J Inno Sci Eng, 2024, 8(2):179-198
https://doi.org/ 10.38088/jise.1482853

Research Article

A New Collision Avoidance Approach for Automated Guided Vehicle Systems Based on Finite State Machines

Mustafa Coban [1*], Gokhan Gelen [1]

[1] *Department of Mechatronics Engineering, Bursa Technical University, 16310 Bursa, Turkey*

Cite this paper as:
Coban, M., Gelen, G., (2024). *A New Collision Avoidance Approach for Automated Guided Vehicle Systems Based on Finite State Machines.*Journal of Innovative Science and Engineering. **Error! Reference source not found.**:179-198

*Corresponding author: Mustafa Coban
E-mail: mustafacoban93@gmail.com

Received Date: 13/05/2024
Accepted Date: 04/07/2024

Abstract

Automated guided vehicles are transportation systems that are widely used in factories, warehouses, and distribution centers. It is of great importance to ensure the control and coordination of vehicles for safe and efficient transportation in multi-vehicle systems. In this study, a control strategy is proposed to enforce collision avoidance of automated guided vehicles operating in a shared zone and overlapping route environment. In the proposed method, while finite state machines are used to model the movement of automated guided vehicles in the environment, the Q-learning method, one of the most common reinforcement learning algorithms, is used for collision avoidance. The presented approach uses the decentralized node-based approach to reduce computational complexity. The proposed method has been validated through simulation performed with vehicle applications that can move both unidirectional and bidirectional. The simulation results show that our presented approach can avoid potential collisions and greatly increase overall efficiency.

Keywords: Automated guided vehicle, Collision avoidance, Finite state machines, Reinforcement learning

1. Introduction

Nowadays, with the widespread use of Industry 4.0 and the Internet of Things, the concepts of smart factories and smart production methods have become very important. In smart production methods, which have the capacity to produce more flexible and faster than conventional production methods, product or part transportation between machines is provided by automated guided vehicles (AGVs). These vehicles, which can carry various load capacities, are frequently used in production and assembly factories, smart warehouses, port terminals, and distribution centers. AGVs provide many advantages to businesses by increasing the automation of production processes, reducing costs, increasing efficiency, and optimizing the use of human labor. With the increase in production volume, the number of AGVs used in facilities also increases. In systems that use multiple AGVs, it is of great importance for the safety and efficiency of the system to assign appropriate tasks to the vehicles and to fulfill their movements by avoiding collisions. The complexity of the structure of these systems makes system control more difficult. Therefore, much research has been carried out in recent years to ensure efficient and safe operation of the systems. Although research is generally in areas such as route planning, collision avoidance, vehicle task assignment and planning, and vehicle positioning, collision avoidance and task assignment are at the forefront of research. Studies have covered a wide range of areas such as centralized or decentralized control methods, grid-based or node-based methods, and classical or artificial intelligence-based methods, for AGVs that can operate in unidirectional, bidirectional, or multidirectional. While classical methods with centralized control were generally applied for simple environments in the early studies related to AGVs, later on, control methods applied to more complex systems, control methods with distributed structure and artificial intelligence-based control methods started to be included in the studies.

In multi AGV systems, the changes between states can be modeled as event-based. Such systems are called Discrete Event Systems (DES) [1]. Methods such as finite state machines (FSM), Petri nets, directed graphs, etc. are used in DES modeling. FSM and Petri nets are the most widely used methods due to their modeling capability and flexible solution methods.

