Master's Thesis

Title

Collision avoidance method for intersection accidents using 5G edge computing environments in Cellular-V2X

Supervisor

Associate Professor Shin'ichi Arakawa

Author

Ryu Yachikojima

February 10th, 2022

Department of Information Networking

Graduate School of Information Science and Technology

Osaka University

Master's Thesis

Collision avoidance method for intersection accidents using 5G edge computing environments in Cellular-V2X

Ryu Yachikojima

Abstract

A key to achieving advanced safety in autonomous vehicles is to perceive and understand the surrounding environment using many sensors equipped in the vehicle. However, because the sensors rely on the visibility of the vehicle, there are limitations to understanding the environment. Therefore, dynamic information sharing among vehicles is important to achieve sophisticated safety, in addition to the self-perception of the vehicle. Cellular vehicle-to-everything (C-V2X) using multi-access edge computing has attracted attention as a method to share information among vehicles and provide centralized safety validation of traffic. Particularly, in safety uses, such as collision detection at an intersection, it is essential to predict the uncertain position of vehicles with a probabilistic process, such as a Markov chain. However, the calculation time of the existing method is too large to meet real-time requirements. In this paper, we developed a probabilistic collision detection method for an edge-computing environment with a cellular system. For this purpose, we modified the existing probabilistic collision detection method. We reduced the calculation time to predict the probability distribution of a vehicle state by utilizing the prediction error between the current state and the predicted state. Furthermore, we implemented a simple simulation to evaluate the capacity of the edge server to handle vehicles and demonstrate that our method can handle more vehicles. Finally, we implement a MEC-based collision detection system with private 5G and driving simulator to evaluate the collision detection performance. As the evaluation result, the proposed method notifies all of the potential collision risks beforehand which is enough to stop the vehicles, whereas the existing method fails to notify more than 80% of the risks.

Keywords

MEC (Multi-access Edge Computing)

ITS (Intelligent Transport Systems)

C-V2X (Cellular Vehicle-to-Everything)

Markov chain

Collision Detection

CARLA (CAR Learning to Act)

Contents

List of Figures

List of Tables

1 Introduction

With the advancements in machine learning and its underlying computing equipment, self-driving vehicles have attracted significant attention. According to Ref. [1], there are six levels of vehicle autonomy, including no automation, driver assistance, and full automation. A key to achieving advanced safety in autonomous vehicles is to perceive and understand the environment surrounding a vehicle using sensors, such as cameras and LiDAR equipped in vehicles [2]. However, because sensors rely on visibility from the vehicle, there are limitations in understanding the environment, for example, a blind intersection with poor visibility [3]. Therefore, information sharing among vehicles is important for achieving sophisticated safety in addition to the self-perception of the vehicle [4–6].

In particular, in safety use cases, dynamic information, such as vehicle position and velocity, must be transmitted and processed at low latency [7]. The combination of multi-access edge computing (MEC) and cellular vehicle-to-everything (C-V2X) is considered a method to realize real-time requirements [8]. C-V2X using MEC is a system in which vehicles exchange information through a cellular system. A typical application of C-V2X using MEC is collision detection, in which vehicles send information about their position and velocity to an edge server located at the edge of the cellular system [9]. The edge server analyzes the information from many vehicles, calculates the crash probability of vehicles, and then sends the probability back to the vehicles if necessary [10]. The problem with C-V2X using MEC is the increase in processing delay at the edge server owing to limited computing resources [11]. In Ref. [12], the collision detection algorithm running on an edge server was proposed based on a linear prediction of vehicle trajectories [12]. That is, the edge server detects collisions between vehicles, assuming that vehicles are moving at the same velocity within the prediction horizon. Although the computational complexity of the algorithm is $O(n)$, where n is the number of vehicles that the edge server handles, the behavior of actual vehicles is not always linear owing to their acceleration and braking by human drivers or by unmanned/self-perceiving vehicles. To realize collision detection considering the behavior of actual vehicles, it is necessary to predict the future position of a vehicle based on its acceleration and braking tendencies. As the position of vehicles differs according to various factors, such as the degree of acceleration and/or road geometry, we take a probabilistic approach based on the Markov chain to predict the position of an actual vehicle. In this approach, the probability distribution of vehicle positions is calculated based on a model that represents how a vehicle changes acceleration and braking and is useful because the crash probability is easily calculated based on the probability distribution of vehicle positions.

In this paper, we developed a probabilistic collision detection method with a low processing delay for an edge computing environment with a cellular system. For this purpose, we modified the existing probabilistic collision detection method proposed in Ref. [13]. The existing method was developed for autonomous vehicles, which always have a planned trajectory, to detect other vehicles on the planned trajectory. In our scenario, collision detection is not executed on an autonomous vehicle but on an edge server. This discretization level causes a problem in that the execution time of the existing method is too long to detect collisions between vehicles because the number of vehicles an edge server handles is larger than that of an autonomous vehicle. To solve this problem, we propose a method for reducing the execution time of collision detection. We reduced the calculation time to predict the probability distribution of a vehicle state by utilizing the prediction error between the current state and predicted state. Then, we implement MEC-based collision detection system with private 5G and driving simulator and evaluate the collision detection method of our method.

The remainder of this paper is organized as follows: In Section 2, we describe the MEC-based collision detection application used in this study. In Section 3, we describe the existing/proposed method for predicting the probability distribution of a vehicle position. In Section 4, we describe the existing method to represent future collision risk, the problem of the method, and propose a method to solve the problem. In Section 5, we evaluate the proposed method with driving simulator and MEC environment based on private 5G. In Section 6, we present the conclusions of this paper.

2 MEC-Based Collision Detection

A MEC-based collision detection application is used to prevent collisions between vehicles at an intersection with poor visibility in the base station area. A centralized analysis of traffic on the edge server enables the detection of the potential threat of collision outside the vehicle's sensor range. Vehicles running in the base station area provide the edge server with state information via a cellular network in a constant period. The details of the application are as follows.

2.1 Vehicle-Side Behavior

Once a vehicle enters the base station area, it starts to send state information to the edge server. State information includes the position and velocity of the vehicle. The vehicle sends state information every T_s [s]. When the vehicle is on a collision course with another vehicle, the vehicle receives a warning message from the edge server and avoids collision. For the avoidance behavior, only braking is assumed for simplicity because the scope of this study is to develop a probabilistic collision detection method that has a sufficiently low latency to be compatible with a MEC-based collision detection application. A more complicated coordinated avoidance algorithm using the predicted probability distributions of vehicle positions is a future work.

2.2 Edge-Server-Side Behavior

The edge server receives state information from all vehicles in the base station area. On the edge server, collision detection is executed at every T_c [s] based on state information. The targets of collision detection are vehicles approaching each intersection within L_C [m]. The collision probability is calculated based on the state probability distribution, which is calculated using existing or proposed methods. When a collision is detected, a warning message is sent to the vehicle during the collision course.

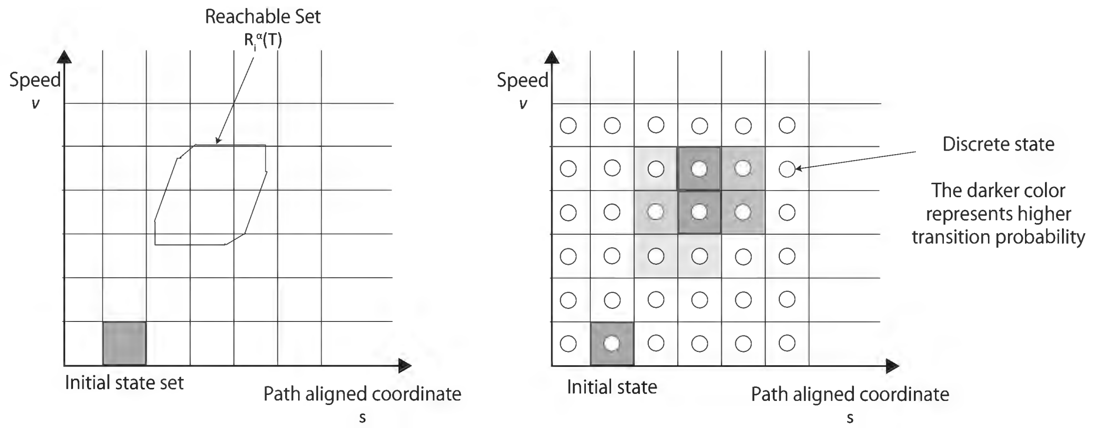


Figure 1: Definition of transition probability

3 Tracking Probabilistic Vehicle Position at MEC

3.1 Overview Of Probabilistic Prediction With MEC

The probability distributions of vehicles in the base station area are calculated and prevent vehicles from colliding with each other since a vehicle comes into the base station area until it goes out of the area. When a vehicle enters the base station area, the vehicle starts to send state information to the edge server and the probability distribution is calculated. After that, the vehicle keeps on sending the state information in a constant interval, and the probability distribution is updated with the prediction error. When the vehicle approaches an intersection in the base station area, collision detection is executed on the edge server, and the warning message is sent to the vehicle if a collision is detected.

