Agent Based Test Generation for Autonomous Driving

Guangyuan Liu

18044524

MSc Robotics

Engineering Mathematics

University of the West of England, Bristol

September 19, 2019

Declaration

This study was completed for the MSc in Robotics the University of the West of England, Bristol. The work is my own. Where the work of others is used or drawn on it is attributed.

The dissertation may be made freely available immediately for academic purposes (this includes display in the Library for consultation purposes)

Word count: 12163

SIGNED: Guangyuan Liu DATE: 17/09/2019

Abstract

In the autonomous driving field, for the sake of security and cost, researches tend to choose simulation technology to find potential problems before doing the test in the real world. Not only does it allow researchers to do repeated tests to test the stability of autonomous vehicles under the same conditions, but it can also test the performance of vehicles by putting it in a dangerous situation which is not allowed in the real world. In addition, in the last decades, the multi-agent system (MAS) has been implemented in many areas. As a novel application of MAS, we think that the agent-based simulation technology is very suitable for the autonomous driving simulation because of its distributed and customizable characteristics, which means all agents have independent internal structure and can autonomously response the outside environment. Also, MAS can allow verification engineers to easily add more agents in a developed system instead of taking a lot of time in writing specific test cases on the traditional testing platform to test whether the vehicle agents obey the assertion. In this paper, we develop an agent-based test scenario aims to test the collision assertion. It includes a straight road with several pedestrian crossing. We propose three modes of the pedestrian agent to analyse the test efficiency. Finally, we propose an approach that can transfer the newly developed agent behaviour from a grid world in Jason platform to a three-dimensional (3D) simulator, Carla.

Keywords: Multi-agent systems, Test generationAgentSpeak, Jason, Carla

Acknowledge

Firstly, I would like to thank Prof Kerstin Eder and Dr Greg Chance who are the supervisors of my dissertation. To my best appreciate, I thank them for their considerable patience and valuable guidance. Without their help, I could not complete my essay.

I also want to thank those who supported and encouraged me. Without their encouragement, I could not have the powerful belief to enable me working hard on the dissertation.

Contents

1 Introduction

As the introduction of the paper, regarding the topic of the multi-agent test system for autonomous driving, this chapter will introduce its research background and motivation. The main contribution and the organisation structure of this article will also be demonstrated briefly.

1.1 Background

In the autonomous driving research field, there is no doubt that the real road driving test is a convincing and direct way to find out potential problems and risks of autonomous vehicles, but for the sake of cost and security, more and more researchers have put their interests in the simulation test. Compared to the on-road testing where it is difficult to control environmental variables, experiments performed on the simulator can more easily obtain operational data of the vehicle under the same conditions. Researchers can utilise the simulation system to repeat the experiment under the same conditions or to control parameters to test the performance of the vehicle, which is very convenient for them to tackle existing problems.

Due to the separable and customised characteristics, MAS has developed rapidly since being created in the 1980s. It draws on and combines numerous achievements of various subjects, including artificial intelligence, sociology, philosophy and so on (Ferber and Weiss, 1999). MAS means that a system composed of multiple interacting agents that have specific goals and relative actions (Bordini and Hübner, 2005). There are a lot of definitions of agents in the past researches. Here this paper considers that the agent in the multi-agent system is the representation of entities of interest of actors in an execution environment, whose typical structure is belief-desire-intention (BDI) model. BDI architecture is proposed by the Stanford Research Institute in the Rational Agency project, and it represents the three parts of the internal structure of the agent respectively (Reynolds, 1987). In the autonomous driving area, it is used to regard vehicles, pedestrians and other elements such as traffic lights as independent agents with beliefs, desires and intentions (Maleš and Ribarić, 2016). The typical model structure and operating process of agents shown in the figure below.

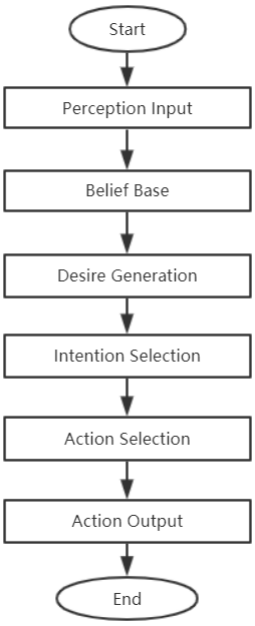

Fig.1. BDI structure and operating process of agents

Firstly, "Belief" is the agent's cognition of the world, including the data describing its own states and the agents' surrounding environment programmed by Java language, such as the speed and direction of other vehicles, road condition and its own movement state, which are the basis for generating the next goal for each agent. Before the system starts running, every agent usually endowed with some initial belief to control them to select the next goal. The beliefs can also be easily added or deleted in the process of system operation to achieve the purpose of changing the state of agents and the environment. The environment is a Java development space where agents can receive information and interact with each other. Furthermore, "Desire" represents the environment state that the agent wants to accomplish next. Based on the content of belief and pre-set rules, the system can set the next state of the environment and find suitable actions to accomplish, such as acceleration, deceleration, alignment and turning. Note that not all desires can influence the choice of goals, and it is also common for different agents to have opposite desires. Finally, "Intention" is the goal to be achieved by agents, which has a guiding role for current actions (Bordini, Hübner, and Wooldridge, 2007). The goal could be given to the agent by other agents via interaction, or it could be picked out in the pre-set options. The agent often has a lot of intentions at the same time. If meeting this situation, the agent will choose one from them in the order they are placed in the program. In conclusion, in a typical running process of MAS, each agent starts with some initial beliefs and intentions, then thinks about the next possible

options and achieve it according to the environment changes. As the cornerstone of MAS, AgentSpeak is an agent-oriented programming (AOP) language and has a deductive style which seems to the Prolog language. The basic idea behind it is to design the know-how in front of specific actions (Bordini, Hübner, and Wooldridge, 2007). Know-how means knowledge about how to choose actions to achieve goals. For example, if the goal of an agent is to go to the airport, the know-how can be two plans that it can take a bus or a taxi. The plans are specific actions responding to the goal. In this way, agents can not only accomplish goals by utilising the know-how, but can also respond appropriately to changes in the environment. As the development platform of AgentSpeak, Jason provides a great development tool that enables agents to interact with each other in a high-level way. It means that the agent is able to communicate as an individual and complete actions to tackle some problems instead of only getting a byte from A to B.

1.2 Motivation

According to the characteristics mentioned above, MAS works in theory for traffic and transportation testing systems that have numerous separate vehicles and other elements. It is worth noting that air traffic control and conflict management systems are some of the first applications of multi-agent technology (Cammarata, McArthur, and Steeb, 1988). There is a very creative airport robot management system (Bordini, Hübner, and Vieira, 2005). According to the different functions of the robot, the system can assign robots to perform labelling and transportation tasks at different times to avoid interference. The system allows the robot to work in an orderly fashion under the normal conditions expected by its designers, but it cannot deal with other special cases correctly such as the type of goods cannot be identified. In this situation, the system will randomly assign a robot to inspect the package. If the robot is not competent, another robot will be selected to do the same work. This approach to special cases is very effective for systems that contain a small number of agents or do not have repetitive experiments, but the amount of time wasted grows rapidly with the number of robot type grows. To tackle this problem, Neil and Brian proposed a mechanism for modular constructions of agents. As shown in figure 2, the beliefs from the environment to the agent is imported as a module. The internal content of belief module is invisible for agents to avoid the conflicts of some beliefs. This mechanism solves the problem of conflicts to some extent and simplifies the process of updating belief, but the execution mechanism of the triggering event has not changed and some agent updates are redundant, which still

reduce operational efficiency to a large extent.

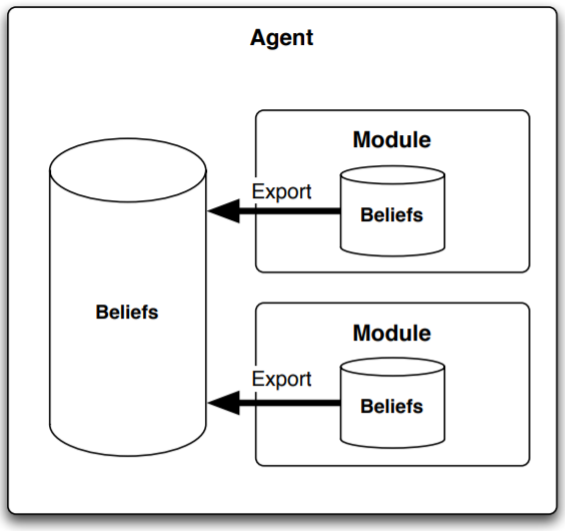

Fig.2. Exportation of beliefs within modules (Madden and Logan, 2009)

Lesser and Corkill (1983) proposed another method to reduce the interference of different beliefs. According to the different geographical distribution of data and the target of communication, the concept of the blackboard was put forward (Corkill and Lesser, 1983). It means that each agent can only sense a part of the monitored area and make decisions via communications with other agents. The agent can perceive some information from the environment and other agents as well as autonomously performs some actions depends on what goals need to be achieved instead of executing plans at the fixed time or process. This approach reduces the amount of calculated data by narrowing the scope of agent interaction. Although these methods improve the operation efficiency of the system by optimising the structure of MAS, they do not reduce useless plans at the agent level. As an essential programming philosophy to transforming abstract theoretical research into the actual application, AOP is a new design method that makes large and complex systems simpler and more elegant by abstracting and distributed them into a lot of independent agents. Via the multi-agent structure, the traditional control system is decomposed into multi-task-oriented agents. In this way, it can be significantly optimised in terms of improving efficiency and reducing time consumption, which is of great significance for the pursuit of safe and efficient autonomous vehicle technology (Wang et al., 2005). Thus, in recent years, more and more researches apply agent-based methods to the problem of autonomous vehicles, such as autonomous parking, transportation planning, distributed control and traffic simulation (Jennings, Sycara, and Wooldridge, 1998).

However, although these are significant problems that need to be solved in the autonomous vehicle field, many previous researches have not created suitable agents that behave realistically. For instance, the speed of vehicles and pedestrians does not correspond to the real situation and the action of pedestrians is also not enough natural, which might result in generating a large number of useless tests and even getting wrong results. Although many of the past researches focused on vehicles, the importance of pedestrian simulation is gradually realised. For security, pedestrians are usually simulated in the test system. Visual simulation can not only provide an intuitive experimental screen for researchers but also verify and optimise the test program in the simulation process to avoid various dangerous unexpected situations in the real road test.

Moreover, the traditional test system is not available for autonomous driving. On the one hand, due to simulation test should reflect the real situation as much as possible, traditional simulation systems are usually very complex and have some limitations, which results in that the autonomous vehicle testing has difficulty in using classic mathematical formalism (Genesereth, Fikes, et al., 1992; Sycara, 1998; Ferber and Weiss, 1999). On the other hand, based on agents' perception and interaction with the environment, each agent of the multi-agent system can automatically execute pre-set actions to respond to the change of environments and states of other agents. The autonomous driving system needs to deal with any assertions in a short time, which requires a lot of assertion data training. Thus, it is essential to design a realistic and efficient autonomous driving test generation system to test assertions.