There are several studies in the literature that use DES for collision avoidance and zone control in systems with multiple AGVs. Fanti [2] proposes a new control method for avoiding deadlocks and collisions of multiple AGVs by modeling AGVs with colored-timed Petri nets. A survey on deadlock control methods for automated manufacturing systems based on directed graphs, automata and Petri nets approaches is presented by Fanti and Zhou [3]. Wu and Zhou [4] propose a Petri net modeling method for deadlock avoidance in an automated manufacturing system with multiple AGVs. A FSM modeling and deadlock avoidance for safe and efficient coordination of multiple mobile robots is presented in [5]. In another study, which provides supervised control of AGVs in flexible manufacturing systems, FSM are used for high-level control of the system [6]. Fanti et al. [7, 8], in their studies conducted in 2015 and 2018, proposes decentralized methods for the control and coordination of AGV systems. In these studies, they performed a task search for the AGVs and determined the paths to avoid deadlocks and collisions. In the study conducted by Wan for collision avoidance in AGV systems, a maximum-allowance controller is designed using labeled Petri nets [9]. In Malopolski's work, a new method is proposed for AGVs with unidirectional, bidirectional, and multidirectional mobility by dividing the environment into grids and

collision and deadlock avoidance for AGVs is realized [10]. Zajac and Malopolski [11] developed a more efficient structured online control policy for AGVs with unidirectional, bidirectional, and multidirectional mobility by using a grid partitioning approach and showed that this method can be used in systems with both centralized and decentralized control architectures. Luo et al. [12] design a Petri nets-based maximum-allowance controller for collision avoidance in a system with multiple AGVs. In order to avoid active and passive deadlocks, an event-triggered colored elementary net (ETCEN) based method is presented for motion coordination of multiple AGV systems [13]. Chen et al. [14] developed a new method with a combination of node and grid methods for the control and coordination of multiple AGV systems and it is observed that the system performance and efficiency improved compared to node-based or grid-based methods. In a study on zone control using supervised control theory for bidirectional AGVs, a new method is developed to reduce system complexity and avoid vehicle deadlocks [15].

Reinforcement learning-based methods have also been widely used for the coordination and control of mobile robots or AGVs. Especially in studies focusing on task assignment and route planning of vehicles, deep reinforcement learning and multi-agent reinforcement learning methods come to the forefront as the complexity of the system increases. In order to determine the routes of mobile robots in port terminals and avoid deadlocks, a method developed with the Q-learning technique is presented [16]. In this study, a method is developed to minimize the waiting time as well as find the shortest routes for robots. Nagayoshi et al. [17] developed a reinforcement learning-based decentralized method for route planning of multiple AGV systems. A reinforcement learning-based scheduling method is presented to reduce the product production time and the waiting time of machines and AGVs in systems with multiple machines and multiple AGVs [18]. Hu et al. [19] developed a deep reinforcement learning-based scheduling method for scheduling AGVs in flexible work systems. A deep reinforcement learning-based method is presented to ensure the navigation of robots in multi-mobile robot systems and to realize an optimal coordination between robots [20]. In this study, a reward function that can be applied to different reinforcement learning algorithms is proposed and it is observed that this method is successful in both trained and unknown environments. Zhou et al. [21] developed a reinforcement learning-based method for real-time routing of automated guided vehicles in container terminals. In this study, a Q-learning method is proposed to find the shortest route based on real-time state information including vehicle locations, destination locations, orientations, and vehicle counts, and it is observed that this method provides a stable and efficient solution. A combination of Petri nets and deep reinforcement learning methods for route planning and coordination of multiple AGV systems is presented [22]. In this study, a deep reinforcement learning based solution is presented by using Markov Decision Processes (MDPs) while modeling the system with P-timed Petri nets. A study based on multi-agent reinforcement learning for optimal route planning, AGV coordination and control in warehouses with multiple AGVs is presented [23]. MDPs and deep reinforcement learning methods are used in a study for real-time planning of a manufacturing system with numerically controlled machines and AGVs [24]. Zhang et al. [25] compare the advantages and disadvantages of centrally controlled and decentralized controlled AGV systems, investigated AGV scheduling algorithms, and evaluated AI-based decision-making algorithms. A study of collision-free optimal route planning for multiple AGVs in automated container terminals is presented [26].

