Development of collision avoidance system for multiple autonomous mobile robots

Yew Cheong Hou¹, Khairul Salleh Mohamed Sahari¹, Leong Yeng Weng¹, Hong Kah Foo², Nur Aira Abd Rahman², Nurul Anis Atikah² and Raad Z Homod³

Abstract
This article presents a collision avoidance system for multiple robots based on the current autonomous car collision avoidance system. The purpose of the system is to improve the current autonomous car collision avoidance system by including data input of other vehicles’ velocity and positioning via vehicle-to-vehicle communication into the current autonomous car collision avoidance system. There are two TurtleBots used in experimental testing. TurtleBot is used as the robot agent while Google Lightweight Communication and Marshalling is used for inter-robot communication. Additionally, Gazebo software is used to run the simulation. There are two types of collision avoidance system algorithm (collision avoidance system without inter-robot communication and collision avoidance system with inter-robot communication) that are developed and tested in two main road crash scenarios, rear end collision scenario and junction crossing intersection collision scenario. Both algorithms are tested and run both in simulation and experiment setup, each with 10 repetitions for Lead TurtleBot sudden stop, Lead TurtleBot decelerate, Lead TurtleBot slower speed, and straight crossing path conditions. Simulation and experimental results data for each algorithm are recorded and tabulated. A comprehensive comparison of performance between the proposed algorithms is analyzed. The results showed that the proposed system is able to prevent collision between vehicles with an acceptable success rate.

Keywords
Internet of Things, collision avoidance system, mobile robot, intelligent transportation system

Date received: 12 September 2019; accepted: 15 March 2020

Introduction
Integrating computational technology into a vehicle was not uncommon these days. The massive diffusion of computational embedded systems in the last decade had contributed to the creation of networks called Internet of Things. This technology concept already started to be implemented in the automotive sector. A premier example of this concept was the self-driving car or more commonly known as autonomous car technology, noticeably being developed by world’s largest and leading companies such as...
as Uber, Google, and Tesla Motor. Currently, Tesla Motor dominates the autonomous car market in the world with its advanced Tesla Autopilot technology. Tesla Autopilot enables the vehicle to be fully self-driven without requiring much human driver input, which means the car will drive by itself by relying on its advanced car sensors such as ultrasonic sensors, radar, and Tesla vision cameras controlled by the central processing unit installed on the vehicle. Tesla Autopilot managed to integrate its autopilot feature with collision prevention with the aim to improve its autonomous car safety. Besides, other car manufacturers also started to develop their own autonomous cars with the collision avoidance system.

The most popular types of collision avoidance systems were mostly relying on the sensors via camera, laser, and radar to detect the obstacles on the road and onboard computer system to navigate the vehicles. However, the technology suffered a setback when the first casualty caused by Tesla Autopilot was reported in the United States on May 7, 2016. Preliminary investigation shows that the sensors and the camera system of the vehicles failed to detect a moving obstacle in front which caused the failure of the system to execute any collision avoidance maneuver such as emergency braking or evasion steering turns causing the accident. A better and comprehensive collision avoidance system requires a continuous improvement cycle to ensure autonomous car safety.

One of the promising solutions to improve the collision avoidance system is by integrating a two-way communication system between autonomous cars. The possibility of collision between autonomous cars and others is substantial with the lack of a two-way communication system or vehicle-to-vehicle (V2V) communication, together with the possibility of navigation-aided equipment error due to environment or external factors. By integrating the V2V communication technology into the current autonomous car, a better and safer collision prevention system or even an automated collision avoidance system can be developed. Autonomous cars will be able to obtain a detailed picture of the unfolding situations around them, revealing and preventing errors that even the sensor system alone would fail to detect and anticipate.

Current autonomous vehicles are solely dependent on sensors and cameras as input to maneuver and navigate the vehicle. Sensors and cameras used by the current autonomous vehicle as input for the navigation are proven to be not secure enough as both can be affected by weather and other external factors. Heavy rain can interfere with the sensors while snow or heavy fog will block the camera view, thus disrupt and cause the error to the image processing. Probability for an autonomous vehicle to involve in collision or accident with other vehicles is there when the sensors and cameras of the vehicle fail to detect other vehicles around it. With a two-way communication system, potential accidents can be mitigated since the vehicles will be able to estimate the other vehicles around them based on the current position, especially when the sensors and cameras fail to detect the obstacles.

In this research, the implementation of the V2V technology concept into the current autonomous car technology is investigated and researched. An algorithm for the collision avoidance system based on the combination of V2V technology and current autonomous car technology is developed in this work. The research will be carried via simulation and real-world experiments using TurtleBot. The objectives of the research will focus on:

- design and development of a collision prevention system by integrating two-way communication technology with current autonomous vehicle technology including sensors and cameras on mobile robots;
- development of an algorithm for the collision prevention system for multiple mobile robots; and
- testing of the collision prevention system algorithm and validating it using TurtleBot in Gazebo software simulation and real-world experiments, respectively.