1.3 Main Contribution

Agents in this paper divided into pedestrian agents and vehicle agents and the test scenario is a straight two-way road with several pedestrians crossing. All agents are independent and take different BDI structures. Because the two vehicles heading in opposite directions only obey some simple rules and keep moving unless pedestrians are crossing the road in front of them, the movement of pedestrians is the crucial factor in the valid test generation. Although researchers in psychology and other social sciences have been observing and studying human behaviour for decades, human behaviour cannot be fully understood and predicted because of its complexity and diversity. Having learned from the achievements of building multi-agent system in past researches, the focus of this paper is to build such a simulation system to test the collision assertion. Based on the work, this paper proposes a model to design the pedestrian agent for test generation. To be specific, a scoring system is developed

to evaluate the performance of pedestrian agents which have random movement or specific rules to control their actions. After the experiment on the Jason platform, the various data obtained are imported into Carla software based on Unreal Engine 4 (UE4) to create a three-dimensional (3D) scenario that can help developers intuitively understand the operation of the test system. The work in this paper provides an evaluation method for the design of a pedestrian agent in the agent-based autonomous driving simulation system.

1.4 Outline

In section 2, the related work about the implementation of MAS in the autonomous vehicle driving testing and traffic area will be outlined, especially the pedestrian simulation. Section 3 presents the construction of the test system with an analytical description of the system design and the approach for data collection. In section 4, the performance of different pedestrian behaviour patterns will be analysed. Finally, this paper will conclude with a summary and propose the next future developing directions.

2 Literature Review

Although the technique of MAS has developed only about 30 years, it has been implemented in many areas, such as business process modelling, large population simulation, computer games and telecommunications. Such a system usually has a very complex structure and numerous independent modules, so an agent-based system is convenient for developers to learn more about the behaviour and interaction of each module respectively (Georgeff, 1988). Having learned from previous research, a sound multi-agent system must have a clear agent structure, rigorous interaction rules and complete environment. Only in this way, each agent has plans to deal with all expected normal and abnormal events. If meeting an unexpected situation that it does not have suitable plans to deal with it, it should turn to other agents for solutions to prevent system crashes (Bazzan and Klügl, 2014). Autonomous driving is such a system that needs to deal with a list of emergencies in a short time. It must have an emergency processing strategy that considering the information of multiple individuals around to determine the best route. Therefore, with the demand for intelligent traffic system is growing, many achievements have been made on MAS in the field of traffic. A list of typical research findings will be analysed in the following sections of this chapter.

2.1 Agent-based simulation

In the past ten years, with the rapid improvement of the performance of the processor, agent-based modelling has been widely used in a lot of areas, such as games, film design, crowd evacuation, military training and so on (Weiss et al., 2017). Currently, in the field of automatic driving modelling, the rapidly developing approach is also the agent-based model [10]. In the agent-based autonomous driving system, every vehicle and pedestrian in the real world are coded as individual agents which has multiple modules such as awareness, behaviour, memory and decision-making. Different agents have different BDI model, so the system can reflect the differences among each individual and gain results that are close to reality. In the agent model, the simulation individuals can automatically perceive the external environment information and determine its next action independently by relying on its own

rules. Therefore, a basic problem in all agent-based modelling methods is how to design appropriate behaviour rules to obtain more realistic simulation results. Once the correct behaviour rules are established, agents can be used to predict and execute "what-if" decision processing and make corresponding intelligent simulation behaviours.

According to the previous research, the related achievements can be divided into the rule-based method and data-based method. The rule-based method makes agents in the environment act as decision modules according to some predefined rules set, which is a traditional agent simulation method. In the 1980s, the flocking behaviour of birds was simulated through using simple manipulation behaviour to control the movement of the agent (Reynolds, 1987), but the bird agents do not have many interactions with environments and do not have too much different from each other. Helbing proposed a classic social force model (SFM) which represented the interaction between agents and virtual environment via the form of force (Zhang, Thangarajah, and Padgham, 2007). Based on this theory, the complex motion of agents is simplified to the solution of a simple dynamic equation. Due to its simplicity of expression and high efficiency of calculation, SFM has been widely used in the field of crowd simulation.The Optimal Energy Allocation (OEA) algorithm is used to simulate the movements of pedestrian agents, which can solve the problem of superposition of large-scale people (Lakoba, Kaup, and Finkelstein, 2005). Based on the Social Force Model, in the visual simulation area, Kairan et al. improved pedestrian self-driving and proposed a quantitative calculation method, including a series of parameters such as the tension coefficient. Then they visualised the crowd evacuation under the emergency on the Analogic simulation platform.

In addition to the rule description using mathematical expressions, researchers use finite state machines and behaviour trees to control the behaviours of characters. In the behaviour tree model, the behaviour tree describes the behaviour logic in the form of nodes and the agent can switch its behaviour logic state by sensing the changes of the external environment. These methods are combined with the crowd simulation development of military sandbox system and a virtual human model with some credibility is obtained (Mckenzie et al., 2008). It has an excellent performance in the group people simulation but lack of individual agent control. Furthermore, because rule-based modelling is highly dependent on expert experience and human experimentation, the models obtained usually have certain limitations and cannot be transplanted to other applications. The complexity of pedestrian movement also makes it difficult to find rules that can correctly represent pedestrian movement. On the one hand, to obtain a better simulation performance, the pedestrian agent has to be modified according to the task

requirements of each system. On the other hand, the current simulation system is usually dynamic, so the performance of the static rule-based method is not very stable.

To reduce the negative effects of the rules, some researchers tend to conduct a large number of randomised experiments to improve the reliability of the system. An agent design method with Bayesian adaptive randomisation was proposed to help patients to gain more effective treatments (Lee, Gu, and Liu, 2010). Due to the performance of processors growing rapidly, machine learning using the idea of randomisation has received much attention. However, randomisation, which takes a lot of time and generates useless data, is not suitable for autonomous vehicles, which need to deal with emergencies quickly in the running process. In this paper, the basic architecture of MAS is used to design the pedestrian agent that conforms to real people from the perspective of rule design.

2.2 Agent-based test

Compared with traditional test software, the difficulty degree of the agent-based test system is higher because in a large MAS, there may have hundreds of agents and each agent has its model, code and state. The multiple distributed processes and the independent programming for each agent not only increase the development time but also make it difficult for researchers to grasp the influence of the change of an agent on the operation of the whole system. To tackle this problem, the view and evaluation standard of agent-based system development is changing, which means that the final actual performance of the system is no longer regarded as the sole evaluation indicator. Thus, the design phase of the system is divided into three aspects, which is shown in the following figure.

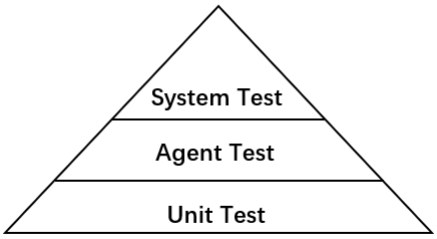

Fig.3. Agent-based test level

Unit tests focus on the internal structure of agents and check that plans of agents are working correctly. Agent tests detect the communication with agents and the interaction with the environment to make sure that all agents can work properly as designed. The system test is to check whether the

system achieves the expected quality properties such as fault tolerance and reliability. Specific research achievements on each layer will be analysed in detail next.

2.2.1 Unit Test

The unit test requires that each agent could follow its own rules and make a correct response to normal and abnormal environment changes. An improved Prometheus agent development methodology (Padgham and Winikoff, 2005) was proposed to build a test framework (Zhang, Thangarajah, and Padgham, 2007). This framework focus on the generation of test cases and the design of agent plans. Compared with traditional software systems, plans are executed by events which is a classic characteristic of MAS. However, this model does not tackle the space problem effectively, so the complex structure of MAS and numerous agents make the system run inefficiently. Learned from this paper, a more structured and efficient approach to ensure that agents can work in an orderly manner is explained in detail in the following chapter.

Furthermore, this paper only focuses on the performance of the unit level, so the work of this paper could not be extended to the integration level. Besides plans of agents, A new testing environment tool (SEAUnit) was designed (Ekinci et al., 2008). The goals of this tool are executed as ordinary plans in the planning engine instead of being seen as a new class, which separates the test tool from the internal execution to avoid a high degree of coupling between the test tool and agent infrastructure. Although the new architecture proposed in this paper may help researchers to develop an agent-based system with the test-driven goal style, the maintenance of agent goals has to need a lot of extra work.

2.2.2 Agent Test

In this level, the test focus on the performance of each agent, such as the capabilities to perceive the environment and achieve goals. Based on Jason and JADE, a framework was built to help developers to build a test suite for the multi-agent system (Caire et al., 2004). Although researchers may be able to use this platform to create multiple levels of test such as unit tests and acceptance test, the test is only valid for the single agent. If adding the number of agents, there will be a lot of work need to be done by researchers and the actual performance does not have a significant improvement. In addition, the toolkit has difficulty in recording data and does not support the low-level development for unit tests.

Furthermore, a tracing method and create a tracer tool were (Lam and Barber, 2004). Through

logging the dynamic running data of each agent, the tool can track the behaviour of agents and generates modelled interpretations. Based on this function, it can effectively help researchers to debug agents when developing multi-agent systems and gain insight into the operational performance and specific behaviour of each agent via these modelled interpretations. However, although it provides an interesting way to evaluate agent behaviours which can be seen as an important improvement for the agent test, it does not propose specific measures for improvement. To further improve the performance of the evaluation tool for systems, a formal framework with two test approaches that are active and passive tests was proposed (Núñez, Rodrıguez, and Rubio, 2005). The active test is defined as a specific event in the agent. Though compared with the formal specification, the results of the active test can reflect potential faults. Moreover, the passive test is only used to observe the behaviour of agents. Different from the traditional situation, active tests are given by agents themselves, which enable developers to set independent test cases in each agent. Unfortunately, it is same as the framework proposed by Claire et al. that the current framework cannot be implemented in the low-level test. Relying on the use of Mock agents, a unit testing method was proposed (Coelho et al., 2006), but much of their work belongs to the agent level. Each Mock agent with a test script is responsible for the single angle. Inspired by the JUnit (Parveen et al., 2009), they proposed a testing framework called SUnit to enable developers to write tests for agent behaviours and communication between agents. Their work has an excellent performance when the test focuses on every single agent and in the next step, they have to address the problem that how to use Mock agents to test agents' integration scenarios.

2.2.3 System Test

The system test aims to check some quality attributes of MAS, such as fault-tolerance, adaption and performance. It means that as a whole system whether MAS can achieve the expected performance in the intended environment. Some previous research have been devoting to test MAS behaviours at the macro level.

A systematic method was presented for the validation of macroscopic system dynamics (Sudeikat and Renz, 2008). This approach showed how to describe cyclic causalities generating the macroscopic behaviours and validate the agreement between systemic MAS models and simulation MAS models. The use of agreements can indicate the proper use of MAS strategies, but the work of this paper is about the static MAS model and it does not provide the support to the automation of MAS simulations. To deal with this problem, the suite test derivation approach mentioned above was extended to the system

level, which put the goal-oriented requirement analysis artefact at the heart of test case derivation (Houhamdi and Athamena, 2011). In this way, it can generate test suites which are used to test the achievement of goals in the dynamic system test process.