3.2 Spatio-temporal Distributions for Probabilistic Vehicle Positions

The crash probability between vehicles is derived by the probability of a spatiotemporal overlap of vehicle positions. The probabilistic spatiotemporal distributions of vehicle positions are calculated using a Markov chain model that considers the acceleration and braking of vehicles. The probabilistic spatiotemporal distributions of vehicle positions represent stochastic reachable sets of vehicles within the prediction horizon and are useful for determining an unsafe set for each vehicle and coordinating the vehicle's behavior.

3.3 Calculation of Probabilistic Vehicle Positions with Markov-Chain

The probability distribution of a vehicle is calculated for discrete time t_k $(k = 1, 2, \cdots, f)$ where $T = t_{k+1} - t_k$ is the time step of the Markov chain, t_0 is the time when the vehicle sends information about the position and velocity of the vehicle in the path coordinate axis, and t_f is the time to predict. Note that the lateral perturbations of a vehicle are not calculated with a Markov chain but are given as a pre-defined distribution. The state of the vehicles is represented by a pair of positions and vehicles in the Markov chain, and the state vector at t_{k+1}, $p(t_{k+1})$, is

$$p(t_{k+1}) = \Gamma \cdot \Phi(T) \cdot p(t_k). \tag{1}$$

Here, Γ is the transition matrix of the acceleration input that models how the acceleration input changes. Γ is calculated for each state and models human driver or vehicle system does not change the acceleration input sharply. Input transition probability differ from a road to another because of road geometry such as curvature. Each element in the matrix represents the transition probability of the acceleration input. Φ is the state transition matrix of a vehicle that models how the position and velocity of the vehicle change. Figure 1 shows the definition of state transition probability. The transition probability between a state to another is defined as the propotion of the intersection between the reachable set of the current state and next state.

Eq. 1 is repeatedly calculated until the autonomous vehicle obtains $p(t_f)$. For collision detection, The probability distribution for the time interval $p([t_k, t_{k+1}])$ is also calculated using Eq. 2.

$$p([0, t_{k+1}]) = \Phi([0, T]) \cdot p(t_k) \tag{2}$$

Then, the probability distributions of vehicle position in the path coordinate is extracted. The probability that a vehicle is in a discrete position or path segment S_e is calculated as the summation of state probabilities whose position field is S_e using Eq. 3.

$$p_e^{path}([t_k, t_{k+1}]) := \sum_m P(s \in S_e, v \in V_m, t \in [t_k, t_{k+1}]) \tag{3}$$

Finally, the probabilistic spatiotemporal distributions of vehicle positions are obtained by integrating the distribution p^{path} calculated above and the lateral distribution p^{dev}

defined as a constant piecewise function. The probability that the vehicle center is at path segment S_e and lateral segment D_f is calculated using Eq. 4.

$$p_{ef}^{pos}([t_k, t_{k+1}]) = p_e^{path} \cdot p_f^{dev} \tag{4}$$

3.4 Updating Probabilistic Vehicle Positions with Prediction Error

The calculation of the probability distribution should be executed in a low processing delay. However, this method requires a large processing time. Therefore, we developed a method to update a probability distribution with the prediction error.

Our basic idea of modifying C-V2X is to update the state probability from the previously calculated state probability. When the existing method is applied to C-V2X, upon receiving the position and velocity of the vehicle, Eq. 1 is repeatedly calculated until the time reaches t_f. Let us assume that the edge server calculates and holds $[p(t_k), \cdots, p(t_{k+f})]$ at t_{k-1}. When the edge server receives the latest position and velocity $p'(t_k)$ at t_k, the state probability is updated as

$$p'(t_{k+m}) = p(t_{k+m}) + (\Gamma \cdot \Phi(T))^m \cdot (p'(t_k) - p(t_k)), \tag{5}$$

for m=1, …, f, and $p'(t_{k+f+1})$ is calculated as $\Gamma \cdot \Phi(T) \cdot p'(t_{k+f})$. Fig. 2 describes the compaction process in Eq. 5. The terms $p'(t_k) - p(t_k)$ in Eq. 5 represents the prediction error at t_k. Subsequently, the state probability in the future is compensated by the prediction error using Eq. 5. Because $(\Gamma \cdot \Phi(T))^m$ can be calculated in advance, the computational complexity of Eq. 5 is much lower than that of the existing method.

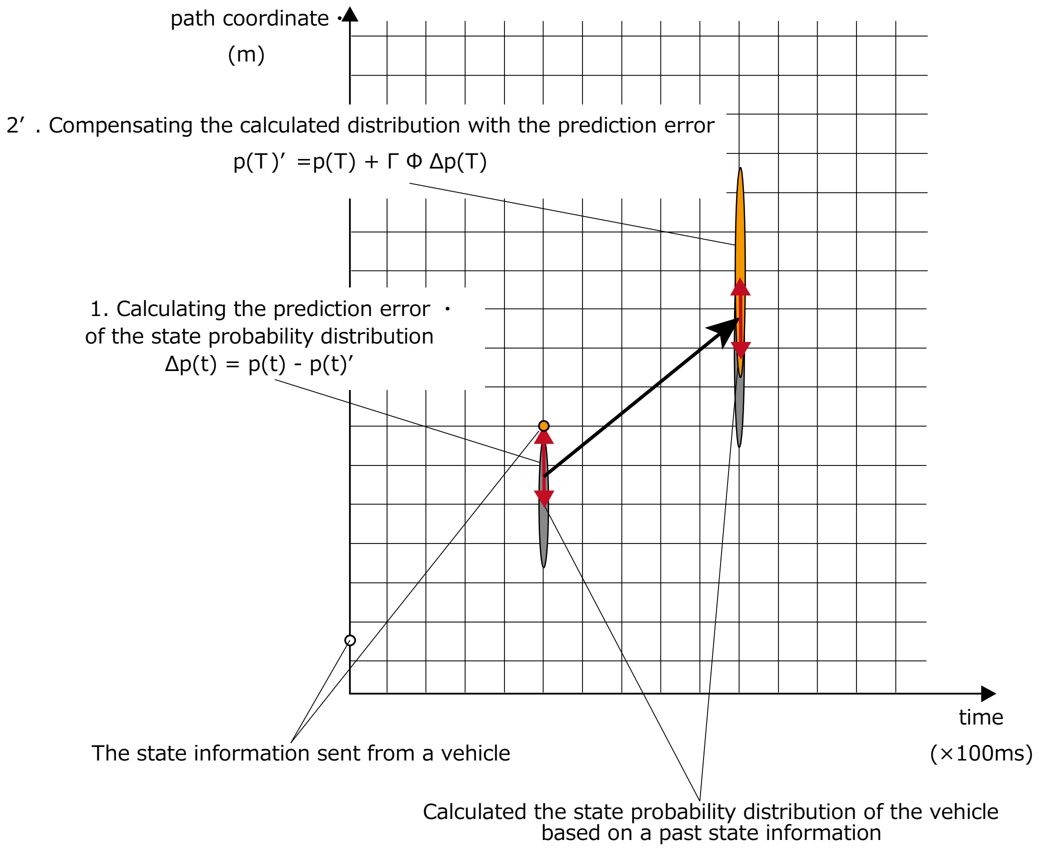

path coordinate
(m)
2′ . Compensating the calculated distribution with the prediction error
p(T)′ =p(T) + Γ Φ Δp(T)
1. Calculating the prediction error
of the state probability distribution
Δp(t) = p(t) - p(t)′
time
(×100ms)
The state information sent from a vehicle
Calculated the state probability distribution of the vehicle
based on a past state information