The content of the article is organized as follows. The “Related work” section discusses the literature review that has been conducted for the collision avoidance algorithm of the autonomous car. The “Research methodology” section describes the approaches used throughout in this research work. The “Result and discussion” section presents a comprehensive analysis of the results obtained from the simulation and experiment conducted using the proposed algorithms. The “Conclusion” section finalizes the current work by providing recommendations for future work.

Related work

A comprehensive review focusing on the current autonomous technology, sensors used in autonomous vehicles, the V2V communication technology concept, and current collision avoidance algorithms available in Robot Operating System (ROS) is reported in this section.

Autonomous vehicles technology

An autonomous car is a self-driving car that takes over the human task to control a car and to relieve the driver from the task of driving. It enables elements within the transportation system to become intelligent by embedding them with smart sensing devices and empowering them to communicate with each other through wireless technologies. According to the definition, intelligent transportation systems are those utilizing synergistic technologies and systems engineering concepts to develop and improve transportation systems from different aspects. An autonomous car provides more convenience and, safety and less energy-intensive to the driver. Current autonomous cars such as Tesla Model 3 and Google Self-Driving Car provide the functionality of “fully self-driven,” meaning the car will drive by itself without requiring much driver input.
by relying on its advanced car sensor and computer vision camera. Fully autonomous car developments are still in its pullulating stages. Currently, there are several research and development divisions of automobile companies such as Audi, Toyota, Ford, Volkswagen, Mercedes, BMW, Volvo, Tesla, and Google still developing their own autonomous car. The notion of realizing autonomous vehicles or self-driving cars can be achieved by designing an efficient advanced driver assistance system.

**Sensors of autonomous vehicles**

The convergence of sensor-based technologies is extensively used in autonomous vehicles. Sensor-based systems offer varying degrees of assistance to the driver. The fusion of available sensors and artificial intelligence is used to assist the autonomous system to understand the vehicle’s surroundings. The fundamental sensors normally used by autonomous vehicles are radar, light detection and ranging (LIDAR) or laser, camera, and so on. Radar is the most commonly used equipment in autonomous vehicles. It is mainly used for motion measurement by using radio waves to determine the velocity, range, and angle of objects. Radar is computationally lighter than a camera and uses far lesser data than LIDAR. While less angularly accurate than LIDAR, radar is capable to work in every condition and can even use reflection to see behind obstacles. LIDAR is favorably used for 3D mapping. LIDAR is a technology that measures distance using laser light. The technology can scan more than 100 m in all directions (360° imaging), generating a precise 3D map of the car’s surroundings that are very vital for autonomous cars to make intelligent decisions. The disadvantages of LIDAR are its high cost to manufacture and the large amount of data that it generates while in operation. Infrared cameras use infrared radiation to form an image. Infrared camera is popular for its capability to capture image or video under low light conditions. Infrared camera is used for autonomous car navigation assistance during night or dark situations. Computer vision by far is the cheapest and most readily available sensor. Computer vision uses massive amounts of data during operation, creating millions of pixels or megabytes at every frame, thus making the processing computation intense and algorithmically complex. The main advantages of computer vision are its capability to see real image and color, making it the best for scene interpretation, in comparison with LIDAR and radar.

**V2V communication technology**

Proposed V2V system or “two-way communication system” will enable each car on the road to “communicate” with each other and broadcast their position, speed, steering wheel position, brake status, and other related data to other vehicles within connection range by using dedicated short-range communication (DSRC) systems that are capable of transmitting information including positions in every 100 ms, in accordance to Vehicle Safety Communications Consortium of Vehicle Infrastructure Integration requirement. This system also enables cars to establish ad hoc networks to cooperate, by providing the information about traffic, allowing sensing and broadcasting of security warnings, such as accidents and road issues. DSRC system is another protocol for wireless communication different from Wi-Fi which is intended for wireless communication between vehicles. The computers aboard each car process various readings being broadcasted by other vehicles multiple times for every second, each time calculating the chance of an impending collision. Transmitters use a dedicated portion of the wireless spectrum as well as a new wireless standard, 802.11p, to authenticate each message.