2.3 Agent-based Traffic System

Due to the restrictions of land use and the increasing number of cars, the flow pressure of roads cannot be alleviated by only increasing the number of lanes. Thus, the demand for a more intelligent traffic control system is increasing. A lot of multi-agent systems for traffic have been developed. All of them are based on the control loop (Papageorgiou et al., 2003). The loop includes:

- The physical agents (vehicles, pedestrians and traffic lights);
- The surveillance mechanism (agent detector);
- The environment with operational strategy

2.3.1 The Static Traffic Control System

The research on the static system has a long history. TRANSYT can be used to optimise the fixed-time operation by generating coordinated plans (Robertson, 1969). It is a well-established off-line optimisation method for traffic control, but can only be used for static situations inferred from historical data, which results in its unstable performance in unexpected situations. Based on TRANSYT, SCOOT-Split Cycle is a traffic-responsive approach which uses real-time data collected by detectors (Hunt et al., 1981) and SCATS-Sydney uses a hierarchical system (Luk, Sims, and Lowrie, 1982). All three systems mentioned above are reliability and robustness, but they also have some problems. These agent-based systems usually set the saturation flow to a fixed value, but it is a tough situation to happen. Researchers have to take the same amount of time to the development and maintenance of unlikely situations. This method that equalising weights reduces the efficiency of algorithms. In addition, these static systems also may not be useful in the secondary road. For instance, the roads in the commercial centre on the outskirts of big cities are likely to have great traffic pressure compared to the surrounding traffic network. Moreover, the traffic situation may be different during floods, snow and other abnormal situations. Thus, there is a gap that the priority of the vehicle agents should be reasonably allocated

according to the traffic flow. The vehicles on the same road can form an agent group that executes the same interactions with the environment to get a higher weight in the game. If the volume of vehicles exceeds the capacity of the road design, and they also get the consent of other road vehicles, the traffic light can be turned to be green or extend the duration time.

2.3.2 The Dynamic Traffic Control System

The dynamic traffic control system usually utilises various track tools to monitor the operation of the system. Based on the fuzzy inference, a single intersection test scenario was created to improve the interaction of agents (Kosonen, 2003). Therefore, there is a need that agents should decide when and how to work together as a group. The author mentioned that it could be done by exchanging the local traffic data with vehicles located on adjacent roads, but the focus could also be put on the internal controller coordination. Also, this paper does not mention the detail of the operational mechanism of the fuzzy control. Thus, future improvement here is how to design the coordination of agents via fuzzy control, especially dealing with the contradiction between local and global optimisation.

Apart from using improved management agreement, an alternative is to use a more organised structure to optimise interactions and reduce conflict between agents. Based on the hierarchical structure, An agent-based traffic control system divides its agents into three types, including authority agents, intersection agents and road segment agents (Roozemond, 2001). Based on a clear management structure, each intersection agent supervised by authority agents control one intersection and is responsible for the normal operation of the vehicle agents. Via communication with neighbouring road agents, intersection agents estimate the traffic situation in the next period on the prediction model and use pre-set rules to choose a suitable strategy. According to the prediction model and rules, the intersection agent can gain a signal plan to optimise flow control in non-conflicting cases. If there are some conflicts, the authorised agents can coordinate and reduce the loss of all agents. For this conflict situation, the author mentioned that some agents might have to sacrifice some value to gain coordination with other agents, but neither this coordination mechanism nor the rules of negotiation were explained in detail. Considering that the traffic simulation system requires any abnormal situations should be dealt within a short time, the agents tend to choose locally optimal results. Therefore, further research is needed on how to coordinate the interaction between superior and subordinate agents.

Another hierarchical multi-agent system has three layers of controllers (France and Ghorbani, 2003). Local traffic agents, including intersections and independent vehicles, are located in the first

layer. By monitoring lane traffic flow, the agent can provide a local optimum traffic signal control plan. Because this plan may conflict with the optimal strategy of other agents, the control agents in the second layer need to monitor and coordinate different local traffic agents. Under the coordination of control agents, the local optimal plan can achieve global optimisation through slight modification. To achieve this goal, local agents need to communicate and store relevant information on the third layer agents. However, this article does not provide detailed information about this approach, including how to build this three-layers system and what information needs to be communicated between agents. Another problem is that this article does not provide the standard for global optimisation. There are multiple controller agents in the second layer so that they may come up with different global optimisation schemes.

2.4 Summary

In conclusion, the multi-agent system has achieved a lot of results in the traffic area. To improve the efficiency of the system containing a large number of agents, group control has received more and more attention. The simulation system can simulate the behaviour of human groups, but researchers lack supervision and control of individuals. Although the hierarchical system optimises the interaction between agents, it needs more progress in the global optimisation strategy. And there is a gap in how to utilise MAS to generate test cases more efficiently.

Though the work of literature review, this paper inspired by the hierarchical control mechanism and started with a straight two-way lane with pedestrians crossing as the first test scenario. Then, to achieve the goal of controlling pedestrian agents' movement, this paper proposes a new control structure analyse the performance of several different movement modes. Finally, drawing on the previous researches, to improve the visualisation of the multi-agent system, this article proposes a method enabling the migration of the system from Jason platform to Carla simulator.

In the next chapter, the multi-agent model and related experiments will be introduced in detail.

3 Agent-Based System Design

Multi-agent method is a well-known approach to build distributed systems. In such systems, numerous agents work together and receive information from the environment. Then, they select a suitable plan to accomplish their goals and update the environment. The interaction of numerous agents may make the system operation process very complicated, so a clear and elegant system structure is needed to ensure the orderly operation of agents.

The cell automata (Wolfram, 1986) is the basement of the multi-agent system. Based on this achievement, vehicles and pedestrians are modelled as independent agents with some simple beliefs and plans in the simulator (Yamamoto, Kokubo, and Nishinari, 2007). These agents can perform fundamental interactions, such as the exchange of information and location. Since automatic driving requires a large amount of data to test the operation in the same scenario, this paper aims to generate the valid tests by designing am agent-based test system. In this section, this paper demonstrates how to create a test generation system for autonomous driving based on the multi-agent system. The structure of this part will follow from the whole construction of the multi-agent system to the specific methods.

3.1 MAS Structure

Based on the purpose of establishing the agent-based test system for autonomous driving, this paper designs a test scenario, including a straight road and several pedestrians crossing. This system can provide test data for subsequent autonomous driving research by collecting changes in the position of vehicles and pedestrians when the valid tests are generated. To improve its efficiency, this paper designs three pedestrian movement methods to compare the performance of system test generation.

The main construction work of the system is on Jason platform which is an implementation of the AgentSpeak language for building MAS. Unlike the existing multi-agent system described in (Jin et al., 2013) that vehicles can only interact with the control centre, the interaction of agents can be wholly independent and non-interfering. The next command will only be executed after the interaction is completed and the state of agents are updated. The main components of the system include two vehicle

agents, six pedestrian agents and the road environment. Unlike the hierarchical structure proposed in previous studies (Roozemond, 2001). Agents of the same type are equal, and each agent can interact with other agents. The environment includes the parameters to build the test scenario, the customised plan required by agents, and the evaluation subsystem. The interaction among these components is shown in the figure below.

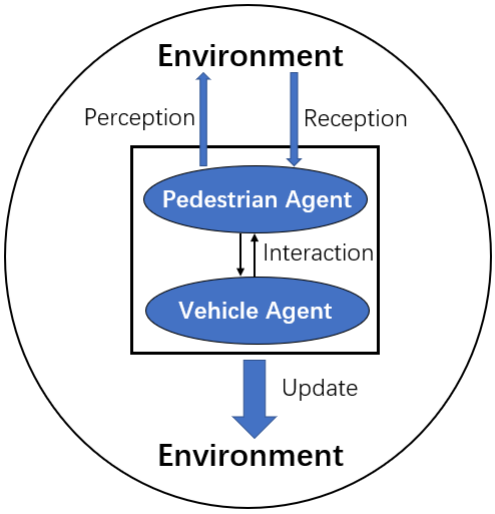

Fig.4. The interaction process of the multi-agent system

The vehicle agents and pedestrian agents are firstly initialised with their own beliefs and goals according to pre-set rules. They then may interact with the environment by perceiving information, such as locating their position in the environment. Then, when a pedestrian agent moves to the vehicle interaction area, the vehicle agent can decide whether to brake or not based on the distance from the pedestrian. The pedestrian agent has relative behaviours according to its movement mode. To achieve this interaction, vehicles need to receive information about the location and the direction of pedestrians around in real-time. If the pedestrian agent generates an valid test or the vehicle agent arrives at the end of the road, the experiment is over. After saving the data from this experiment, the agents will be initialised again.

The same type of agents may invoke the same plan which contains some variables, so it is necessary to ensure that the variables of different agents can only use their data. As shown in figure 5, some previous researches (France and Ghorbani, 2003) proposed to use agent name as the input variable to invoke variables, such as the location and the state, but this approach caused variables to be rewritten by each agent so that other agents lose their information. Therefore, we propose a method that all variables

needed by each agent are defined separately and stored in an array.

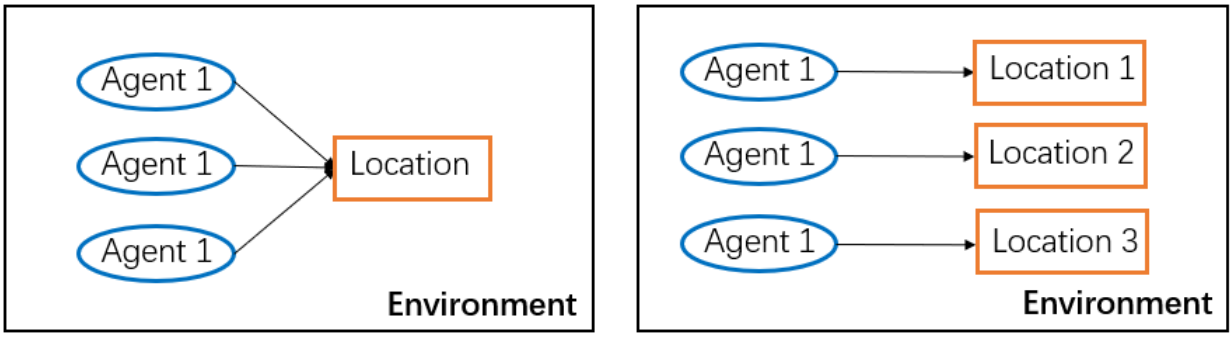

Fig.5. The agent independent control

This method makes the system structure clearer, which is convenient for developers to modify each agent, and avoid agent running error, especially memory usage chaos. Because the plans of the agent are executed alternately as the system runs, the plan to read into memory needs to be periodically suspended to wait for the rest of the agents to complete their plans. The former approach is prone to errors in practical testing, so we propose the latter approach.

Besides using separate interaction mechanism at the system level, pedestrian agents gain movement modes from the environment and initialise them into their independent plan. The pedestrian agents have three movement modes and separate scoring systems to evaluate the performance of each mode.