Figure 2: Compensatation of state probability using prediction error

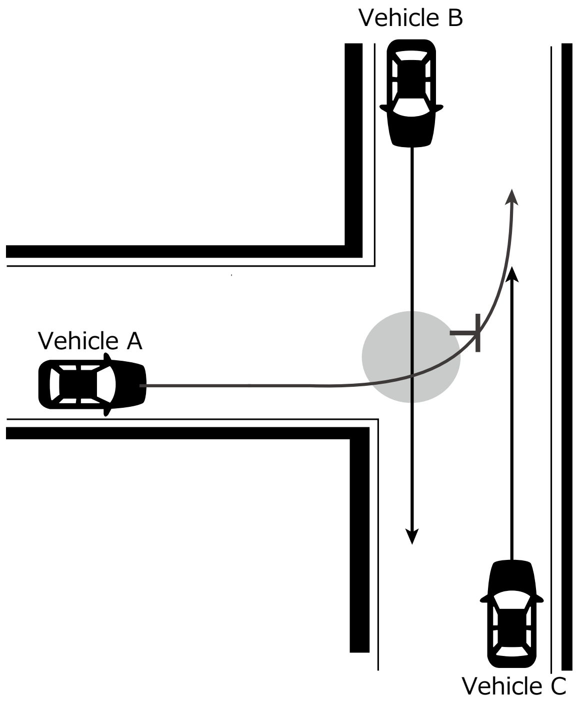


Figure 3: A scenario in our evaluation

4 Collision Detection with Probabilistic Vehicle Position

4.1 Crash Probability

The concept of the existing crash probability is a two-dimensional integral of the joint probability distributions of two vehicle positions in the range of the overlapping region. However, vehicle position distributions are calculated as discrete probability distributions; therefore, the summation of the joint probabilities is used instead of the integral.

$$p^{crash}([t_k, t_{k+1}]) = \sum_{\{e,f,g,h\}\in\Omega} \hat{p}^{pos}_{e,f}([t_k, t_{k+1}]) \cdot p^{pos}_{g,h}([t_k, t_{k+1}]) \tag{6}$$

$\hat{p}^{pos}_{e,f}([t_k, t_{k+1}])$ represents the probability that the vehicle's center is in path segment S_e and lateral segment D_f. The hat symbol is used to distinguish the vehicle from another vehicle. Ω represents the pairs of segments; that is, the two vehicles' bodies intersect each other when the vehicle center is in S_e and D_f, and the other of vehicle center is in S_g and D_h.

Table 1: The experimental environment

OS	Ubuntu 20.04 LTS
CPU	Intel(R) Core(TM) i9-11900K @ 3.50GHz
Memory	32 GB

Table 2: Discretization levels

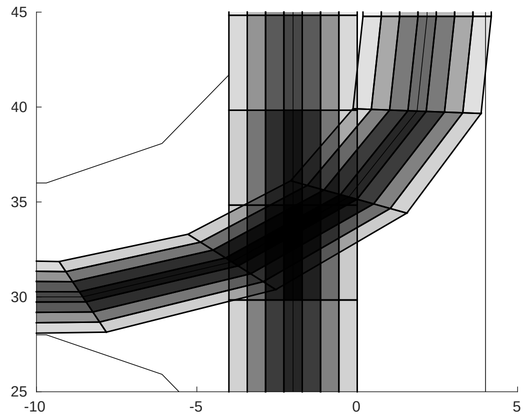
	(1)	(2)	(3)
time step [s]	0.5	0.25	0.125
path segment length [m]	5.0	2.5	1.25
lateral segment width [m]	0.57	0.28	0.14
collision time [s]	6.25	6.125	6.0625

4.2 Collision Detection using Crash Probability

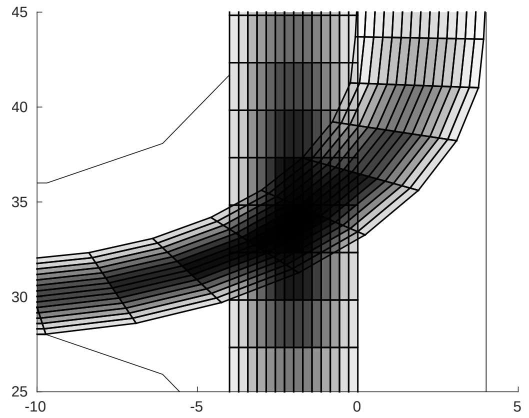
4.2.1 Tradeoff between Discretization-Level and Processing Delay

Let us consider what discretization level is the most effective for collision detection using the proposed method before the experiment to evaluate the MEC compatibility of the proposed method in terms of processing delay. Generally, the accuracy of collision detection increases as the discretization level becomes finer. However, the calculation time is also expected to increase, so it is essential to optimize the discretization level for the collision detection performance. To measure the tradeoff between the discretization level and processing delay, we executed collision detections under the three different discretization levels listed in Tab. 2. The time prediction horizon t_f is set to 10 [s]. The state information used in collision detection was generated based on two simple vehicle trajectories that collide with each other. Fig. 2 shows the scenario in our evaluation: two vehicles enter the intersection simultaneously. The trajectories represent vehicles A and B moving at a velocity of 16 [m/s]. The specifications of the PC regarded as an edge server are summarized in Tab. 1. The toolbox, COntinuous Reachability Analyzer (CORA) in MATLAB [14] was used for implementing our method.

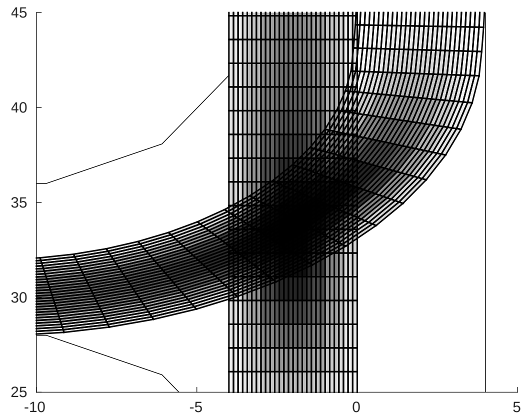
We plot the crash probabilities calculated using the state information $(x(t),\ v(t))$,



(a) Discretization level (1)



(b) Discretization level (2)



(c) Discretization level (3)

Figure 4: The probability distributions of Vehicle A and B calculated with the state information $(x(0), v(0))$

t=$\{1, 2, 3, 4, 5, 6\}$ [s] in Fig. 5. This result shows the finer the discretization level is, the larger the crash probability calculated with the same state information. This state information represents the position and velocity of the two vehicles on the collision course and is accurate for detecting collisions. Therefore, this implies that finer discretization enables the perception of collision risk more accurately.

Fine discretization contributes to the accuracy of collision detection because it enables the accurate expression of the vehicle position. Fig. 4 shows the probability distributions of Vehicles A and B calculated using state information $(x(0), v(0))$. As can be seen from the figure, the probability distribution calculated under fine discretization levels can describe accurately where the vehicles are likely to be. The crash probability calculated under