**Collision avoidance algorithms in ROS**

The ROS is considered as a flexible framework for writing robotic application software. ROS is an open-source toolkit with the aim to prevent robot developers to start from scratch when developing their own robot. ROS enables the programming code to be reused in robotics research and development that has been done by previous researchers. ROS provides many useful tools, hardware abstraction, and message-passing system between nodes. Nodes run independently and communicate with others via topics using one-to-many subscriber model and the TCP/IP protocol. Packages developed include features such as low-level driver for sensors, universal transport layer and ready-to-use implementation of the latest state-of-the-art algorithm for data fusion, 2D and 3D mapping, path planning, and autonomous driving system. One example of the application of ROS in this research is the 3D mapping framework. In this research, ROS will be used in programming and running the TurtleBot using available packages such as the navigation stack and mapping stack. Google Lightweight Communication and Marshalling (LCM) is a set of tools or libraries for information exchange and data marshaling for various programming languages. There are several collision avoidance system packages available in the ROS library including optimal reciprocal collision avoidance system (ORCA), ClearPath algorithm, and multi-robot collision avoidance with localization uncertainty system (ORCA). ORCA was introduced by Van Den Berg et al. Using the ORCA system, robot agents independently computed half-planes of collision-free velocities for each other agent. Each robot will take half of the responsibility of avoiding pairwise collision to ensure collision-free motion. This results in smoother motion of the robot when avoiding collision between them. ClearPath algorithms are developed and introduced by Guy et al. with the capability to efficiently compute collision-free velocities. The algorithm is suitable for different variations of velocity obstacles (velocity obstacles, reciprocal velocity obstacles, or
hybrid reciprocal velocity obstacles) used in ROS. ClearPath follows the concept that the collision-free velocity is defined as the closest to the preferred velocity on the intersection of two-line segments for any two-velocity obstacle, or the projection of the preferred velocity onto the closest leg for each velocity obstacle. Multi-robot CALU was introduced by Claes et al. This system uses ORCA, and the combination with the extension of non-holonomic robots (NH-ORCA) and adaptive Monte Carlo localization (AMCL). This system computes collision avoidance action independently for each robot (decentralized system) based on sensors’ information and data exchange between each robot.

**Research methodology**

The methodology adopted for this work is explained in this section. To enable the simulation and experiment to be conducted, a meticulous combination of hardware and software is required. Figure 1 shows the block diagram of the system architecture that has been used in this research. Hardware used in this work include TurtleBot, Kinect sensor, Hokuyo laser range finder, Raspberry Pi/Laptop, and a Wi-Fi router. All of the hardware were assembled and connected to the TurtleBot. Most of the system connections are powered by the TurtleBot battery, except for the laptop computer.

The algorithm was loaded into the Raspberry Pi/Laptop to program the TurtleBot for the autonomous navigation system. The Raspberry Pi/Laptop acts as the central processing unit of the systems connected to the TurtleBot responsible of running the navigation system and also the collision avoidance system. The laptop computer was connected to the Wi-Fi router wirelessly and acts as the secondary data input for TurtleBot navigation and movement system. The Wi-Fi router allows TurtleBot to communicate with another TurtleBot using Google LCM, which is similar to the DSRC system. This system enables the TurtleBot to exchange data with each other. Kinect sensors and Hokuyo laser range finders were acting as the primary data input to determine the navigation and movement of the TurtleBot. Kinect sensors consist of a color camera and depth camera that is capable of capturing image and recording video in color format, plus an infrared (IR) emitter and an IR depth sensor that can measure the distance between the object and the sensor via reflection of the IR light beams. For software, ROS in Python and C are used in this research. The setup was done to replicate the autonomous car set up with V2V communication. The research work started with developing the algorithm which was tested through simulation. Later, the developed algorithms were tested and validated through field experiments.

**Proposed system algorithms**

In this research, two main algorithms were developed namely collision avoidance system without inter-robot communication (CASA-1) and collision avoidance system with inter-robot communication (CASA-2). CASA-1 represents the current autonomous collision avoidance system (without peer-to-peer communication system) where only input from sensors is considered for its collision avoidance system. CASA-2 represents the proposed collision avoidance system algorithms with peer-to-peer communication where the input from sensors and also data exchanged between robots using inter-robot communication containing velocity and position are considered for its collision avoidance system.

During the inter-robot communication phase, each robot broadcasts its velocity and position information in the global coordinate frame on the common ROS topic through wireless networks and also subscribes to the same topic and caches velocity and position data of other robots. The robot positions were determined based on forwarding integrated in time according to the motion model of robots via the last known position and velocity information. The inter-robot communication system is based on the DSRC using Google LCM. Figure 2 shows the flowchart of the proposed collision avoidance system. Two main inputs were considered in this system, the sensors input and data from the LCM system for the collision avoidance system. The system ends when there is no possible collision occurrence and the TurtleBot reaches the navigation goal.