In the next part, the agent model will be introduced in detail. To make the part easy to follow, the necessary knowledge about the agent model is explained first.

3.2 Agent Design

By customising the BDI content of agents respectively, the agents can become individuals who can perform independent behaviours in the environment. Each agent has their own initial beliefs, rules and goals, which can be seen as their initial state. They can update beliefs according to their rules and interaction with other agents. Via these parameters, agents can understand who they are and what to do next. Then they can look for and execute the best suitable plan to deal with it.

The plan of the agent usually consists of three parts, including triggering events, context and body.

$$Triggering\ event : context -> body \tag{3.1}$$

The triggering event represents the specific goal needs to be achieved. The goal can be the change of its state, such as the movement, and the change of other agents in the environment, such as making the pedestrian agents stop. Generally, in the structure of plans, the next event is set on the last line, so after each plan is executed, it will specify the next triggering event. In the second part, the context indicates the conditions that need to be met for the event to be executed. The main content is the beliefs that the agent needs to have. As mentioned above, the beliefs are gained from their rules and the other agents. The third part contains details on how to implement the plan. It can execute the already integrated commands as well as get them from the environment through the API interface of Jason platform. It is worth noting that events with the same name can have multiple versions. The same event can have different conditions and different plans for different situations. For instance, as shown in figure 6, the same events have different plans because of the presence of the belief "safeVP". The vehicle agent will only move if both beliefs exist. If the pedestrian agent makes the vehicle agent delete the belief "safeVP", the vehicle will stop.

```
+!run : safeVP & moving
   <-
   move_v(X+5);
   !run.
+!run : not safeVP & moving
   <-
   stop_v;
   !run.
```

Fig.6. The structure of agent plan

```
/* Initial beliefs and rules */
Belief  vehicle(1,0).
        vehicle2(1,0).
        pedestrian(1,0,0).
        pedestrian(2,60).
/* Initial goals */
Goal    !prepare_startVehicle.
/* Plans */
Plan    +!prepare_startVehicle:true <-
        ?vehicle(1,X); locate(X);
        -vehicle(1,X); !start_vehicle.
```

Fig.7. The initial part of structure of the vehicle agent

In this test system, there are two vehicle agents and six pedestrian agents whose number can be easily expanded. As shown in figure 7, the vehicle agent can perceive the position information of all

pedestrians and other vehicles at the beginning of the experiment. Following operation stream, the plan to prepare to start vehicle make the vehicle agent recheck its position, then perform the "locate(X)" function to update its location and belief in the environment. The specific content of this function can be gained via the interaction with the environment. Besides the plan to prepare to start the car, vehicle agents have a lot of plans, including starting, acceleration, deceleration and parking, to simulate the running of a car.

```
/* Initial beliefs and rules */
vehicle(1000,130).
vehicle2(0,250).
safeVP.
crossSafe :- safeVP & crossDesire.
/* Initial goals */
!prepare_start_pedestrian.
/* Plans */
+!prepare_start_pedestrian:true
<-
   locatep(X,Y);
   !start_pedestrian.
+!start_pedestrian: safeVP & not movingp
   <-
   start_p;
   !walk.
```

Fig.8. The initial part of structure of the pedestrian agent

As shown in figure 8, the pedestrian agents only have the initial position of vehicles. Since the purpose of this article is to study the influence of pedestrian agents on the test generation, pedestrian agents do not perceive the position information of other pedestrian agents and their movement does not interfere with each other. If they maintain a safe distance from vehicles, they can keep the belief called "safeVP" which is the critical context to perform the crossing plan. If one pedestrian agent wants to cross the road under safety, it will add the belief called "crossDesire", thereby adding the "crossSafe" belief according to the rule to activate the event of crossing the road. As mentioned above, since the number of pedestrian agents may be a lot, to prevent confusion in the agent calling process, the system can create separate databases for each agent and determine the ID of the agent by retrieving the keywords to control the process of each agent in an orderly manner. The yellow fields that appear in figure 8 are the specific plans which include some variables to control the movement. The agent needs to enter its identity to indicate identity when calling these programs and can only access to own related

variables.

3.3 Environment Design

The environment is the space where agents can perceive information and interact with each other agents (Bordini and Hübner, 2005). The purpose of this system is to generate valid tests in the case of vehicles travelling in a straight line, so the test scenario created in this article is a two-lane road with two pavement area. Two vehicles can only move on the road. Several pedestrians move freely on the pavement, and they may sometimes cross the road to the other side. At the beginning of the experiment, the two cars start from the starting points on both sides and drive in opposite directions. When they arrive at the end of the road, all agents are initialised because pedestrians cannot reach the front of the vehicle and generate a valid test.

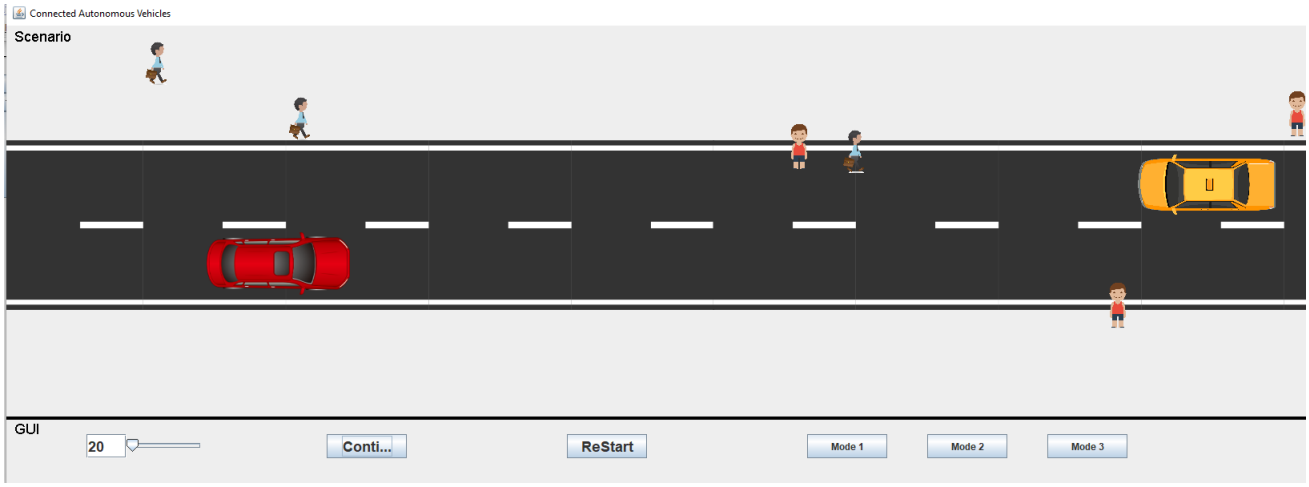

Fig.9. The test system scenario

As shown in figure 9, according to the relevant road traffic law, the entire scene area is 99*24 meters, including two 6-metre-wide roads and four 3-metre-wide pavements. To conveniently and accurately record the location of the agent, the entire map consists of grids, and the size of the grid is 100 square centimetres. Thus, the number of grids is 990*240. Because this system aims to find valid test cases when pedestrian agents cross the road, it is not a case that fits our needs if they stay at the road and wait for the vehicle. To punish this situation, each square contains different scores. As shown in figure 10, the squares on the road display minus seven points and the score on the pavement shows minus one, which means using different living cost to punish the pedestrian agent staying on the road. For the sake

of convenience, all the grids are not listed in the figure.

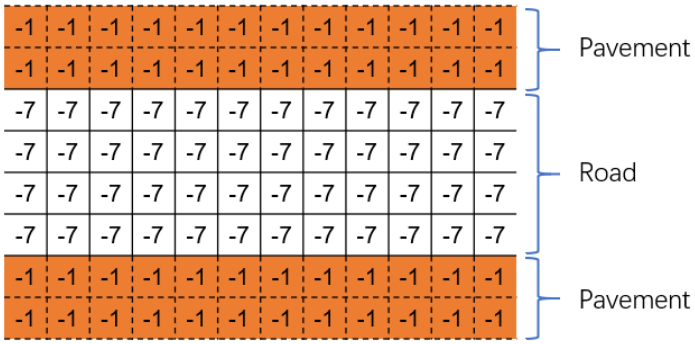

Fig.10. The grid test scenario

The speed of vehicles and pedestrians is constant. To make the simulation system meet the real situation, the max speed of vehicles is set to 20 mph which is about 9 m/s, and the pedestrian speed is set at one-sixth of the speed of the car, that is 1.5 m/s. To conduct experiments more conveniently, these two parameters can be easily changed in the toolbar below. In addition to speed adjustment, the movement mode of pedestrian agents can also be easily controlled with the buttons below.

This article proposes three movement modes, namely Random Movement, Random Behaviour and Proximity movement. Random Movement means that pedestrian agents have five choices with the same probability at each step: up and down, left and right and stay in the same place. Pedestrian agents with this movement modes are more ego, which means they do not worry about the danger of staying on the road and also does not care about the position of the car. Although the vehicle agent can stop before a certain distance before the pedestrian, the pedestrian agent might still move toward the vehicle or stay at the same place. It is worth noting that at the initial speed, the time it takes the vehicle to reach the end of the road is 11 seconds. To prevent pedestrians with the first mode from staying on the road resulting in that the vehicle cannot continue to move forward, the maximum experiment time is set to 30 seconds. If the time is more than 30 seconds, the agents will be reinitialised.

Random Behaviour means that the pedestrian agent can either walk on the pavement alongside the road or cross the road according to its random position. Pedestrian agents appear randomly on the pavement area when the system starts running. The pavement areas on both sides of the road are divided into three sections respectively, in which the pedestrians are spawn in the middle two areas perform the crossing road action, and the rest perform the forward motion. Similar to the first pattern, pedestrian agents are not influenced by the location of the vehicle agent.

The third behaviour type allows pedestrian agents to perceive the location of vehicle agents and to select plans based on the distance. Whilst walking along the pavement, the pedestrian agents may randomly generate the belief that they want to cross the road. They then calculate the time t_1 that they get to the centre of the road and the time t_2 that the car to get there. If $t_1 = t_2$, the pedestrian agent executes the crossing road plan and gain a test case. The operational process is shown in the following figure.

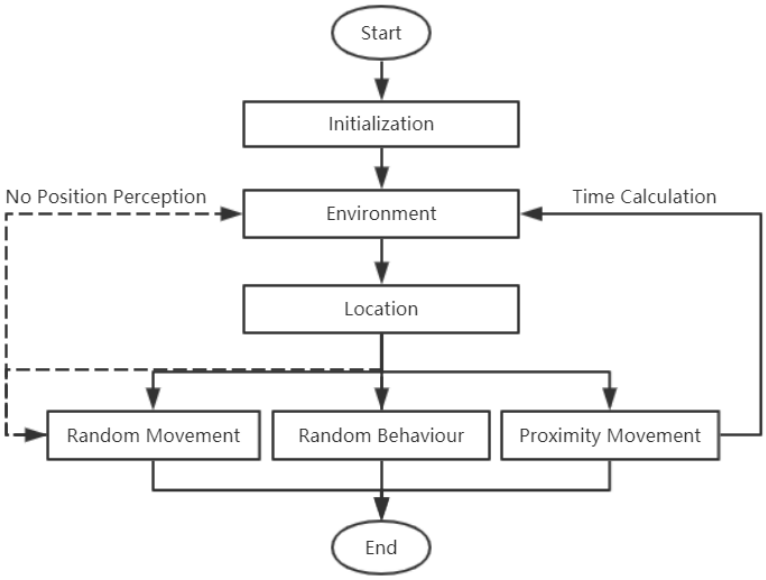

Fig.11. Pedestrian Agents Movement Selection Process

As shown in figure 11, after initialisation, pedestrian agents first locate themselves in the environment, then they will perform the corresponding movement according to the manually selected mode. If choosing the first two modes, they will not perceive the location of the vehicle agents. Pedestrian agents with the Proximity Movement mode perceive the position of vehicles and calculate the time in real-time to prepare for generating the test. The behaviour type was chosen before the experiment started. The experiment can be repeated automatically when the vehicle safely arrives at the end of the road or gain a test case.