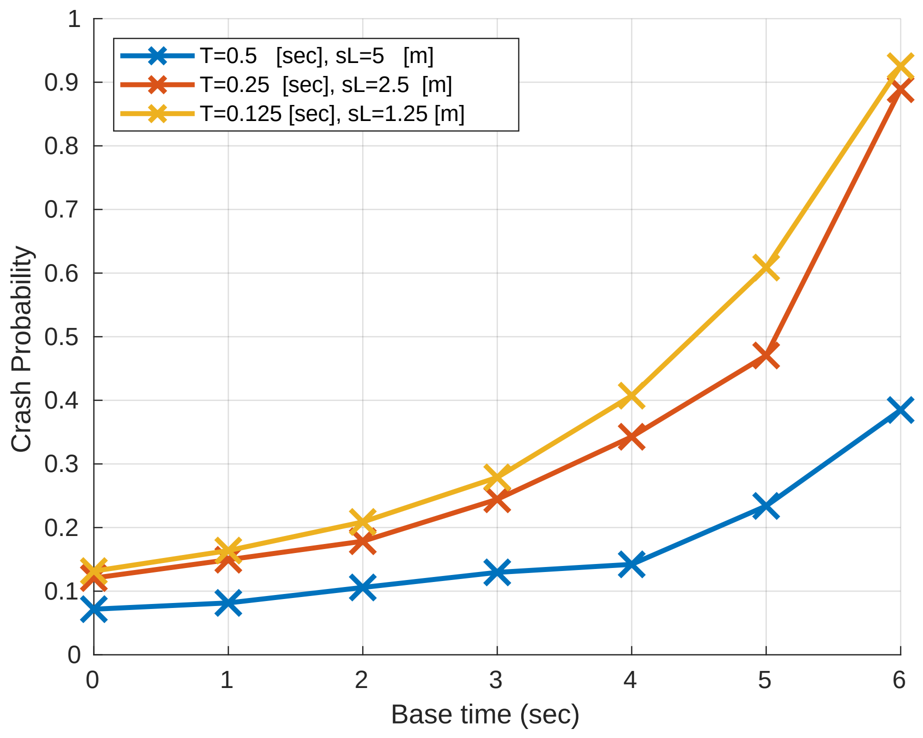


Figure 5: Comparison of the accuracy of crash probability calculated under three discretization levels: The crash probability at each time t is calculated with the state information $(x(t), v(t))$

discretization level (3) is most accurate, but the difference in crash probabilities between (2) and (3) is less than 5% at 3.5 s, the deadline considering the braking time until collision. Considering this tradeoff, discretization level (2) is expected to be the most effective for collision performance.

4.2.2 Sensitivity of Crash Probability for Collision Risk

Crash probability, the summation of joint probabilities where the segments of two vehicles' position overlap, is a way to evaluate collision risk. However, its sensitivity is low because the joint probability is the product of two probabilities, and the segments that overlap each other are a part of all segments. As shown in Fig. 5, crash probability is much lower than 100% using with the state information at 5 [s], which is only 1 [s] before the collision occurs. Therefore, we introduce another metric for capturing the collision risk in the following section.

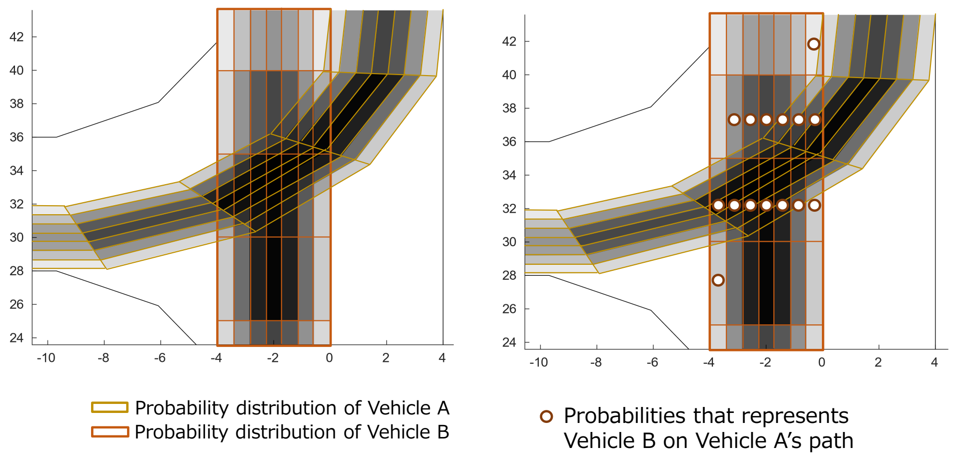


Figure 6: Collision area of Vehicle B.

4.3 Collision Detection using Overlapping Probability

In this paper, we developed a method that calculates the overlapping probability that the two vehicles exist at the same time at the place where the collision can occur, which we call the collision area.

First, we calculated the crossing probability of having one vehicle in the collision area and the other. The crossing probability was calculated as the summation of the discrete probabilities of the vehicle position in the collision area in Eq. 7.

$$p^{crossing}([t_k, t_{k+1}]) = \sum_{\{g,h\}\in E} p^{pos}_{g,h}([t_k, t_{k+1}]) \tag{7}$$

A sample calculation of crossing probability in the merging scenario is shown in Fig. 6. The figure on the left-hand side shows the probability distributions of having two vehicles. In this scenario, the collision area represents the repulsion that intersects with each other. Therefore, the crossing probability of Vehicle B is calculated as the summation of the discrete probabilities of the discrete segments indicated by the points in the figure on the right-hand side. We then calculated the overlapping probability by multiplying the crossing probabilities in Eq. 8.

$$p^{overlap}([t_k, t_{k+1}]) = \hat{p}^{cross}([t_k, t_{k+1}]) \cdot p^{cross}([t_k, t_{k+1}]) \tag{8}$$

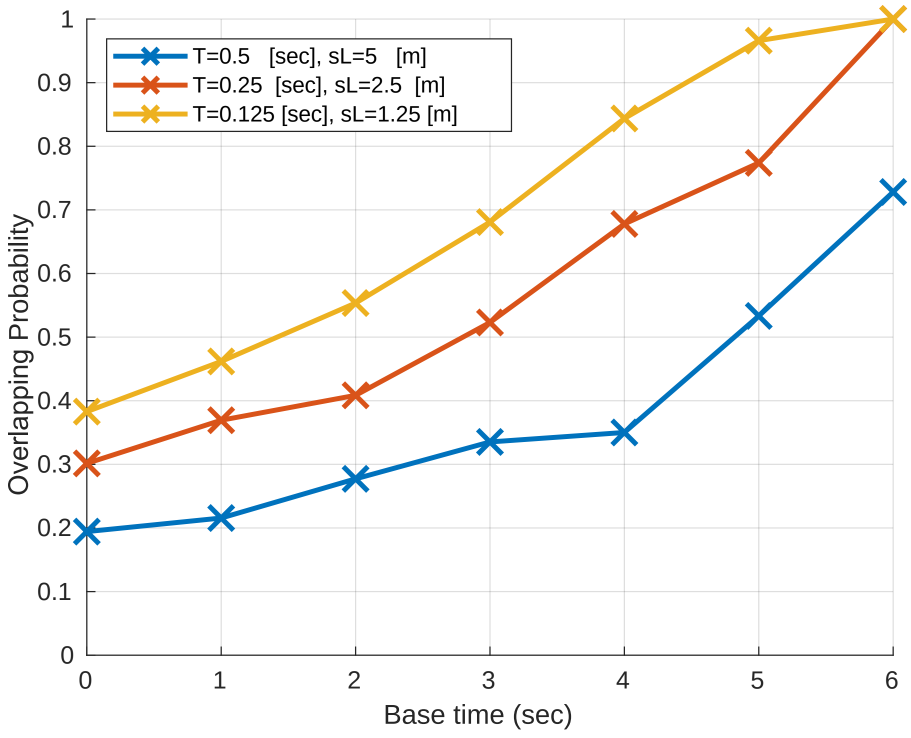


Figure 7: Comparison of the accuracy of overlapping probability

4.4 Sensitivity Of Overlapping Probability

We plot the overlapping probabilities calculated with the state information $(x(t), v(t)), t = \{1, 2, 3, 4, 5, 6\}$ [s] in Fig. 7. This result shows that the finer the discretization level, the larger the overlapping probability calculated with the same state information as the crash probability. Furthermore, the overlapping probability has a better sensitivity to collision risk than crash probability. In particular, the overlapping probability at 5 [s] is 20% greater than the crash probability at 5 [s], which implies that the sensitivity is increased by the overlapping probability.