**Experimental setup**

Two sets of TurtleBot were placed on a track with each connected with the Kinect sensor and Hokuyo placed on the TurtleBot facing front and placed in a line as shown in Figure 3. Wi-Fi router was placed on top of one of the TurtleBot that was connected to the central processing unit. Both of the TurtleBots were placed with a certain distance between each other. The initial position and distance of the TurtleBot are calculated using rear end collision scenario (RECS) setup and junction crossing intersection collision scenario (JCICS) setup. This setup was designed to replicate the distance of autonomous cars in real-world situations. Experimental setup is explained as below:
Rear end collision scenario

Figure 4 shows the experimental setup of the research, namely RECS. For the simulation and experiment setup, equation (1) was used to determine the minimum distance needed between two TurtleBots:

\[ D = \frac{1}{2} at^2 + \dot{D}t \]  

(1)

where \( t \) represents time to collision, \( D \) represents range between two TurtleBots, \( \dot{D} \) represents Lead TurtleBot velocity minus Follower TurtleBot velocity, and \( a \) represents the acceleration of Lead TurtleBot.

Similarly, equation (2) can be used to find the minimum distance between two TurtleBots for the Lead vehicle stopped scenario:

\[ t = \frac{L_o}{V_{o1}} \]  

(2)

For this scenario, assuming 2 s of time before the collision with the default constant speed of 0.5 m/s for TurtleBot, the minimum distance required between two TurtleBots was determined to be 1 m. The distance was acceptable as a Kinect sensor is able to detect obstacles up to 3.5 m away.

The procedure of the experiments is explained below:

1. TurtleBot-2 is set to move autonomously while TurtleBot-1 is set to move manually.
2. Sensors of the autonomous TurtleBot-2 were set to detect TurtleBot-1 in front via sensors and wireless communications setup.
3. TurtleBot-1 is moved manually via wireless control.
4. TurtleBot-2 followed TurtleBot-1 autonomously with constant speed.
5. TurtleBot-1 is instantaneously stopped manually.
6. Response of TurtleBot-2 is observed and recorded.
7. The experiment is repeated for 10 times.

Junction crossing intersection collision scenario

Figure 3 shows the experimental setup of the research, namely JCICS. For the JCICS, antipodal circle method is used to calculate the distance required between two TurtleBots as shown in Figure 5.

\[ D_1 = D_2 \]  

(3)

where \( D_1 \) represents the distance between TurtleBot-1 and the circle origin and \( D_2 \) represents the distance between TurtleBot-2 and the circle origin. In this scenario, assuming 2 s of time before collision with default constant speed of 0.5 m/s for both of the TurtleBots, the minimum distance required between TurtleBot and the circle origin is determined to be 1 m. This means both

---

1. Both TurtleBot hardware connections are calibrated and ensure nodes are registered in ROS.
2. Both TurtleBot are configured and set up for RECS and JCICS, respectively.
3. Both TurtleBot are placed/set to the initial positions and the algorithms are uploaded to both TurtleBot for field testing.

---

**Figure 2.** Flowchart of the proposed collision avoidance system.
TurtleBots will be placed perpendicular to each other with the same 1 m distance since the distance of TurtleBot-1 to the circle origin is equal to the distance of TurtleBot-2 to the circle origin.

The procedure of the experiments is shown below:

1. TurtleBot-2 is set to move autonomously while TurtleBot-1 is moved manually.
2. Sensors of the TurtleBot-2 were set to detect TurtleBot-1 in front of junction via sensors and wireless communications setup.
3. TurtleBot-2 in front of the junction is moving autonomously with constant speed.
4. TurtleBot-1 moves manually to intercept/cross paths with the TurtleBot-2 via manual control.
5. Response of TurtleBot-2 is observed and recorded.
6. The experiment is repeated 10 times.

The result and discussion are as follows:

There are two results obtained in this work which are CASA-1 and CASA-2 being implemented on simulation and real-world experiment. This work also contains the
information of finalized simulation and experiment setup with a detailed explanation, and also the RECS and JCICS simulation and experiment implementation.

**Simulation and experimental setup**

Using Gazebo simulation, straight and junction roads world were created, respectively, each with two TurtleBots included in the simulation world. The real environment was then mapped using gmapping function, guided with the ROS visualization Rviz program to visualize the mapping process. Figures 6 and 7 show the simulation world setup and the mapped image of the simulated world that simulate straight road and junction road scenario, respectively. For real-world experiments, open space with a flat surface was used to create the workspace for the experiment. Maps of the open space were generated using ROS gmapping package on TurtleBot. Figure 8 shows the setup of the experiment workspace and the mapped image of the workspace.

**Simulation and experiment implementation**

Both simulation and real-world experiments were conducted separately in RECS and JCICS. By utilizing the created world and map, both simulation and real-world experiments were carried out by using two TurtleBots. For each scenario, four different conditions are tested which are Lead TurtleBot sudden stop (LTS), Lead TurtleBot decelerate (LTD), Lead TurtleBot slower speed (LTSS), and straight crossing path (SPD). Simulations and experiments were repeated 10 times for each condition.