Table 1 Example of some test results

Number	ID	Score	Start Px	Start Py	End Px	End Py	Assertion	Time
1	1	726	10	21	234	190	FALSE	5.08
1	2	542	120	35	501	20	FALSE	5.08
1	3	914	235	185	260	230	TRUE	5.08
1	4	376	542	41	506	151	FALSE	5.08
1	5	322	766	220	698	132	FALSE	5.08
1	6	764	806	192	783	221	FALSE	5.08
2	1	206	202	184	251	162	FALSE	7.35

3.4 Data Collection

To evaluate the performance of the movement mode and record the specific situation when an assertion occurs. The data recorded includes the following items.

$$Data =< n, i, score, sp, ep, a, t > \tag{3.2}$$

The n indicates the number of entries in the data. Because the system contains multiple pedestrian agents, its value is the number of experiments multiplied by the number of agents. i shows the identity of the pedestrian agents. sp and ep record the start coordinates and the end coordinates of the pedestrian agent. a indicates whether the pedestrian generates a valid test case or not. t represents the simulation time of one experiment. At the end of each experiment, the data is recorded and written to the file, which is used to evaluate pedestrian models.

Because the vehicle agents with simple plans only drive on the road, the action of the pedestrian agent determines the generation of test cases. Every pedestrian agent has its scoring system, and the original score of each pedestrian is 1000 points at the beginning of the experiment. As mentioned above, pedestrian agents are minus one point when moving on the pavement, but seven points are deducted for each step on the road. Also, to find a mode that makes it easier to generate test cases, if the agent generates a required test, it will add 350 points which can make up for the points deducted because of moving on the road.

When an experiment is finished, the data is collected and imported into the database file separately. Their starting and final positions are recorded for later visualisation in Carla simulator. In the last

column of Table 1, the *Number* indicates the number of experiment so the agents in each experiment have the same value. *ID* indicates the identity of pedestrian agents. Besides recording the score and the position of agents, the data records whether the agent generates the required test case. To simplify the data structure, the duration of the same simulation experiment is scored in all pedestrian agents.

To objectively evaluate the performance of each behaviour type, the system run and record data 10,000 times for three types of data respectively, and take the average value to reduce the influence of abnormal situations. The formula for calculating the mean value is as follows. $value(i)$ indicates the value of each piece of data, and the $min(i)$ and $max(i)$ represent the extreme part of value. N is the total number of experiments. To reduce the impact of extreme situations on overall performance, after summing a column of values, the partial minimum value and the partial maximum value are subtracted. Finally, we gain the mean value μ of three behaviour types. After gaining the mean value, We can get the standard deviation from formula 3.4.

$$\mu = \frac{\sum_{i=1}^{10000} value\,(i) - \sum_{i=1}^{1000} min\,(i) - \sum_{i=1}^{1000} max\,(i)}{N} \tag{3.3}$$

$$s_N = \sqrt{\frac{\sum_{i=1}^{N} (value\,(i) - \mu)^2}{N}} \tag{3.4}$$

To accurately evaluate the performance of each pattern, we will evaluate from three aspects, including score, duration time, and test generation efficiency. All the indicators of the three aspects are derived from the above formula.

3.5 Visual Simulation Scenario Construction

Since Jason platform does not provide the necessary support for the 3D environment, to develop more realistic simulated 3D scenes, we transfer the newly developed agent behaviour from a grid world in Jason platform to a 3D simulator, Carla.

Carla is a vehicle driving simulation platform based on the Unreal Engine. As shown in figure 12, it has sophisticated 3D scene models and a list of developing tools which can easily modify the entire scenario. It also integrates a lot of commands to manage the vehicle agent and the pedestrian agent. We designed a new map with a straight line and two loops to keep the vehicle going, which is shown in figure 13. We recorded the original program and extended the navigation mesh of the pedestrian agent

to allow them can walk on the road, which is not possible on the original map. Finally, by importing the data into the map to make the agents can obtain the location information collected from the test generation experiment, we created a more vivid and graphic 3d multi-agent model.

Fig.12. Carla simulator interface

Since the development language of Carla is Python, and Jason is based on Java, we need to consider changing the structure of the data to address the differences between the two operational processes. As shown in figure 14, the agent position data collected from the experiment in the Jason is input Carla to create the position array. Carla program contains open loops and closed loops, which means that part of the program is executed once, and the other part is executed cyclically. Therefore, we set the selection function to put the starting position, ending position and next starting position of the agent into the independent array to ensure the data can be invoked in an orderly manner.

3.6 Summary

This article aims to find a method that can generate valid tests faster in the multi-agent system, so we created a test scenario and collected operational data of three movement modes. Then, though calculating the value of data, we will analyse the performance of three movement modes in the next section.

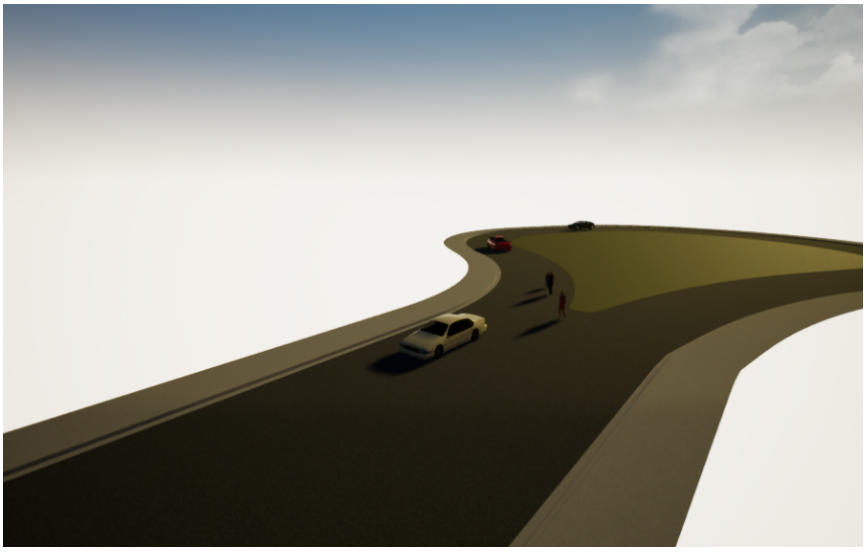

Fig.13. Test generation system based on Carla

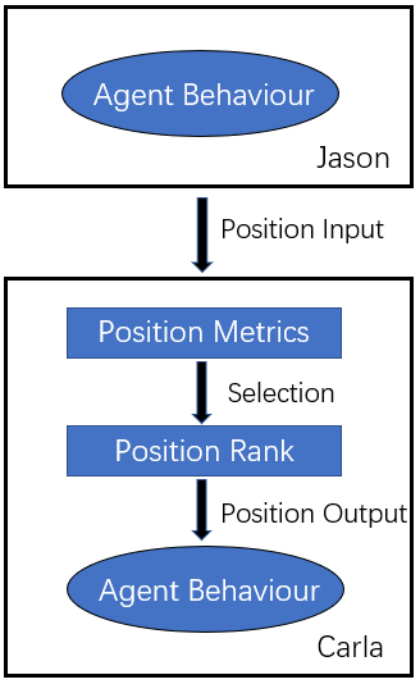

Fig.14. The process of transferring agent behaviour

4 Results & Discussion

In this section, we will introduce the data collected in the experiment and analyse the performance of three movement modes. The evaluation consists of three items: the score, the experiment duration time and the total number of valid test cases. We will also horizontally compare the value of three modes when the number of pedestrian agents is six and then vertically compare the performance of the same movement mode having different quantities of pedestrian agents.

4.1 Score Values

It is important to ensure that the test generation should happen when pedestrian agents cross the road instead of staying on the road to wait for the vehicle agent. This ensures the position data can be used to provide useful information for the following autonomous driving research. As mentioned above, we selected 8000 sets of data to average the score of each movement mode. The final average scores of the three behaviour types are shown below.

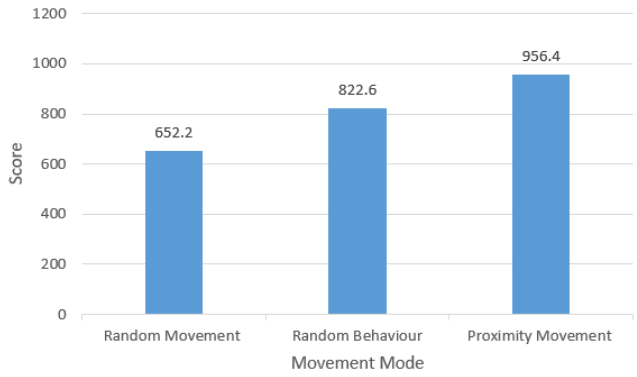

Fig.15. The mean score of three modes

Random Movement scores the lowest with 652.2 points. Random Behaviour is 822.6 points, which is about 27 percent higher than the first one. The Proximity Movement scores the highest with 956.4 points.

In the actual situation, the Random Movement pattern may make the pedestrian agent can only

move back and forth in a small area, so some agents who stay on the road are continuously subtracted points until the end of the experiment, which results in their low scores and the inability to generate the test case that meets the requirements mentioned above. The Random Behaviour Mode allows pedestrian agents to randomly choose the behaviour of going straight and crossing the road. In this case, although crossing the road in front of the car can generate valid tests, the probability of pedestrians and cars meeting is not very high due to the large size of the map, which results in that pedestrian agents can often cross the road safely and no test can be generated. In the Proximity Movement mode, the pedestrian agent can move to the right place according to the timing when it meets the vehicle. Hence, most experiments can end with generating a test case, which yields the conclusion that the Proximity Movement perform is better than Random Mode in the generation of valid tests.