4.5 Simple Simulation of MEC-Based Collision Detection

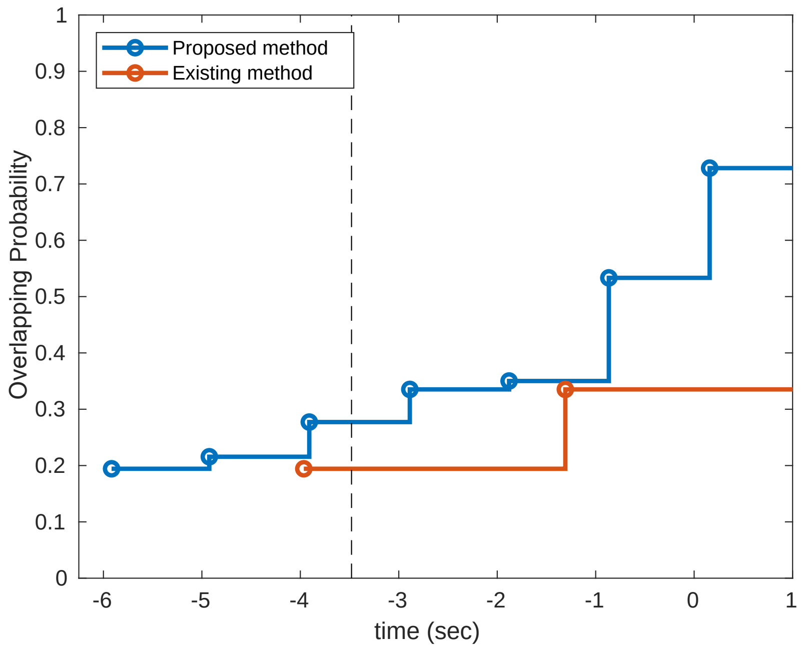
We evaluated the effect of the reduction in processing delay and the capacity of the proposed method for an MEC-based collision detection application.

4.5.1 Implementation

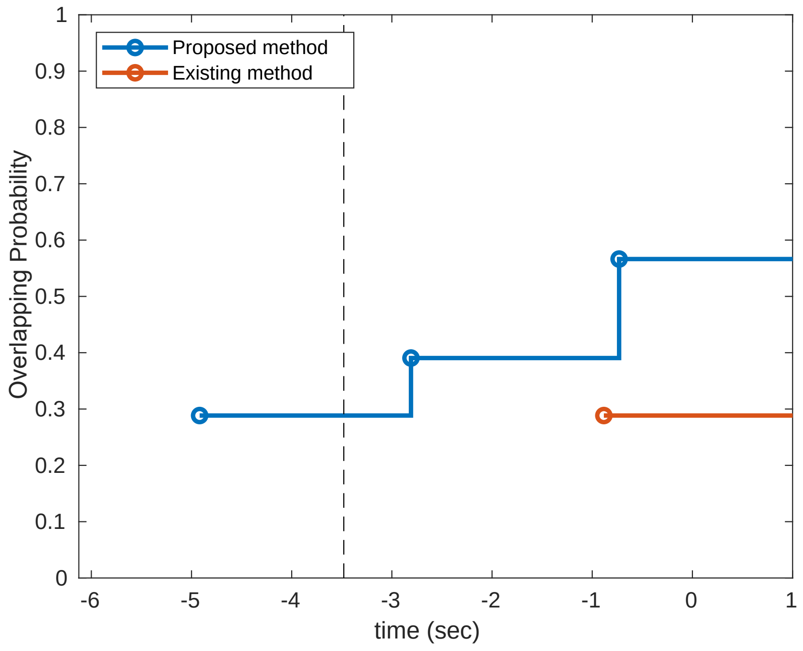
As evaluation metrics, we used the elapsed time from a vehicle starting communication with the edge server until the collision between the two vehicles is detected. To measure the elapsed time, we implemented a collision detection application that detects collisions between vehicles on a simple vehicle simulator using the scenario shown in Fig. 3. We did not use the actual cellular system, and the network latency was set to 50 [ms] in our current evaluation. The vehicle simulator simulates the simple behavior of vehicles that move at a constant speed of 16 [m/s] and collide with each other in a few seconds. Each vehicle sends the state information at $T_s = 500$ [ms]. The edge server updates the state probability using Eq. 5 and calculates the overlapping probability of all vehicle pairs every $T_c = 1000$ [ms] under discretization level (2). The state information is generated based on the trajectories of vehicles, and the trajectories are prepared such that vehicles crash 6 [s] after the beginning of the scenario.

4.5.2 Evaluation of Proposed Method

Fig. 8 shows the overlapping probability of vehicles depending on the elapsed time calculated under discretization levels (1) and (2) when the number of vehicles is 2. In these figures, the elapsed time 0 is set as the time when collision occurs between vehicles. The dashed line in the figure represents the deadline for detecting the collision to start the braking system to avoid an actual crash at time 0. The deadline is the sum of the one-way communication delay and braking time. The braking time is given by a simple physical equation, $T_d = \frac{v_{max}}{2\mu g} + T_r$ is the maximum speed of a vehicle constricted by the regulation, μ is the coefficient of dynamic friction, g is the gravitational acceleration, and T_r is the reaction time of the human driver. v_{max}, μ, g and T_r are set to 16 [m/s], 0.7, 9.8 [m/s^2], and 1 [s], respectively. Fig. 8a shows the probability transition calculated under the discretization level (1) when the number of vehicles is two. Comparing the overlapping probability calculated with the existing and proposed methods, the proposed method can detect collision risk around 2 [s] earlier than the existing method. Moreover, the overlapping probability at the deadline is 10 % higher than that of the existing method, which shows better sensitivity. Fig. 8b shows that the collision detection using the existing



(a) Discretization level (1)



(b) Discretization level (2)

Figure 8: Comparison of overlapping probability at the edge server

method missed the deadline under the discretization level (2). Note that, in the case of discretization level (3), the overlapping probability is not obtained until the deadline even with our method. When the overlapping probability was calculated using the existing method, its calculation was not completed at the deadline of collision detection because of its processing delay. The proposed method reduces the processing delay, senses the collision 4 [s] earlier than the existing method, and enables the calculation of the overlapping probability until the deadline arrives. Moreover, the calculation of overlapping probability with the proposed method meets the deadline even when the number of vehicles becomes 4 in a single thread computation, while the counterpart of the existing method is 2. This means the number of vehicles an edge server can handle is doubled by the proposed method.

Fig. 8b shows that the collision detection with the existing method finally missed the deadline. This implies that it is difficult to detect collisions in real time under the discretization level (2) because of the processing delay. Considering the real-time requirements, the collision detection performance under the discretization level (1) was the highest.

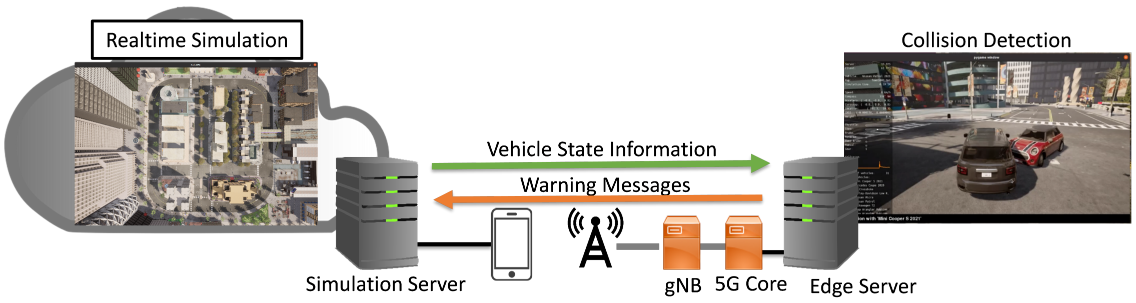


Figure 9: Overview of MEC-based collision detection in Driving Simulation

5 Evaluations of MEC-Based Collision Detection Using Driving Simulator

5.1 MEC-Based Collision Detection System for Intersection Accidents in Driving Simulator

To evaluate the collision detection performance of the proposed method for the vehicle traffic more similar to the real world, we use the driving simulator and implement the collision detection system for preventing collisions in a simulation. Figure 9 shows the overview of the system. The simulation server simulates vehicle traffic and send the vehicles' state information to the edge server. The edge server execute collision detection with the information and send warning message if any collision risk found.

5.2 Implementation

In the real situation, collision detection is performed for preventing actual vehicles from colliding with each other. In our evaluation, vehicles in the driving simulator which has communication functions are substituted for actual vehicles. The vehicles communicate with the edge server via the driving simulator communication function. The wireless network between the edge server and the mobile devices is 5G cellular network as same as actual situation.