**RECS simulation and experiment.** TurtleBot-1 and TurtleBot-2 were set and run for LTS, LTD, and LTSS conditions. Figure 9 shows the process of the simulation for the RECS simulation. TurtleBot-2 (Follower) was set to move autonomously toward another endpoint via navigation goals. TurtleBot-1 (Lead) was controlled manually and leads in front of the TurtleBot-2. TurtleBot-1 was manually stopped/decelerate after halfway of the track. TurtleBot-2
collision avoidance reactions were observed and recorded. Figure 10 shows the process of the real-world experiment. TurtleBot-2 (Follower) is set to move autonomously toward another endpoint via navigation goals. TurtleBot-1 (Lead) was controlled manually and leads in front of the TurtleBot-2. TurtleBot-1 was manually stopped/decelerate after halfway of the track. TurtleBot-2 collision avoidance reactions were observed and recorded.

**Junction road collision scenario simulation and experiment.**
TurtleBot-1 and TurtleBot-2 were set and run accordingly for junction crossing collision scenario. Figure 11 shows the process of the simulation for the junction crossing collision scenario. TurtleBot-2 was set to move autonomously toward an endpoint via navigation goals. Simultaneously, TurtleBot-1 was controlled manually toward TurtleBot-2 navigation path perpendicularly. TurtleBot-2 collision avoidance reactions were observed and recorded. Similarly, TurtleBot-1 and TurtleBot-2 were set and run accordingly for JCICS. Figure 12 shows the process of the real-world experiment for the JCICS. TurtleBot-2 was set to move autonomously toward another endpoint via navigation goals. TurtleBot-1 was controlled manually and leads toward TurtleBot-2 navigation path perpendicularly. TurtleBot-2 collision avoidance reactions were observed and recorded.

**Simulation and experiment results with CASA-1 and CASA-2 algorithm**

**Rear end collision scenario.** Based on the obtained simulation results in Table 1, it is shown that the success rate of
collision avoidance running with the CASA-1 algorithm is 60%, 70%, and 80%, respectively, for LTS, LTD, and LTSS scenarios. It is shown that the LTS scenario has the lowest success rate since only 6 out of 10 trials are successful, followed by LTD and LTSS scenarios. Figure 13 shows the detailed simulation and experiment results for the RECS for LTS, LTD, and LTSS scenarios with CASA-1 algorithm. On average, the time required for the TurtleBot-2 to reach its navigation goals while evading collision is about 17.0 s for LTS scenario, 18.0 s for LTD scenario, and 17.0 s for LTSS scenario. Note that the average times include the times where collision evasion is unsuccessful, which usually caused longer time required for TurtleBot-2 to complete the navigation goals.

Experiment results in Table 1 show an increase in success rate compared with simulation results with 70%, 80%, and 80% success rates for LTS, LTD, and LTSS scenarios, respectively. LTD and LTSS have the same success rate compared with the simulation results. These differences might be due to human error or random condition such as...
the environmental conditions during the experiment. Overall, the results are acceptable and consistent with the simulation results. The average time required for the TurtleBot-2 to reach its navigation goals while evading collision is 17.0 s for LTS scenario, 17.4 s for LTD scenario, and 18.2 s for LTSS scenario.

Figure 14 shows the detailed simulation and experiment results for the rear end collision for LTS, LTD, and LTSS scenarios using CASA-2 algorithm. Based on the simulation results in Table 1, after integrating the velocity and position into the current system (CASA-2), a 100% success rate of collision avoidance was achieved for LTS, LTD, and LTSS scenarios, as shown in Table 1. On average, the time required for the TurtleBot-2 to reach its navigation goals while evading collision is about 15.0 s for LTS scenario, 17.0 s for LTD scenario, and 15.7 s for LTSS scenario. Overall, the experiment results are consistent with the simulation results with CASA-2 algorithm.

Table 1. Success rate results for the rear end collision scenario with CASA-1 and CASA-2 algorithms.

| Number of trials | LTS  | LTD  | LTSS |
|------------------|------|------|------|
| Simulation CASA-1| 60%  | 70%  | 80%  |
| Experiment CASA-1| 70%  | 80%  | 80%  |
| Simulation CASA-2| 100% | 100% | 100% |
| Experiment CASA-2| 100% | 100% | 100% |

CASA-1: collision avoidance system without inter-robot communication; CASA-2: collision avoidance system with inter-robot communication; LTS: Lead TurtleBot sudden stop; LTD: Lead TurtleBot decelerate; LTSS: Lead TurtleBot slower speed.