As can be seen from the studies in the literature, the most common problems in AGV systems are task assignment to AGVs and collision avoidance. The complexity of the methods proposed in existing studies leads to high performance requirements. Therefore, it is necessary to develop more comprehensible solutions with lower

performance requirements. Due to the complexity of collision avoidance algorithms for both classical methods and AI-based methods, in order to contribute to the need for a simpler and innovative method, a new method for collision avoidance based on FSM and Q-learning is proposed in this study. In the proposed method, FSM are used in the motion modeling phase due to its high modeling capability and simplicity, while the system is controlled by reinforcement learning method due to its innovative and efficient solutions. The proposed method is validated by simulation studies on both unidirectional and bidirectional vehicle systems and the results are discussed.

The rest of the paper is organized as follows. Section 2 gives basic information about FSM and Q-learning. Section 3 describes the proposed method for collision avoidance in AGV systems. Section 4 demonstrates applications of the proposed modeling and control methods for AGVs providing unidirectional and bidirectional motion. Section 5 shows the simulation results applied to validate the proposed method. Finally, Section 6 evaluates the results obtained from the study and concludes the paper with an outlook for future work.

2. Preliminaries

2.1. Finite State Machines

Finite state machines are abstract machines used in system modeling that consist of a finite number of states, transitions between states and actions. In state machines, which is one of the formal language theories, states store the instantaneous information of the system, while transitions show the state change in the system. The realization of transitions is defined within certain rules and the system moves from the current state to the next state. The action is the description of the activity of the system during the state or transition. In this study, system modeling is performed using FSM. Figure 1 shows a basic FSM model.

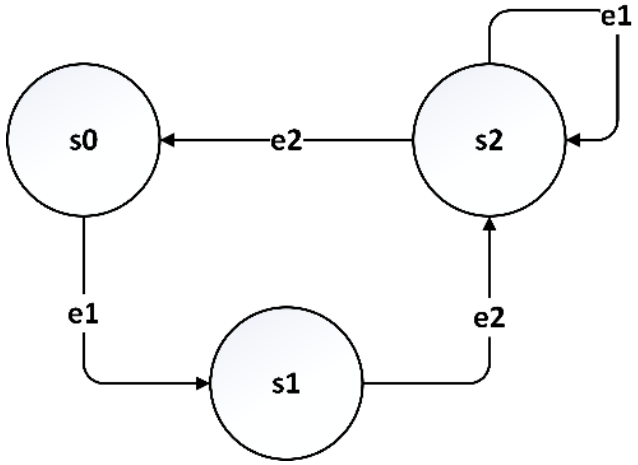

Figure 1: Finite state machines.

In the model shown in Figure 1, s0, s1 and s2 indicate the states of the system, while e1 and e2 indicate transitions. If event e1 occurs while the system is in state s0, the system transitions to state s1. When event e2 occurs while the system is in state s1, the system transitions to state s2. If event e1 occurs while the system is in state s2, the system remains in state s2, while if event e2 occurs, the system transitions to state s0. The working logic of FSM can be explained in this way.

2.2. Reinforcement Learning

Reinforcement learning is a machine learning approach that learns what needs to be done to achieve a given goal. Reinforcement learning allows an agent to interact with its environment and observe the results of that interaction. As a result of these observations, it tries to learn what actions to take by receiving a positive or negative reward. The goal here is to ensure that actions are performed in the most ideal way by achieving the highest amount of reward. This method is used in many fields such as robotics, game programming, process control problems, resource management, and statistics. Modeling with machine learning is usually modeled as a MDPs, which requires prior knowledge about the artificial intelligence system. Reinforcement learning algorithms, on the other hand, do not require prior knowledge about the MDP and can be used when exact methods are insufficient.

In reinforcement learning algorithms, the correctness of the decisions made by artificial intelligence as a result of its interactions with the environment is tried to be controlled with a reward or punishment system. The actions performed by the agent are trained in line with the reward gained, and it is tried to understand how correct or incorrect the actions performed are. The goal of the agent is to gain the highest reward throughout the training. There are five basic elements in the reinforcement learning algorithm. These are agent, state, action, value, and reward. The agent represents the trained artificial intelligence, while the state represents the situation the agent is in. Action indicates the actions that the agent can perform, while value indicates how valuable the agent's state is. Finally, the reward represents the amount of reward the agent will gain or lose as a result of the actions it takes in the situation. The basic block diagram of the reinforcement learning algorithm is presented in Figure 2.