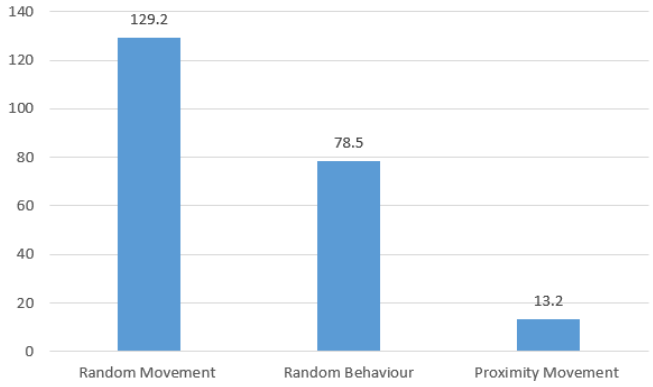

Fig.16. The standard deviation of score

The standard deviation is used to evaluate the dispersion of a group of data over its mean value. To compare the stability of the performance of the three modes, we calculated the standard deviation value of score when the number of agents is six. As shown in figure 16, the standard deviation value of Random Movement is the highest, 129.2. The value of Random Behaviour is a litter lower than the first mode, which is 78.5. The value of Proximity Movement is 13.2 which is the lowest value.

As mentioned above, pedestrian agents with Random Movement mode often hang out in a small area, so the pedestrian agents on the pavement keep their original scores and the score of the agents on the road are subtracted all time, which results in a considerable difference in their values and indicates that many agents in this mode cannot generate test. The value of Random Behaviour is still a little high because only the agents in the middle of the pavement will cross the road, resulting in lower scores. The other agents only walk on the pavement. If the pedestrian agent reaches the pavement on the other side of the road safely and does not generate a test, the difference can be as high as 400 points. Proximity

Movement has the lowest value, which shows that the ability of agents to generate tests is close.

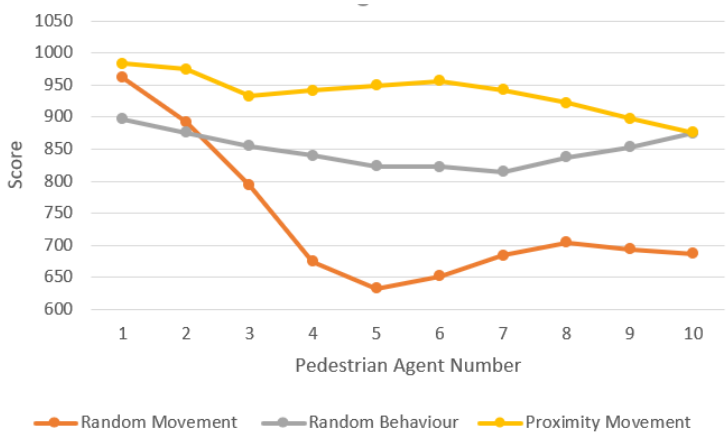

Fig.17. Mean Score of different agent numbers

To more objectively evaluate the performance of the three modes on the score, we also compared the performance of the three modes with different quantities of agents. As shown in figure 17, the three modes have high scores when the number of agents is small. As the number increases, the score of Random Movement decreases quickly and finally stabilises at around 700 points. Also, as the number of agents increases, the score of Random Behaviour only decreases slightly. After the number of agents is greater than 7, the score of Random Behaviour increases steadily with the increase in the number of agents and is eventually almost the same as the score of Proximity Movement. Finally, the score of Proximity Movement rises steadily when the number of agents is greater than three and reaches the maximum when the number is six.

The reason why the score of three modes is the highest when there is only one pedestrian agent is that agents using the random behaviour have a high probability of not moving to the road, so their scores are not subtracted. With the number of agents growing, some agents may cross on the road without generating tests which result in the lower scores. It is worth noting that the value of Random Behaviour is equal to the value of Proximity Movements when the number of agents is 10. It is because in the random behaviour mode, the number of agents crossing the road increases, increasing the probability of encountering the vehicle agent. If an agent generates the required test, all agents will be initialised immediately, so if the pedestrian agent can meet the vehicle agent as soon as possible, all agents will not have much time to be scored down, which leads to the increase of their scores. However, the system containing a large number of agents requires researchers to spend a lot of time developing. Therefore, considering the workload of developers and the value of standard deviation, Proximity is more suitable

for test generation.

In conclusion, although the scores of all movement modes are highest when the number of agents is 1, the system in this situation cannot generate test efficiently. Besides, although adding more agents in MAS is cheaper than the traditional platform, in the actual situation, the increase in the number of agents may significantly still increasing the workload of developers. Therefore, because Proximity Movement has the highest mean score and the lowest standard deviation value, Proximity Movement mode has the best performance in test generation.

4.2 Simulation duration time

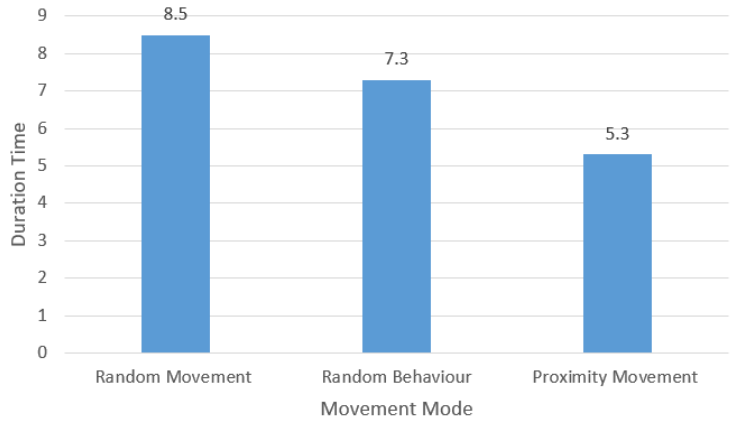

Fig.18. The mean duration of three modes

The average duration indicates the time spent in an experiment, which represents the efficiency of the test generation. We use the same method that the time is averaged from the selected 8000 samples, as calculating the mean score. As shown in figure 18, Random Movement has the longest time, 8.5 seconds, and Random Behaviour takes 7.2 seconds. Proximity Movement has the shortest time which is 5.3 seconds.

Because of the reason mentioned above that agents with Random Movement may only move within a small area, the agents that appear far away from the road may not be able to move quickly to the road so that they are not useful to generate tests. Also, although some agents with Random Behaviour mode can cross the road freely, they may not be able to meet vehicles in most cases because the map is very large relative to the agent. Thus, its time is only a little shorter than the first mode. Lastly, Proximity Movement mode allows agents to keep getting close to the vehicle, so pedestrian agents on

the pavement beside the road can move to the road on time following the rules to generate a test. To further reduce the time, all agents appear in the middle position of the pavements to avoid the agents appearing at the edge of the map are unable to move to the front of the vehicle agent because of the speed difference.

If the pedestrian agent appears in the rear area of the vehicle, it cannot meet the vehicle because its speed is not as fast as the vehicle agent, so the pedestrian agents with the third mode only appear in the middle of the pavement at the start of the experiment.

The time of Proximity Movement is slightly less than half the time it takes for the vehicle agent to leave the map, which indicates that most required tests are generated when the vehicle agent is in the middle of the road. The data obtained from this case is more comprehensive and more convenient for later migration to the new platform because the position of pedestrian agents is not limited by the edge of the map. Therefore, the agent using Proximity Movement mode generates tests more efficiently than the other two modes.

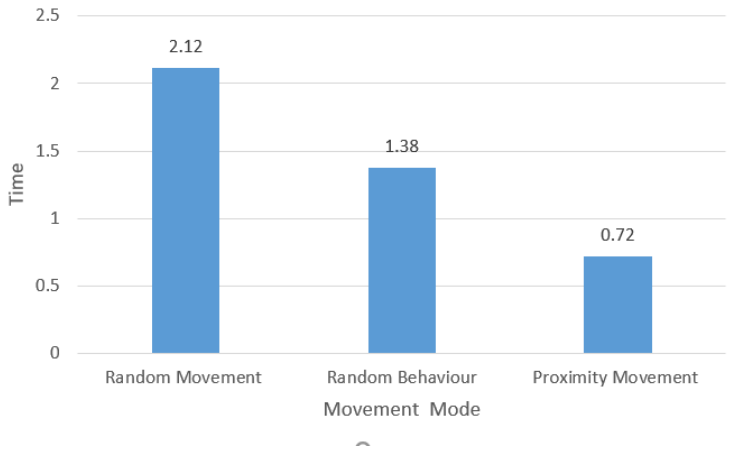

Fig.19. The standard deviation of time

This figure shows the standard deviations for the duration of the three patterns. The Random Movement is the highest, 2.12. Random Behaviour is lower than in the first mode, which is 1.38. Proximity Movement is the lowest, which is 0.72.

Combining the data in figure 18, the data indicates that the time spent in the first mode is between 6.4 seconds and 10.6 seconds, which shows the results are not concentrated. This is because the number of agents is not enough, so its time depends on where they appear. If the initial location of the agent is near the road, its time is likely to be short. Conversely, if it is far away from the road, the experiment may not end until the vehicle arrives at the destination. In the second mode, the value is a little lower than

the first mode, which indicates that Random Behaviour does not cause the agent to linger. Proximity Movement mode has the lowest value, which indicates that this rule of executing the plan according to the environment can make the agent go to the vehicle and generate a test as soon as possible. Therefore, Only Proximity Movement mode can guarantee the stability of the system operation.

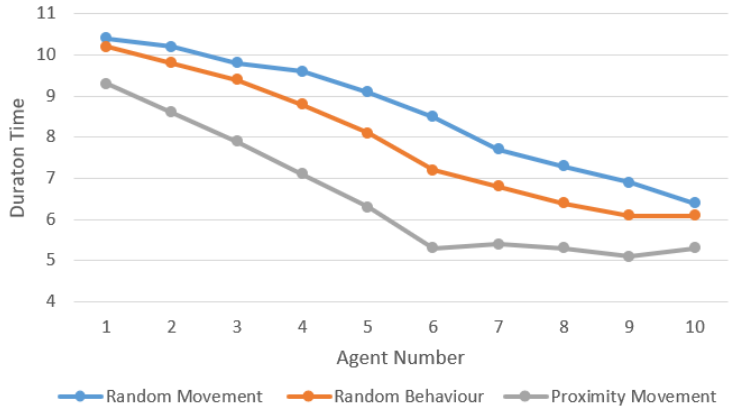

Fig.20. Duration time of different agent numbers

As shown in figure 20, when the number of agents is equal to one, the time of the first two modes is close to 11 seconds, which indicates that agent cannot generate test in most experiments. As the number of agents increases, the duration of all three modes decreases. The time of Proximity Movement is always the lowest and remains stable when the number of agents is higher than six.

As the number of agents growing, there are more agents moving to the road, and the probability of generating the required tests and ending the experiment in advance is also higher. Due to the uncertainty of agent movement in the first mode, the duration declines more slowly than the second pattern. Because the third mode agents have a good performance in valid test generation, which is proved in the previous part, an agent can quickly generate test as long as it appears not too far away from the road. Hence, the time of Proximity Movement remains stable when the number of agents exceeds five.

Although according to the trend, as the number of agents increases, the times of three modes can eventually converge, the too complicated structure requires developers do a lot of extra work and is not conducive to the maintenance of the system. Besides, the standard deviation of the first movement mode is higher than Proximity Movement mode, which demonstrates that the third mode is more stable than the other modes. Thus, Proximity Movement is the best method in the time comparison.