Table 3: Driving simulation server environment

OS	Ubuntu 20.04 LTS
CPU	Intel(R) Core(TM) i9-11900K @ 3.50GHz
GPU	NVIDIA GTX 2080 @ 33MHz
Memory	96 GB

Table 4: Edge server environment

OS	Windows 10
CPU	Intel(R) Core(TM) i7-7820HQ, 2.90GHz
Memory	16 GB

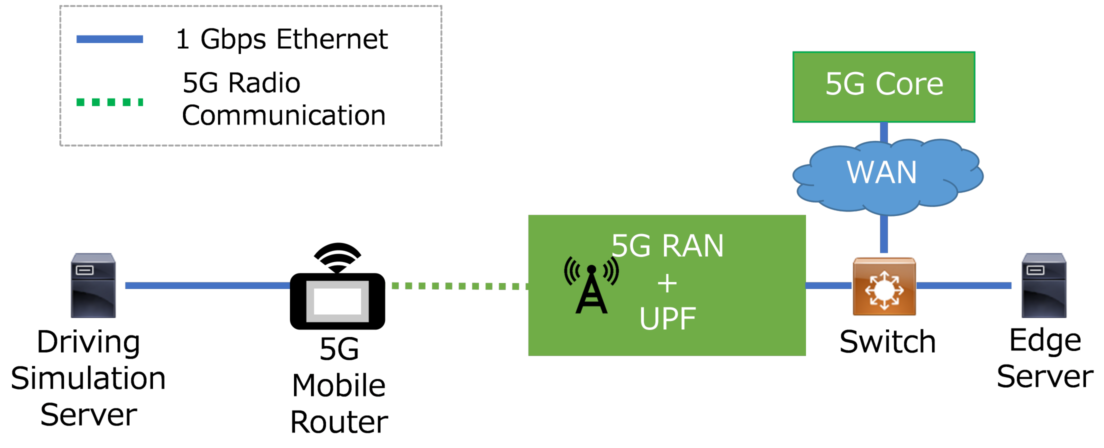


Figure 10: The network diagram of MEC system

5.2.1 MEC-Based Collision Detection Environment

Figure 10 shows the network diagram of MEC system. The driving simulation server and edge server communicate with each other via our private 5G system. The private 5G system runs in SA (Stand Alone) mode and the UPF (User Plane Function) is connected with the edger server. The driving simulation server accesses the private 5G network via 5G mobile router. Table 3 and 4 show the driving simulation server environment and the edge server environment respectively.

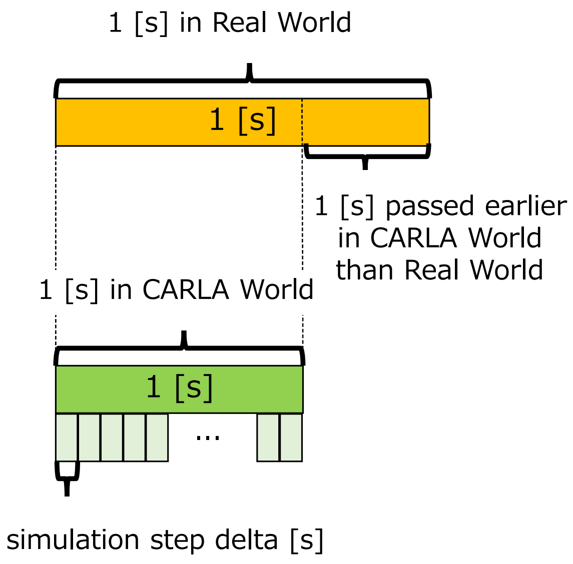

1 [s] in Real World
1 [s]
1 [s] passed earlier in CARLA World than Real World
1 [s] in CARLA World
1 [s]
...
simulation step delta [s]

Figure 11: The time difference of real time and simulation time

5.2.2 Driving Simulator: CARLA

We choose CARLA (Car Learning to Act) [15] as the simulator for the measurement of collision detection performance of our method. CARLA is an open-source driving simulator for autonomous driving research. CARLA architecture is a server-client type simulator. The server simulates CARLA world physics and actors' behavior with Unreal Engine. The client request the world's information, spawning specified actors, changing actors' behavior from the server. This feature helps us to validate our collision detection system in terms of real time collision detection and avoidance because collision detection can be executed in real time based on extracted vehicles' state information and the collision-detected vehicles' behavior can be changed in real time.

In our evaluation, Traffic Manager (TM) in CARLA is used to generate vehicle traffic. TM simulates random traffic in a town where the path of the town is closed, and the vehicles circulate the town until the simulation is ended. If a vehicle under the TM control approaches an intersection in the town, TM decides which direction for the vehicle to go randomly. Therefore, the vehicle traffic generated by TM becomes random.

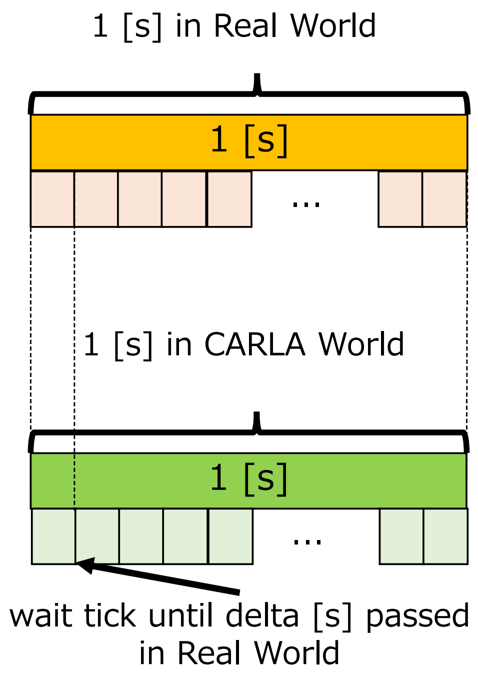


Figure 12: Adjusting simulation time to real time

5.2.3 Realtime Simulation in CARLA

CARLA is not a real time simulator, which means that the simulation time of a CARLA world is passed earlier than real time in default settings. This is because a step in a simulation proceeds earlier than time interval in a step in real time as shown in Figure 11. To realize real time simulation, the simulation is run in server-client synchronous mode and forced the simulation does not tick until the time interval in a simulation step passed in real time with the client-side Python script as shown Figure 11.

5.2.4 Communication between Vehicles in Driving Simulator and Edge Server

In actual situation, vehicles communicate with the edge server via a cellular network; vehicles send its state information to the edge server and the edge server send warning message back to the vehicles. In our implementation, the driving simulator and the communication modules is substituted to actual vehicles. The communication modules consists of two modules, state information sender and warning message handler implemented with Python. The data flow among modules is described in Figure 13. The state information

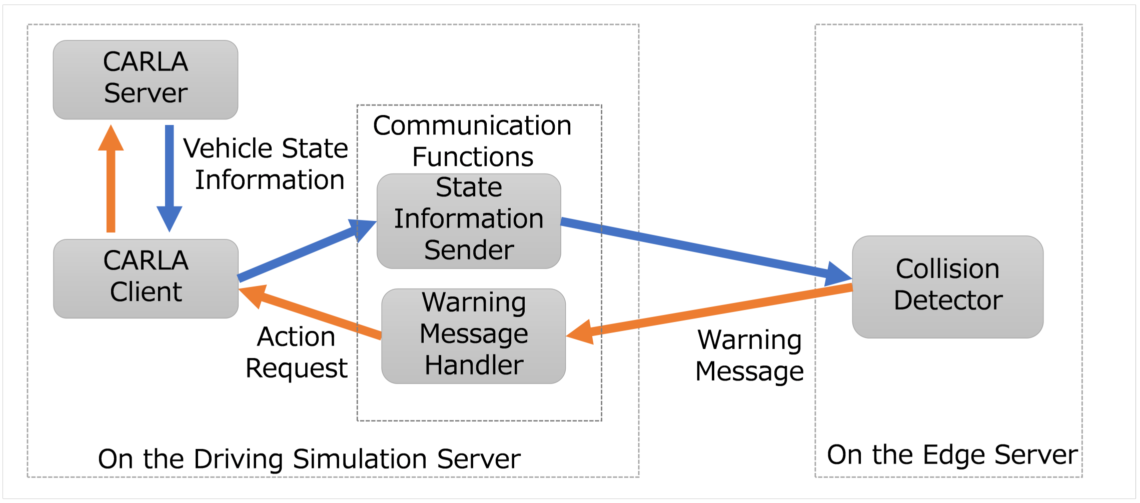


Figure 13: The data flow between driving simulator and edge server

sender sends the state information of all the vehicle in the simulation in a constant period T_s. The state information sender generates state information with the extracted velocity and position of a vehicle from the CARLA server via CARLA client . The warning message handler receives warning messages from the collision detector and make the warned vehicles take some actions such as braking. The action is defined beforehand and the warning message handler make the vehicle out of TM's control and request the action to the CARLA server via the carla client. After the collision risk becomes low, the warning message handler receives no warning message for the vehicle so set the vehicle under the TM's control. The information/action request in the both communication modules is performed with Python API of CARLA. The communication modules run in a thread for a module, so state information and warning messages are sent and received concurrently.