Junction crossing intersection collision scenario. From the simulation results obtained (Table 2), it is shown that the success rate of collision avoidance for junction crossing intersection is just merely 60% with 6 successful attempts out of 10. This may be due to Kinect sensor’s limitation on horizontal viewing angle (70°), thus unable to detect TurtleBot-1 that approaching from perpendicular direction. Figure 15 shows the detailed simulation and experiment results for the junction crossing intersection collision for SPD with the CASA-1 algorithm. The average time required for the TurtleBot-2 to complete its navigation for the SPD scenario is around 21.0 s. The success rate of collision avoidance via experiment was increased by 10%, as shown in Table 2, if compared with the simulation results for the same scenario. The experiment results were consistent with the simulation results. For the SPD scenario, the average time for the TurtleBot to complete its navigation goals is around 19.0 s, 3.0 s shorter compared to the simulation time. Note that the average times are calculated that includes the times where collision evasion is unsuccessful, which usually caused more time needed for the TurtleBot to fully complete its navigation goals.

Figure 16 shows the detailed simulation and experimental results for the junction crossing intersection collision for SPD with the CASA-2 algorithm. In simulation, a 100%
collision avoidance success rate is achieved via virtual simulation when using the CASA-2 algorithm. Table 2 shows the simulation results for the JCICS. In this SPD scenario, the average time needed for the TurtleBot to complete its navigation goals is merely 14.0 s. This is due to the smoothness of the navigation, where there is no collision and jerky movement when traveling toward the navigation goals. The experiment shows a 100% collision avoidance success rate when using the CASA-2 algorithm. Table 2 shows the real-world experiment results for the JCICS when running with the CASA-2 algorithm. For the SPD scenario, by using the CASA-2 algorithm, the average time needed for the TurtleBot to complete its navigation goals is around 13.7 s, slightly shorter in terms of time compared with the average time of simulation with 14.0 s. The results consist of the simulation and are accepted. The shorter time...
required for TurtleBot-2 to complete the experiment may be due to the better navigation system provided by the CASA-2 algorithm.

**Discussion**

There are several failure cases, problems, and limitations encountered during the testing mainly involving TurtleBot localization method, limitation of TurtleBot itself in simulating as an autonomous car, and the recalibration of the TurtleBot needed after a few runs of the experiment.

**Trajectory data**

Figures 17 and 18 show the trajectory data example of TurtleBot during the collision avoidance maneuver using the CASA-1 algorithm. The trajectory data are obtained by using Rviz in real time during the simulation and experiment. The figure on the left shows the trajectory path continues with a smooth gradient indicating a successful collision avoidance maneuver is achieved by TurtleBot. Figures on the right show the failed collision avoidance where the trajectory path seems to be stuck and “looping,” indicating that collision has occurred. For this condition, TurtleBot is stuck and entered into recovery mode by turning in a circular motion to scan the surrounding. Hence, TurtleBot will be able to find alternative routes before continuing its path toward the navigation goals.

Figure 19 shows the trajectory data example of TurtleBot during the collision avoidance maneuver using the

| Table 2. Success rate results for junction crossing intersection collision scenario with CASA-1 and CASA-2 algorithms. |
|----------------------------------------------------------|
| Number of trials | SPD |
| Simulation CASA-1 | 60% |
| Experiment CASA-1 | 70% |
| Simulation CASA-2 | 100% |
| Experiment CASA-2 | 100% |

CASA-1: collision avoidance system without inter-robot communication; CASA-2: collision avoidance system with inter-robot communication; SPD: straight crossing path.

![Figure 15](image-url) Graph of SPD scenario for junction crossing intersection collision using CASA-1 algorithm in simulation and experiment. SPD: straight crossing path; CASA-1: collision avoidance system without inter-robot communication.

![Figure 16](image-url) Graph of SPD scenario for junction crossing intersection collision using CASA-2 algorithm in simulation and experiment. SPD: straight crossing path; CASA-2: collision avoidance system with inter-robot communication.
CASA-2 algorithm. Using the CASA-2 algorithm, both TurtleBots will be able to communicate and exchange data (velocity and position) through the LCM system. Then, TurtleBot-2 is able to prevent the collision with TurtleBot-1 while navigating to its own goals in both road scenarios (RECS and JCICS). The trajectory data of TurtleBots are obtained by using Rviz in real time during the simulation and experiment, which means both are connected wirelessly via the LCM system.