4.3 Test Accuracy

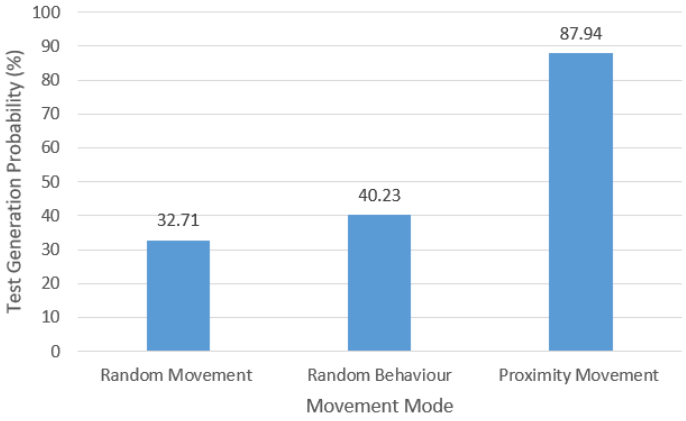

Fig.21. The probability of test generation of three modes

The test accuracy is defined as the ratio of valid test cases as a percentage of all tests generated. In these experiments, 10'000 tests were performed for each behaviour type. As shown in figure 21, we gained Random Movement only has 32.71 percent to generate a valid test case in each experiment and Random Behaviour has 40.23 percent. The Proximity Movement mode has the highest probability, which is 87.94 percent.

Not only the value of Random Movement is slightly lower than Random Behaviour, but also most of them are obtained from waiting for the vehicle agent on the road. Hence, these tests generating represent non-natural pedestrian behaviour are invalid. Furthermore, the probability of Random Behaviour is 40.23 percent. Although many of these cases are valid and effective, the useless data takes up too much space, and the test generation efficiency is too slow to meet the requirements of autonomous driving. As for the third mode, it can be seen from figure 21 that the Proximity Movement has a much higher valid test generation efficiency than the other two modes. Its valid test generation efficiency is close to 88 percent, and it covers almost all possible situations. From the results, we conclude that the agent can greatly improve the efficiency by simply following some simple rules compared with the random test.

Similar to the order of figure 22, Proximity has the highest value, which is 4.4, and the value of Random Behaviour is lower which is 3.7. The standard deviation of Random Behaviour mode is the lowest which is 3.6.

The result shows that the data of Random Movement has the lowest degree of dispersion, which seems to indicate that its performance of generating valid tests is the most stable. However, its efficiency

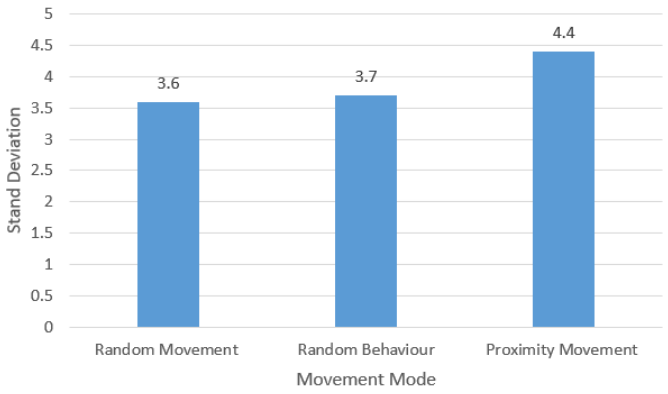

Fig.22. The standard deviation of the number of the valid tests

of test generation is less than half of the value of Proximity Movement mode. The significant difference between the two modes in the test generation numbers indicates that Random Movement is not as good as Proximity Movement. Therefore, this result cannot be used as an indicator to evaluate the three modes.

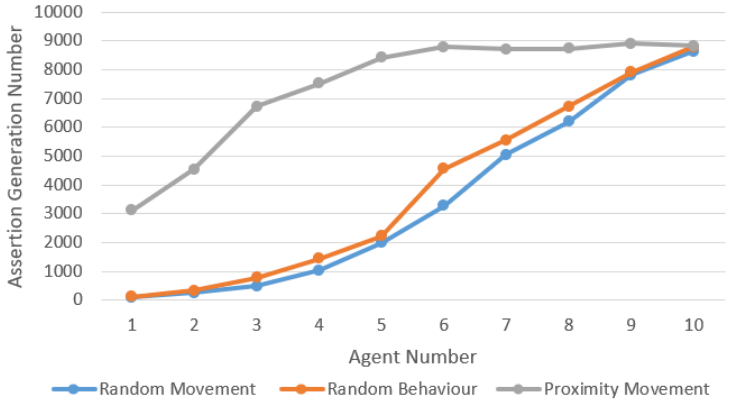

Fig.23. Total number of valid tests of different agent numbers

As seen in figure 23, the number of tests of three modes increases with the number of agents growing. The values of the first two modes are always very close. They are difficult to generate the qualified tests when the number of agents is equal to one, but they grow very fast, especially the value of Random Movement can reach 8600 when having ten pedestrian agents. Proximity Movement always has the highest value. When the number of agents is greater than six, the value remains around 8900.

When the number of agents is not much, the performance of the agent of the first two modes depends on its location, so it is difficult or impossible to generate valid test cases. Only when their number increases, the probability that they can appear in the right place can increase. The reason

why the value of Proximity Movement remains at around 9000 is that in order to imitate the natural behaviour of human beings, the agent randomly generates the belief that they want to cross the road to control its behaviour, so in some cases, all the agents do not have this belief.

In conclusion, although the values of the first two modes are not much different, as mentioned earlier, Random Movement may generate useless test cases. Hence, Random Behaviour has a better performance than the first mode. Because the agent with Proximity Movement mode can automatically move toward the meeting point, even if there is only one pedestrian agent, it still has a 30 percent probability of generating test cases. For cases where no valid test is generated, we suspect that the pedestrian is too far away from the car or does not have the belief in crossing the road.

4.4 Summary

Based on the data collected from more than 30,000 experiments, we analysed and evaluated the performance of three modes from four parameters. Except for the standard deviation of test generation numbers, other results support the assumption that the rule-based method can generate valid tests faster than the random pattern. We also have learned that it is possible to draw contradictory conclusions from a single data, and multiple data should be considered in the evaluation.

In the next section, we will summarise all the work by re-describing the main content of the paper. Then, we will introduce some shortcomings of the system designed in this paper and the future research directions.

5 Conclusion and Future Work

MAS has been widely used in the field of transportation. The future intelligent traffic systems (ITS) could be composed of intelligent and independent MAS. These agents keep operating between traffic control centres, road intersections, highways, and streets. To realise the ultimate intelligent transportation system, they acquire accurate information and make the correct decisions at the right time through the Internet, wireless network and self-organising network (Wang et al., 2005). In the autonomous driving field, the vehicle needs to handle any possible assertion. It is very challenging to collect this data from real road experiments in the development process due to the complexity and incorruptibility of the real environment. Thus, we build a test generation system based on MAS to collect the relevant data.

The generation of all test cases for autonomous driving is always an intractable problem, so more targeted tests need to be intelligently found. To meet safety requirements, the autonomous vehicle should respond appropriately to pedestrians with various action routes ahead, so as many test cases as possible should be generated to test the performance of the vehicle. The data of test cases should include the route and time of all agents to reflect the specific situation when the assertion occurs. To gain the test cases, this paper design an agent-based test generation system and proposes a hypothesis that the pedestrian who has the natural behaviours can generate useful valid tests faster than random patterns. Based on this hypothesis, we designed three movement modes for pedestrian agents and analysed their performance from four aspects: score, time, number of generations and number of agents. Besides the movement mode design, to avoid errors in the invocation process because the same kind of agent may compete for resources, we also proposed a control structure to optimise the process of invoking agent.

Finally, we proposed an approach to migrate this system based on Jason platform to the Carla platform to get better three-dimensional visualisation. The position changes of all agents can display on the 3D map of Carla via data obtained.

5.1 Thesis Summary

Focusing on the test generation for autonomous driving, this paper introduces the work from the following perspectives.

Firstly, in the Introduction section, to explain the work of this paper, through a brief introduction of previous research findings, we demonstrated the background knowledge of MAS and its development in the traffic area. In the first part of this section, we firstly introduced BDI structure and Agent Speak language. We explain the role and operation logic of the three parts of an agent in detail and analyse the principle that an agent can independently choose the plan to achieve its goal. Then in the second part, we introduced the concept of Agent-oriented programming by analysing advantages and disadvantages of two multi-agent systems and proposed a hypothesis that the pedestrian agent with natural behaviour can gain a better test generation performance. In the third scenario, we proposed the test scenario that is a straight road with several pedestrian crossing. All vehicles and pedestrians are independent agents. Because we focus on studying the influence of pedestrian movement on test generation, the vehicle agent can only run on the road. For migration to a 3D simulator, the system needs to record the data about the position and several performance parameters of pedestrian agents.

In the development process, the data record is messy at first, the same piece of data is often written many times, and some data are missing. It is because multiple data read and write commands cause the out-of-order memory reads. To avoid the data error, we set up separate variables for each agent and simplify the lines of the file output statement. At the same time, we put the execution commands of the data in the proper position of the program. For example, we moved the data, such as the number of experiments and the initial position of the agent, to the position at the behind of the system initialisation, which avoids data being written to the wrong location in the file. Finally, regarding the migration approach, we write the starting and stopping coordinates of each agent as well as the starting coordinates of the next experiment into an array and input into agents of the simulator. Since Carla pedestrian control framework has the open-loop and the closed-loop control mechanism. Only in this way can it avoid program errors.

Next, taking the hypothesis about the test generation efficiency, three kinds of movement patterns were tested, and the data obtained were analysed. The movement mode includes Random Movement, Random Behaviour and Proximity Movement. After performing vertical analyses and horizontal analyses on the three modes, we reasoned some conclusion:

- The assumption in this paper is correct. Proximity Movement mode is easier to generate tests than other random patterns.
- Proximity Movement mode takes the least time in experiments.
- The test generation performance of Proximity Movement mode stays stable when the agent is greater than six.

Few researchers studied the application of multi-agent systems in test generation in the autonomous driving field. It is always the primary task of an automatic driving simulator to provide test cases that meet the requirements of quality and quantity. Therefore, the main work of this paper is to propose a pedestrian design approach to improve the test generation efficiency and an agent-based system structure to avoid the confusion of the system running process caused by the agent competing for resources.

5.2 Future Work

Few researchers studied the application of multi-agent systems in test generation in the autonomous driving field. It is always the primary task of an automatic driving simulator to provide test cases that meet the requirements of quality and quantity. Therefore, the main work of this paper is to propose a pedestrian design approach to improve the test generation efficiency and an agent-based system structure to avoid the confusion of the system running process caused by the agent competing for resources.

5.2.1 Agent Behaviour Design

We propose three movement-mode in this paper and prove that the rule-based behaviour like Proximity Movement has a good performance in test generation as well as the agents with natural behaviours can generate more realistic test cases. Therefore we will study the pedestrian agent with more natural behaviours. For example, pedestrian agents can choose to wait until the vehicle leaves before crossing the road.

In addition, we will extend this system to test other assertions, such as stopping at a red light.

5.2.2 Test Map Design

To better show the interaction between pedestrians and cars in the straight road, the map of this article only contains a straight road. In future work, we will design many kinds of roads, such as intersections and turning roads. The number of lanes will be increased to study vehicle interaction. Besides, traffic lights and speed bumps can be added to create a more realistic test scenario.

