

[15] Mohammad Khayatian, Mohammadreza Mehrabian, Harshith Allamsetti, Kai-Wei Liu, Po-Yu Huang, Chung-Wei Lin, and Aviral Shrivastava. Cooperative driving of connected autonomous vehicles using responsibility-sensitive safety

(rss) rules. In *Proceedings of the ACM/IEEE 12th International Conference on Cyber-Physical Systems*, pages 11–20, 2021.

[16] Mohammad Khayatian, Mohammadreza Mehrabian, and Aviral Shrivastava. Rim: Robust intersection management for connected autonomous vehicles. In *Real-Time Systems Symposium*, pages 35–44. IEEE, 2018.

[17] Edward Andert, Mohammad Khayatian, and Aviral Shrivastava. Crossroads: Time-sensitive autonomous intersection management technique. In *Proceedings of the 54th Annual Design Automation Conference 2017*, page 50. ACM, 2017.

[18] Masoud Bashiri and Cody H Fleming. A platoon-based intersection management system for autonomous vehicles. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 667–672. IEEE, 2017.

[19] Rachel Dedinsky, Mohammad Khayatian, Mohammadreza Mehrabian, and Aviral Shrivastava. A dependable detection mechanism for intersection management of connected autonomous vehicles (interactive presentation). In *Workshop on Autonomous Systems Design (ASD 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[20] Lejla Banjanovic-Mehmedovic et al. Autonomous vehicle-to-vehicle (v2v) decision making in roundabout using game theory. *Int. J. Adv. Comput. Sci. Appl*, 7:292–298, 2016.

[21] Bai Li, Yue Zhang, Youmin Zhang, and Ning Jia. Cooperative lane change motion planning of connected and automated vehicles: A stepwise computational framework. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 334–338. IEEE, 2018.

[22] Jianqiang Nie, Jian Zhang, Wanting Ding, Xia Wan, Xiaoxuan Chen, and Bin Ran. Decentralized cooperative lane-changing decision-making for connected autonomous vehicles. *IEEE Access*, 4:9413–9420, 2016.

[23] Pedro Fernandes and Urbano Nunes. Platooning of autonomous vehicles with intervehicle communications in sumo traffic simulator. In *13th International IEEE Conference on Intelligent Transportation Systems*, pages 1313–1318. IEEE, 2010.

[24] Siyuan Gong, Anye Zhou, and Srinivas Peeta. Cooperative adaptive cruise control for a platoon of connected and autonomous vehicles considering dynamic information flow topology. *Transportation Research Record*, 2673(10):185–198, 2019.

[25] Michael Duering and Patrick Pascheka. Cooperative decentralized decision making for conflict resolution among autonomous agents. In *International Symposium on Innovations in Intelligent Systems and Applications*, pages 154–161. IEEE, 2014.

[26] Michael During and Karsten Lemmer. Cooperative maneuver planning for cooperative driving. *IEEE Intelligent Transportation Systems Magazine*, 8(3):8–22, 2016.

[27] Yimin Chen, Chao Lu, and Wenbo Chu. A cooperative driving strategy based on velocity prediction for connected vehicles with robust path-following control. *IEEE Internet of Things Journal*, 2020.

[28] Kostas E Bekris, Konstantinos I Tsianos, and Lydia E Kavraki. A decentralized planner that guarantees the safety of communicating vehicles with complex dynamics that replan online. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3784–3790. IEEE, 2007.

[29] Jufeng Peng and Srinivas Akella. Coordinating multiple robots with kinodynamic constraints along specified paths. *The International Journal of Robotics Research*, 24(4):295–310, 2005.

[30] Mohammad Hekmatnejad et al. Encoding and monitoring responsibility sensitive safety rules for automated vehicles in signal temporal logic. In *17th ACM-IEEE Conference on Formal Methods and Models for System Design*, pages 1–11, 2019.

[31] Mohammad Farhadi, Mehdi Ghasemi, Sarma Vrudhula, and Yezhou Yang. Enabling incremental knowledge transfer for object detection at the edge. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 396–397, 2020.

[32] Wei Xiao, Christos G Cassandras, and Calin Belta. Decentralized merging control in traffic networks with noisy vehicle dynamics: A joint optimal control and barrier function approach. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3162–3167. IEEE, 2019.

[33] Wei Xiao and Calin Belta. Control barrier functions for systems with high relative degree. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 474–479. IEEE, 2019.

[34] R Craig Coulter. Implementation of the pure pursuit path tracking algorithm. Technical report, Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, 1992.

[35] Rahmi Akçelik, Robin Smit, and Mark Besley. Calibrating fuel consumption and emission models for modern vehicles. In *IPENZ transportation group conference, Rotorua, New Zealand*, 2012.