5.3 Evaluation of Proposed method

5.3.1 Evaluation Metrics

As evaluation metrics of collision detection performance, we use the overlapping probability and the distance d [m] between the vehicles on collision course at the timing of a vehicle approaching the intersection within L [m]. The distance d [m] represents the difference of the timing entering the intersection; if the vehicle enters the intersection at the same timing, the distance becomes about $\sqrt{2}(L+l)$ [m], l [m] is the distance between the start

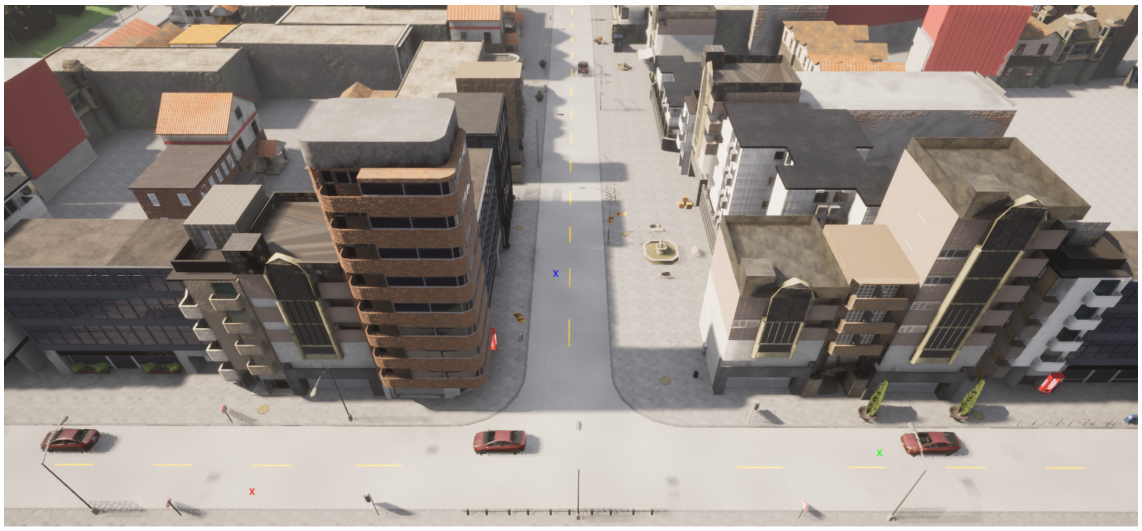


Figure 14: T-intersection used for our evaluation

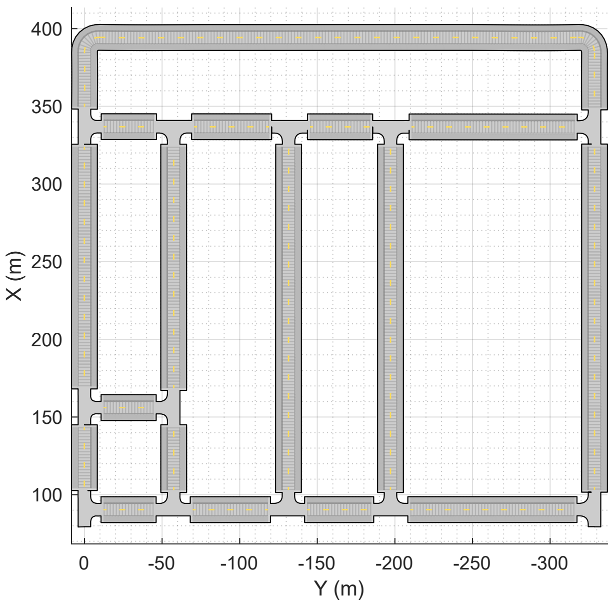


Figure 15: The preset map of CARLA used for our evaluation

point of intersection and the intersecting point of the two vehicles' trajectories. Therefore, when d [m] is nearer to L [m], the collision risk is desirable to be evaluated higher because the timing of entering the intersection is the same.

5.3.2 Evaluation Scenario

As evaluation scenario, we assume the collision detection for an intersection with bad visibility. In this scenario, collision detection is performed for a collision between a merging vehicle and a vehicle running in another lane within 120 [m] from the intersection. The collision detection period in the edge server, T_c is set to 0.5 [s], and time step of Markov chain T to 0.5 [s]. For this scenario, the preset map of CARLA which includes a T-intersection is used. The map and the intersection shown in Figure 14 and 15.

The traffic density k is set to 7 [vehicles/km]. CARLA has no function to set traffic density, so traffic density is set by spawning as many vehicles as the number of vehicles when the traffic density is k in the map where the total length of the all lane 6.4 [km]. This is valid because all the vehicles keep on running in the closed map and the number of vehicles does not change through a simulation.

The speed restriction of the road is set to 60 [km/h], the general speed regulation of Japanese road. The distance L [m] is set to 15 [m], the distance which takes for a vehicle running at 60 [km/h] to slow down to yield speed 20 [km/h]. This is calculated with a simple physics equation $v^2 - v_0^2 = 2ax$ for a moving object, v_0 is the first velocity, v is the changed velocity, a is the acceleration, x is the distance which takes the object change velocity v_0 to v. A straight vehicle at yield speed can stop in 1 [m] and avoid collision with a merging vehicle even if it recognize the merging vehicle after entering the intersection. For the same reason, the target of collision detection is limited for vehicles whose speed is over yield speed. Moreover, overlapping probability and distance between vehicles at over yield speed is recorded for evaluation.

5.3.3 Evaluation Result

Figure 16 and 17 shows the overlapping probabilities and the distance d [m] between a merging vehicle and a straight forward vehicle comming from right side of Figure 15 at the timing of a merging vehicle approaching the intersection within L [m]. In this case, l is 8.5 [m] so $\sqrt{2}(L + l)$ is 33.2 [m]. Figure 16 repsresents when the proposed method is applied, Figure 17 shows when the proposed method is apllied.

The blue plot represents the overlapping probability for the distance d received from

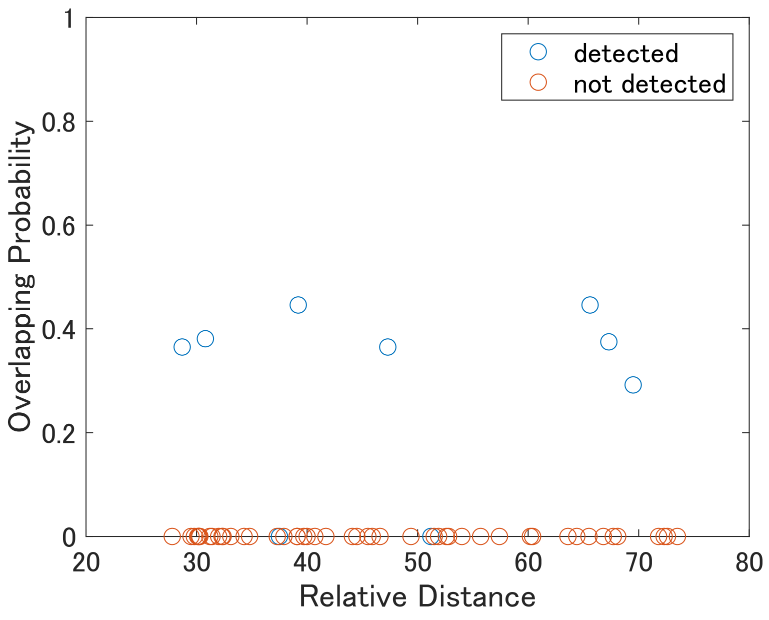


Figure 16: The relationship between the distance and overlapping probability when the existing method is applied