Localization problem using AMCL

TurtleBot defines its location based on the AMCL method rather than GPS for an autonomous vehicle. Monte Carlo localization, also known as particle filter localization, is a widely applied localization method in the field of mobile robotics. The algorithm estimates the position of a robot as it moves and senses the environment. For TurtleBot, it uses its sensor’s readings to estimate its location relative to its starting position. Due to this reason, a false localization problem or a Kidnapped robot scenario commonly occurs if there are several locations that are represented similarly according to the sensor values. For example, in straight road collision scenario simulation world setup, a long hallway in which both ends are almost the same. As a result, TurtleBot sometimes will localize itself in the wrong position. To reduce this error in the work, boxes are arranged differently in sequence in the simulation world while two markers (a box and a table) are used in a real-world experiment.

TurtleBot limitation in representing actual autonomous vehicles

Due to TurtleBot’s non-holonomic drive characteristic and only use two wheels rather than four wheels for normal autonomous cars, collision scenario data of TurtleBot in this work might not be able to fully represent the actual collision scenario of the four wheels autonomous car. Besides that, TurtleBot’s relatively low maximum speed limit (0.7 m/s) and its sensors’ ability to detect obstacles up to 5 m mean that TurtleBot has a higher probability to evade collision since it got longer time to detect and avoid collision. Hence, it might influence the overall results. Both aspects are considered as the research limitation.

Frequent recalibration of TurtleBot positioning in a real-world experiment

Other than that, the TurtleBot position needs to be recalibrated for each 5–7 runs in the real-world experiment. This may be due to the wheel slip since the surface of the experiment location is made out of tile. However, the calibration is simple and easy to be done. Before the experiment started, a point location is marked using tape based on the Rviz publish location function. The point is then set as the navigation goal for the TurtleBot. By doing this, if the TurtleBot does not stop at the marked spot after completing the navigation goal, calibration can be done by simply lifting and placing the TurtleBot at the exact marked spot.

Conclusion

An improvement of the collision avoidance system of the current autonomous car is proposed. The proposed system incorporates the DSRC system into the current autonomous
car collision avoidance system which enables the data transmission (velocity and position) between vehicles to further improve the current collision avoidance system. The proposed approach was tested using TurtleBot via simulation and real-world experiments. The algorithm of the collision avoidance system was firstly developed under two specified scenarios, then tested in the simulation. Later, the proposed algorithms were validated in real-world experiments. By using the proposed algorithms, the collision between TurtleBots can nearly be prevented with an almost 100% success rate compared with 72.5% without using the algorithm. Although promising results are obtained in this research, there are several limitations since some assumptions are made in this work. The research work was completed on a miniature version by using mobile robots such as TurtleBots where attempted to replace the actual four wheels vehicles. TurtleBots with differential drive robots were used for the simulation and experiment. Simulation and experiment were carried out in a controlled environment. Hence, several recommendations were suggested to improve this work. Firstly, it is recommended to use four-wheeled robots such as Husky or Jackal UGV rather than two-wheeled TurtleBot since both have higher maximum speed and physically more suitable to represent an autonomous car. For the collision avoidance system, the system that can pinpoint the accurate location of TurtleBot in a known map such as beacon system might be better than using an estimated location via AMCL is applied to improve its accuracy. Other recommendations include integrating a Wi-Fi modem into the TurtleBot-2 for a better connection between TurtleBot and workstation and also further improving the current navigation stack algorithm to make the TurtleBot movement more robust, flexible, and respond faster.

Acknowledgement
The authors would like to acknowledge the CAMARO Research Group of Universiti Tenaga Nasional for the usage of facilities and equipment.

Declaration of conflicting interests
The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding
The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the Ministry of Education, Malaysia (grant no. FRGS/1/2019/TK04/UNITEN/01/2).

ORCID iDs
Yew Cheong Hou https://orcid.org/0000-0002-7375-1089
Khairul Salleh Mohamed Sahari https://orcid.org/0000-0002-3838-501X