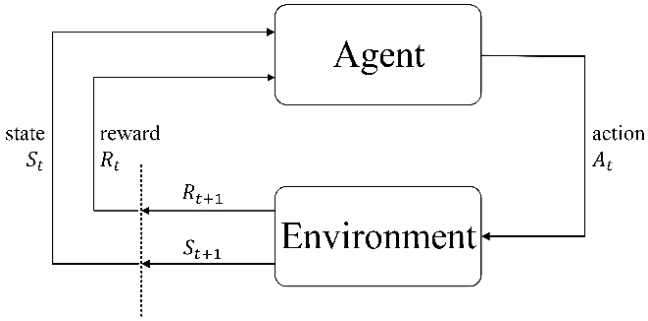

Figure 2: Reinforcement learning block diagram.

Looking at the basic block structure of reinforcement learning in Figure 2, the agent takes an action to move from its current state within the framework of a policy. As a result of this action, it expects a reaction from the environment. These reactions consist of the new state that is traversed by the occurrence of the action and the reward received from this action. Based on the rewards received, the agent is trained and realizes whether the actions it takes are right or wrong. This method aims to maximize the reward and train the agent in the best way possible.

Reinforcement learning involves many learning algorithms. Two of the most popular algorithms are Q-learning and Deep Q-learning. Both algorithms work on a state→action→reward→state logic. In the Q-learning algorithm, the desired or undesired places for the agent to go are determined and written in the reward table. The experience of the agent in each iteration on the way to the reward is stored in a table called the Q-table. Initially, the agent moves randomly. As soon as it finds a reward or punishment, it updates the Q table and thus keeps it in its memory. By continuously performing these operations, the agent learns its environment and can make the right decisions.

The difference of the deep Q-learning algorithm from the Q-learning algorithm is that artificial neural networks are used instead of Q-tables.

Q-learning is a model-free and value-based learning algorithm. Value-based algorithms update the value function based on an equation. In the Q-learning method, Q values are updated using the Bellman equation. The 'Q' stands for quality and represents how useful a certain action is in earning rewards in the future. The Bellman equation that will implement the Q-learning algorithm and update the Q values is defined as in Equation 1.

$$Q_{t+1}(s_t,a_t)=Q_t(s_t,a_t)+\alpha.\left[R(s_t,a_t)+\gamma.\max_a Q_t(s_{t+1},a)-Q_t(s_t,a_t)\right] \quad (1)$$

In Equation 1, '$Q_t(s_t,a_t)$' represents the current Q value, '$\max_a Q_t(s_{t+1},a)$' represents the maximum predicted reward value, '$R(s_t,a_t)$' represents the reward value, and '$Q_{t+1}(s_t,a_t)$' represents the new Q value. In this equation, the coefficient 'α' is the learning rate, while the coefficient denoted by 'γ' is the discount factor, which determines the importance of future rewards. The learning rate is a parameter that determines how fast the network applies the information it has learned. When this ratio is small, learning is slow, while a large ratio may cause oscillation problems and the performance of the network may decrease. Therefore, it is important to find a balance in the choice of learning rate. A value between 0 and 1 is chosen for the learning rate. The discount factor is a value between 0 and 1 and is usually chosen close to 1 in order to give more importance to future rewards. While 'r' in the equation represents the reward received as a result of the action taken, the corresponding Q value is calculated according to the equation for each action taken. Using this equation, the Q table is updated according to the state and actions, allowing the agent to learn the environment. The flow diagram of the Q-learning algorithm is given in Figure 3.