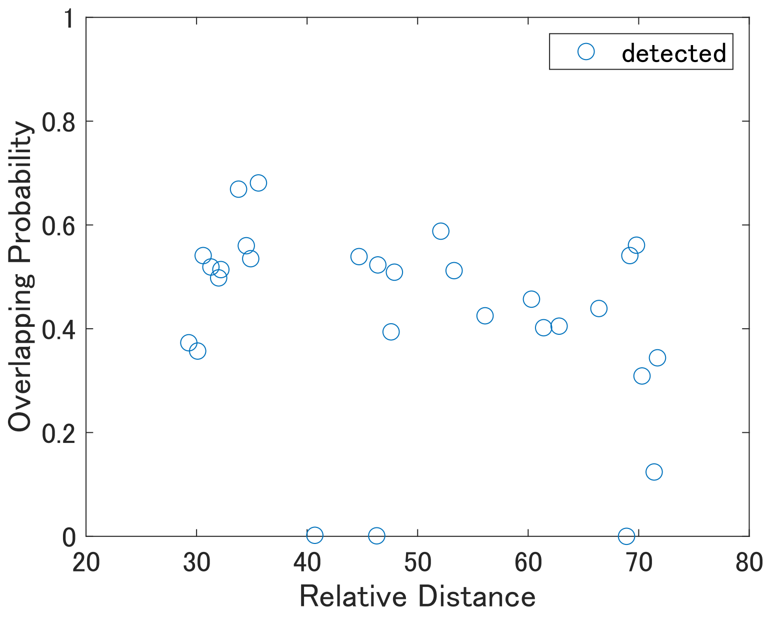


Figure 17: The relationship between the distance and overlapping probability when the proposed method is applied

the edge server, and the red plot represents the case when the overlapping probability is not received from the edge server at the distance L [m] from the intersection. The figures show our proposed method notifies all of the potential collision risks beforehand which is enough to stop the vehicles, whereas the existing method fails to notify more than 80% of the risks. With the proposed method, there are no samples whose d is around 33.2 [m] which implies that merging vehicles receive the warning message when there is approaching vehicles in another lane within 120 [m] over yield speed.

Note that there are two points of blue plots locate around 0 even though around 33.2 [m] even with the proposed method. These plots represents that the vehicles braked more than L [m] away from the intersection for some reasons such as avoiding collision with a leading vehicle and the overlapping probability is calculated lower. Moreover, overlapping probability is higher as d gets close to 33.2 [m], which means our method can evaluate collision risk higher if the time difference of vehicles entering the intersection is smaller.

6 Conclusion

In this paper, we developed a probabilistic collision detection method with lower processing delay than the existing method for an edge computing environment. Our method reduces the processing delay of the probabilistic collision detection method by utilizing the prediction error. We evaluate the effect of our method with a simple vehicle simulator with an intersection scenario. The results show that our method reduces the calculation time by 85%, which enables to detect a collision 3.5 seconds before the crash with a sufficient probability.

Moreover, we implement MEC-based collision detection system for intersection accidents in the driving simulator using private 5G, and evaluate the collision detection method of our method in the situation similar to the situation in real world. Then, we evaluate the collision detection performance in terms of the approaching distance and overlapping probability received by the vehicles on collision course. With the proposed method, all of the collision risk is received by vehicles before they enter the intersection, while not received with the exisiting method. In addition, the proposed method vehicles to receive the collision risk corresponding to the time difference of entering the intersection.

In this paper, the probability distribution is used only to assess the collision risk. Our future work is to coordinate the vehicle traffic with probability distributions of vehicles' state in centralized way, and assign the celular resources for an urgent communication from MEC server to vehicles.

Acknowledgments

As an acknowledgment of my gratitude, I would like to express my gratitude who support this master thesis. I would like to express the deepest appreciation to Associate Professor Shin'ichi Arakawa of Osaka University Without his guidance about probability theory and persistent help this thesis would not have been possible. My deepest appreciation goes to Professor Masayuki Murata of Osaka University for His insightfull and kindfull advice. I could be aware of problems in my research and the solution with his advice. I would like to express my gratitude to gives me constructive comments for Associate Professor Yuichi Ohsita and Assistant Professor Daichi Kominami of Osaka University. I would like to thank all of the member of Advanced Network Architecture Research Laboratory. The end of my acknowledgement, I thank my parents for supporting my master thesis.

References

[1] "Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems (J3016_201806)," https://www.sae.org/standards/content/j3016_201806/, last accessed 2021/12/11.

[2] D. Feng, C. Haase-Schütz, L. Rosenbaum, H. Hertlein, C. Gläser, F. Timm, W. Wiesbeck, and K. Dietmayer, "Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1341–1360, Feb. 2021.

[3] S.-W. Ko, H. Chae, K. Han, S. Lee, D.-W. Seo, and K. Huang, "V2X-based vehicular positioning: Opportunities, challenges, and future directions," *IEEE Wireless Communications*, vol. 28, no. 2, pp. 144–151, Apr. 2021.

[4] H. Bagheri, M. Noor-A-Rahim, Z. Liu, H. Lee, D. Pesch, K. Moessner, and P. Xiao, "5G NR-V2X: Toward connected and cooperative autonomous driving," *IEEE Communications Standards Magazine*, vol. 5, no. 1, pp. 48–54, Mar. 2021.

[5] H. Ma, S. Li, E. Zhang, Z. Lv, J. Hu, and X. Wei, "Cooperative autonomous driving oriented mec-aided 5G-V2X: Prototype system design, field tests and ai-based optimization tools," *IEEE Access*, vol. 8, pp. 54 288–54 302, Mar. 2020.

[6] R. Ravindran, M. J. Santora, and M. M. Jamali, "Multi-object detection and tracking, based on dnn, for autonomous vehicles: A review," *IEEE Sensors Journal*, vol. 21, no. 5, pp. 5668–5677, Dec. 2021.

[7] T.-K. Lee, J.-J. Chen, Y.-C. Tseng, and C.-K. Lin, "Effect of packet loss and delay on V2X data fusion," in *Proceedings of 2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pp. 302–305, Sep. 2020.

[8] M. Emara, M. C. Filippou, and D. Sabella, "MEC-enhanced information freshness for safety-critical C-V2X communications," in *Proceedings of 2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, pp. 1–5, Jun. 2020.

[9] M. Boban, A. Kousaridas, K. Manolakis, J. Eichinger, and W. Xu, “Connected roads of the future: Use cases, requirements, and design considerations for vehicle-to-everything communications,” *IEEE Vehicular Technology Magazine*, vol. 13, no. 3, pp. 110–123, Sep. 2018.

[10] G. Avino, P. Bande, P. Pantelis A Frangoudis, C. Vitale, C. Casetti, C. Fabiana Chiasserini, K. Gebru, A. Ksentini, and G. Zennaro, “A MEC-based extended virtual sensing for automotive services,” *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1450–1463, Dec. 2019.

[11] A. Moubayed, A. Shami, P. Heidari, A. Larabi, and R. Brunner, “Edge-enabled V2X service placement for intelligent transportation systems,” *IEEE Transactions on Mobile Computing*, vol. 20, no. 4, pp. 1380–1392, Apr. 2021.

[12] M. Malinverno, G. Avino, C. Casetti, C. F. Chiasserini, F. Malandrino, and S. Scarpina, “Performance analysis of C-V2I-based automotive collision avoidance,” in *Proceedings of 2018 IEEE 19th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pp. 1–9, Jun. 2018.

[13] M. Althoff, O. Stursberg, and M. Buss, “Model-based probabilistic collision detection in autonomous driving,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 2, pp. 299–310, Apr. 2009.

[14] M. Althoff, “CORA,” https://github.com/TUMcps/CORA, 2022.

[15] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “CARLA: An open urban driving simulator,” in *Proceedings of the 1st Annual Conference on Robot Learning*, pp. 1–16, Nov. 2017.