5.2.3 System Structure Design

MAS is a complex system, including numerous independent agents. To avoid the disorder of the system operation, previous researches adopted hierarchical control in the case of numerous agents. However, adopting this structure allows only one of each type of agent to interact with the environment, leading to the unequal information of the same agent. In future work, the optimised interaction approach of the same type of agent should be added to ensure that they can obtain the same information.

Autonomous driving involves machinery, optics, artificial intelligence and other subjects, so we can improve the agent-based test system in many ways. The response of autonomous vehicle agents to environmental changes and the natural behaviour of pedestrians when cars approach are directions worthy of further study. Therefore, only the cooperation of experts in multiple fields can promote the development of the agent-based test generation system.

Appendix

University of the West of England
BRISTOL

Faculty of Environment & Technology
Faculty Research Ethics Committee (FET FREC)

ETHICAL REVIEW CHECKLIST FOR UNDERGRADUATE AND POSTGRADUATE MODULES
Please provide project details and complete the checklist below.

Project Details:

Module name	**MSc Robotics Dissertation**
Module code	UFMED4-60-M
Module leader	Dr. Maryam Atoofi
Project Supervisor	Professor. Kerstin Eder, Dr Greg Chance
Proposed project title	Agent-Based Test Generation for Autonomous Driving

Applicant Details:

Name of Student	Guangyuan Liu
Student Number	18044524
Student's email address	Guangyuan2.Liu@live.uwe.ac.uk

	CHECKLIST QUESTIONS	Y/N	Explanation
1.	Does the proposed project involve **human tissue, human participants, environmental damage, the NHS, or data gathered outside the UK**?	N	*If the answer to this is 'N' then no further checks in the list need to be considered.*
2.	Will participants be clearly asked to give consent to take part in the research and informed about how data collected in the research will be used?		
3.	If they choose, can a participant withdraw at any time (prior to a point of "no return" in the use of their data)? Are they told this?		
4.	Are measures in place to provide confidentiality for participants and ensure secure management and disposal of data collected from them?		
5.	Does the study involve people who are particularly vulnerable or unable to give informed consent (eg, children or people with learning difficulties)?		

FET FREC - Ethical Review Module Checklist v7 September 2014

	CHECKLIST QUESTIONS	Y/N	Explanation
6.	Could your research cause stress, physical or psychological harm to anyone, or environmental damage?		
7.	Could any aspects of the research lead to unethical behaviour by participants or researchers (eg, invasion of privacy, deceit, coercion, fraud, abuse)?		
8.	Does the research involve the NHS or collection or storage of human tissue (includes anything containing human cells, such as saliva and urine)?		

Your explanations should indicate briefly for Qs 2-4 how these requirements will be met, and for Qs 5-8 what the pertinent concerns are.

- If Qs 2-4 are answered Yes (Y) and Qs 5-8 are answered No (N), no further reference to the Research Ethics Committee will be required, unless the research plan changes significantly.
- If any of Qs 5-8 are answered Yes (Y), then approval from the Faculty Research Ethics Committee is required *before* the project can start. Approval can take over a month. Please consult with your supervisor about the process.

Your supervisor must check your responses above *before* you submit this form.

Submit this completed form via the *Assignments* area in Blackboard (or elsewhere if so directed by the module leader or your supervisor).

After you have uploaded, your supervisor will confirm that the checklist above has been correctly completed by marking this form as Passed/100% via the *My Grades* link on the Blackboard Welcome tab.

If your submitted answers indicate that further ethical approval is indeed required, then you must also send this completed form to ResearchEthics@uwe.ac.uk

Guidance is available at http://www1.uwe.ac.uk/research/researchethics.
Further guidance can be obtained via the module leader, in the first instance, or the Department's Faculty Research Ethics Committee representatives, including your department's *AHoD for Research*.

FET FREC - Ethical Review Module Checklist v7 September 2014

References

Bazzan, A. L. and Klügl, F. (2014). A review on agent-based technology for traffic and transportation. *The Knowledge Engineering Review*. 29.3, pp. 375–403.

Bordini, R. H. and Hübner, J. F. (2005). "BDI agent programming in AgentSpeak using Jason". In: *International Workshop on Computational Logic in Multi-Agent Systems*. Springer, pp. 143–164.

Bordini, R. H., Hübner, J. F., and Vieira, R. (2005). Jason and the Golden Fleece of agent-oriented programming. In: *Multi-agent programming*. Springer, pp. 3–37.

Bordini, R. H., Hübner, J. F., and Wooldridge, M. (2007). *Programming multi-agent systems in AgentSpeak using Jason*. Vol. 8. John Wiley & Sons.

Caire, G., Cossentino, M., Negri, A., Poggi, A., and Turci, P. (2004). *Multi-agent systems implementation and testing*. na.

Cammarata, S., McArthur, D., and Steeb, R. (1988). Strategies of cooperation in distributed problem solving. In: *Readings in Distributed Artificial Intelligence*. Elsevier, pp. 102–105.

Coelho, R., Kulesza, U., Staa, A. von, and Lucena, C. (2006). "Unit testing in multi-agent systems using mock agents and aspects". In: *Proceedings of the 2006 international workshop on Software engineering for large-scale multi-agent systems*. ACM, pp. 83–90.

Corkill, D. D. and Lesser, V. R. (1983). *The use of meta-level control for coordination in a distributed problem solving network*. Tech. rep. MASSACHUSETTS UNIV AMHERST DEPT OF COMPUTER and INFORMATION SCIENCE.

Ekinci, E. E., Tiryaki, A. M., Çetin, Ö., and Dikenelli, O. (2008). "Goal-oriented agent testing revisited". In: *International Workshop on Agent-Oriented Software Engineering*. Springer, pp. 173–186.

Ferber, J. and Weiss, G. (1999). *Multi-agent systems: an introduction to distributed artificial intelligence*. Vol. 1. Addison-Wesley Reading.

France, J. and Ghorbani, A. A. (2003). "A multiagent system for optimizing urban traffic". In: *IEEE/WIC International Conference on Intelligent Agent Technology, 2003. IAT 2003.* IEEE, pp. 411–414.

Genesereth, M. R., Fikes, R. E., et al. (1992). Knowledge interchange format-version 3.0: reference manual.

Georgeff, M. (1988). Communication and interaction in multi-agent planning. In: *Readings in distributed artificial intelligence*. Elsevier, pp. 200–204.

Houhamdi, Z. and Athamena, B. (2011). Structured system test suite generation process for multi-agent system. *International Journal on Computer Science and Engineering*. 3.4, pp. 1681–1688.

Hunt, P., Robertson, D., Bretherton, R., and Winton, R. (1981). *SCOOT-a traffic responsive method of coordinating signals*. Tech. rep.

Jennings, N. R., Sycara, K., and Wooldridge, M. (1998). A roadmap of agent research and development. *Autonomous agents and multi-agent systems*. 1.1, pp. 7–38.

Jin, Q., Wu, G., Boriboonsomsin, K., and Barth, M. (2013). "Platoon-based multi-agent intersection management for connected vehicle". In: *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. IEEE, pp. 1462–1467.

Kosonen, I. (2003). Multi-agent fuzzy signal control based on real-time simulation. *Transportation Research Part C: Emerging Technologies*. 11.5, pp. 389–403.

Lakoba, T. I., Kaup, D. J., and Finkelstein, N. M. (2005). Modifications of the Helbing-Molnar-Farkas-Vicsek social force model for pedestrian evolution. *Simulation*. 81.5, pp. 339–352.

Lam, D. N. and Barber, K. S. (2004). "Debugging agent behavior in an implemented agent system". In: *International Workshop on Programming Multi-Agent Systems*. Springer, pp. 104–125.

Lee, J. J., Gu, X., and Liu, S. (2010). Bayesian adaptive randomization designs for targeted agent development. *Clinical Trials*. 7.5, pp. 584–596.

Luk, J., Sims, A., and Lowrie, P. (1982). "SCATS-application and field comparison with a transyt optimised fixed time system". In: *International Conference on Road Traffic Signalling, 1982, London, United Kingdom*. 207.

Madden, N. and Logan, B. (2009). "Modularity and compositionality in Jason". In: *International Workshop on Programming Multi-Agent Systems*. Springer, pp. 237–253.

Maleš, L. and Ribarić, S. (2016). "A model of extended BDI agent with autonomous entities (integrating autonomous entities within BDI agent)". In: *2016 IEEE 8th International Conference on Intelligent Systems (IS)*. IEEE, pp. 205–214.

Mckenzie, F. D., Petty, M. D., Kruszewski, P. A., Gaskins, R. C., Nguyen, Q.-A. H., Seevinck, J., and Weisel, E. W. (2008). Integrating crowd-behavior modeling into military simulation using game technology. *Simulation & Gaming*. 39.1, pp. 10–38.

Núñez, M., Rodrıguez, I., and Rubio, F. (2005). Specification and testing of autonomous agents in e-commerce systems. *Software Testing, Verification and Reliability*. 15.4, pp. 211–233.

Padgham, L. and Winikoff, M. (2005). *Developing intelligent agent systems: A practical guide*. Vol. 13. John Wiley & Sons.

Papageorgiou, M., Diakaki, C., Dinopoulou, V., Kotsialos, A., and Wang, Y. (2003). Review of road traffic control strategies. *Proceedings of the IEEE*. 91.12, pp. 2043–2067.

Parveen, T., Tilley, S., Daley, N., and Morales, P. (2009). "Towards a distributed execution framework for JUnit test cases". In: *2009 IEEE International Conference on Software Maintenance*. IEEE, pp. 425–428.

Reynolds, C. W. (1987). *Flocks, herds and schools: A distributed behavioral model*. Vol. 21. 4. ACM.

Robertson, D. I. (1969). TRANSYT: a traffic network study tool.

Roozemond, D. A. (2001). Using intelligent agents for pro-active, real-time urban intersection control. *European Journal of Operational Research*. 131.2, pp. 293–301.

Sudeikat, J. and Renz, W. (2008). “A systemic approach to the validation of self–organizing dynamics within MAS”. In: *International Workshop on Agent-Oriented Software Engineering*. Springer, pp. 31–45.

Sycara, K. P. (1998). Multiagent systems. *AI magazine*. 19.2, pp. 79–79.

Wang, F.-Y. et al. (2005). *Agent-based control for networked traffic management systems*.

Weiss, T., Litteneker, A., Jiang, C., and Terzopoulos, D. (2017). “Position-based multi-agent dynamics for real-time crowd simulation”. In: *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, p. 27.

Wolfram, S. (1986). *Theory and applications of cellular automata: including selected papers 1983-1986*. World scientific.

Yamamoto, K., Kokubo, S., and Nishinari, K. (2007). Simulation for pedestrian dynamics by real-coded cellular automata (RCA). *Physica A: Statistical Mechanics and its Applications*. 379.2, pp. 654–660.

Zhang, Z., Thangarajah, J., and Padgham, L. (2007). Automated Unit Testing for Agent Systems. *ENASE*. 7, pp. 10–18.