References
1. Autopilot, https://www.tesla.com/autopilot (accessed 9 September 2019).
2. United States National Transportation Safety Board. Highway Preliminary Report: HWY19FH008, https://www.ntsb.gov/investigations/accidentreports/pages/hwy19fh008-preliminary-report.aspx (accessed 9 September 2019).
3. Tsilker B and Orlov S. Computing parallelization efficiency estimation in the intelligent transportation systems. In: Proceedings of the 10th International Conference “Reliability and Statistics in Transportation and Communication” (RelStat’10), pp. 218–224, 20–23 October 2010.
4. Lange S, Ulbrich F, and Goehring D. Online vehicle detection using deep neural networks and lidar based preselected image patches. In: 2016 IEEE intelligent vehicles symposium (IV), Gothenburg, Sweden, 19–22 June 2016, pp. 954–959. IEEE.
5. Bajpayee D and Mathur J. A comparative study about autonomous vehicle. In: 2015 international conference on innovations in information, embedded and communication systems (ICIIECS), Coimbatore, India, 19–20 March 2015, pp. 1–6. IEEE.
6. Yuan T, Duraisamy B, Schwarz T, et al. Track fusion with incomplete information for automotive smart sensor systems. In: 2016 IEEE radar conference (RadarConf), Philadelphia, PA, USA, 2–6 May 2016, pp. 1–4. IEEE.
7. Zhang W, Zhang J, Wang B, et al. The research on early-warnin methods of tree barriers of transmission lines based on LiDAR data. In: 2016 4th international conference on applied robotics for the power industry (CARPI), Jinan, China, 11–13 October 2016, pp. 1–4. IEEE.
8. Li TH, Chang SJ, and Chen YX. Implementation of autonomous fuzzy garage-parking control by an FPGA-based car-like mobile robot using infrared sensors. In: 2003 IEEE international conference on robotics and automation (Cat. No. 03CH37422), Taipei, Taiwan, 14–19 September 2003, Vol. 3, pp. 3776–3781. IEEE.
9. Cherubini A, Spindler F, and Chaumette F. Autonomous visual navigation and laser-based moving obstacle avoidance. IEEE Trans Intell Transp Syst 2014; 15(5): 2101–2110.
10. Zheng Y, Jin L, Gao L, et al. Development of a distributed cooperative vehicles control algorithm based on V2V communication. Procedia Eng 2016; 137: 649–658.
11. Jiang L, Wang YP, Bai-gen C, et al. Multi-sensor based vehicle autonomous navigation for vehicle infrastructure integration: concept and simulation analysis. In: Proceedings 2011 international conference on transportation, mechanical, and electrical engineering (TMEE), Changchun, China, 16–18 December 2011, pp. 698–702. IEEE.
12. Tsuboi T, Yamada J, Yamauchi N, et al. Cell design for next DSRC applications. In: 2009 international conference on ultra modern telecommunications & workshops, St. Petersburg, Russia, 12–14 October 2009, pp. 1–5. IEEE.
13. Ammoun S, Nashashibi F, and Laurgeau C. Real-time crash avoidance system on crossroads based on 802.11 devices and GPS receivers. In: 2006 IEEE intelligent transportation
systems conference, Toronto, Ontario, Canada, 17–20 September 2006, pp. 1023–1028. IEEE.

14. Robotic Operating System (ROS), [Online] https://www.ros.org. (accessed 9 September 2019).

15. Maruyama Y, Kato S, and Azumi T. Exploring the performance of ROS2. In: Proceedings of the 13th international conference on embedded software, Pittsburgh, PA, USA, 2–7 October 2016, pp. 1–10. New York: ACM.

16. Będkowski J, Pelka M, Majek K, et al. Open source robotic 3D mapping framework with ROS—robot operating system, PCL—point cloud library and cloud compare. In: 2015 international conference on electrical engineering and informatics (ICEEI), Denpasar, Indonesia, 10–11 August 2015, pp. 644–649. IEEE.

17. Huang AS, Olson E, and Moore DC. LCM: lightweight communications and marshaling. In: 2010 IEEE/RSJ international conference on intelligent robots and systems, Taipei, Taiwan, 18–22 October 2010, pp. 4057–4062. IEEE.

18. Van Den Berg J, Guy SJ, Lin M, et al. Reciprocal n-body collision avoidance. In: Pradalier C, Siegwart R, and Hirzinger G (eds) Robotics research. Berlin, Heidelberg: Springer, 2011, pp. 3–19.

19. Guy SJ, Chhugani J, Kim C, et al. ClearPath: highly parallel collision avoidance for multi-agent simulation. In: Proceedings of the 2009 ACM SIGGRAPH/eurographics symposium on computer animation, New Orleans, Louisiana, USA, 1–2 August 2009, pp. 177–187. New York: ACM.

20. Claes D, Hennes D, Tuyls K, et al. Collision avoidance under bounded localization uncertainty. In: 2012 IEEE/RSJ international conference on intelligent robots and systems, Vilamoura, Portugal, 7–12 October 2012, pp. 1192–1198. IEEE.

21. Brown TL. Adjusted minimum time-to-collision (TTC): a robust approach to evaluating crash scenarios. In: Proceedings of the driving simulation conference North America, Orlando, FL, USA, 30 November–2 December 2005, pp. 40–48.

22. Lee JD, McGehee DV, Brown TL, et al. Collision warning timing, driver distraction, and driver response to imminent rear end collisions in a high-fidelity driving simulator. Hum Factors 2002; 44(2): 314–334.

23. Alonso-Mora J, Breitenmoser A, Beardsley P, et al. Reciprocal collision avoidance for multiple car-like robots. In: 2012 IEEE international conference on robotics and automation, Saint Paul, MN, USA, 14–18 May 2012, pp. 360–366. IEEE.