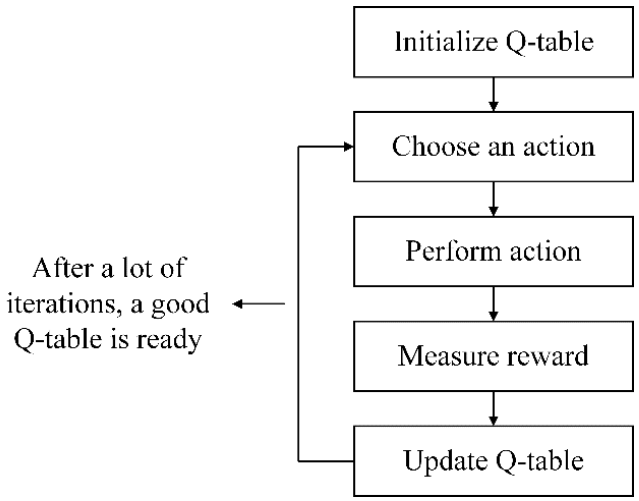

Figure 3: Flowchart of learning algorithm.

According to the algorithm given in Figure 3, first a table Q with all elements consisting of zero is created. Then an action is selected and this action is performed. The reward obtained as a result of this action is determined and the Q table is updated by calculating Q values from the Q-learning function equation. After certain iterations, the learning is completed and a good Q table is obtained. This Q table contains the states and the Q values of the actions corresponding to these states. The action with the highest Q values corresponding to the states represents the most appropriate action to be selected. Thus, the optimal actions are determined for each state in the system.

3. Proposed Method

In the solution method developed for the development of collision avoidance algorithms and simulation applications of AGVs within the scope of the study, the environment model of the system is first determined. According to this environmental model, FSM models are created for each vehicle. Q-tables are obtained from the FSM models of the vehicles by Q-learning method. From the generated Q tables, the actions that each vehicle should perform are determined and the controller is developed. The developed controller is verified in simulation applications.

For the modeling and control studies of systems with AGVs, firstly, a platform with two vehicles is studied. This platform is divided into two types as unidirectional and bidirectional mobility of the AGVs. Within the scope of the study, controller design is realized for both types of operation. The block diagram of the proposed method for the design of these controllers is shown in Figure 4.

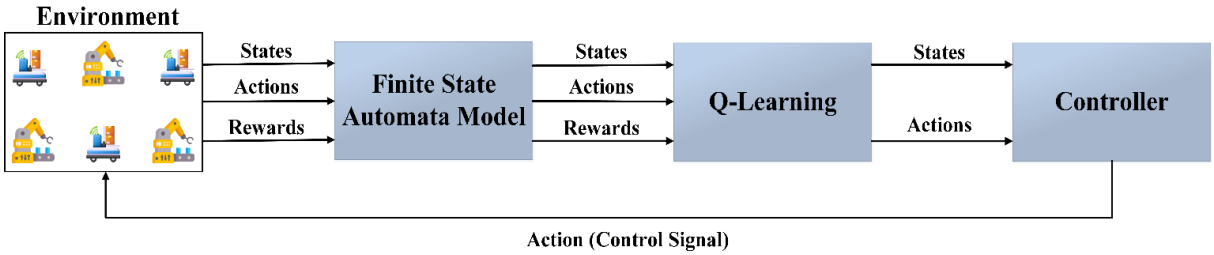

Figure 4: Block diagram of the proposed method.

The block diagram in Figure 4 shows the proposed control method for both vehicles on the working platform. Based on the state, action, and reward information received from the environment, each vehicle is modeled with FSM and a Q-learning algorithm is generated. From the Q-learning algorithm, the actions that the vehicles should perform in response to their states are determined and sent to the controller. Control signals containing the actions that the vehicles should perform are sent to the vehicles by the controller and new states are observed. With this cycle, system control is achieved.

4. Applications

For the modeling and control studies of systems with AGVs, a platform with two vehicles is studied. This platform is divided into two types as unidirectional and bidirectional according to the mobility of the AGVs. Vehicles with unidirectional operation type can stop at the positions shown as nodes on their routes and follow a forward line between these nodes, while bidirectional AGVs can move forward, stop and move backward between nodes. The nodes represent the positions of the vehicles. In this study, it is assumed that the position of the vehicles at the nodes is detected through a radio frequency (RF) signal or a QR code placed at the node.

4.1. Unidirectional AGV System

Figure 5 shows the platform determined for unidirectional AGV system.

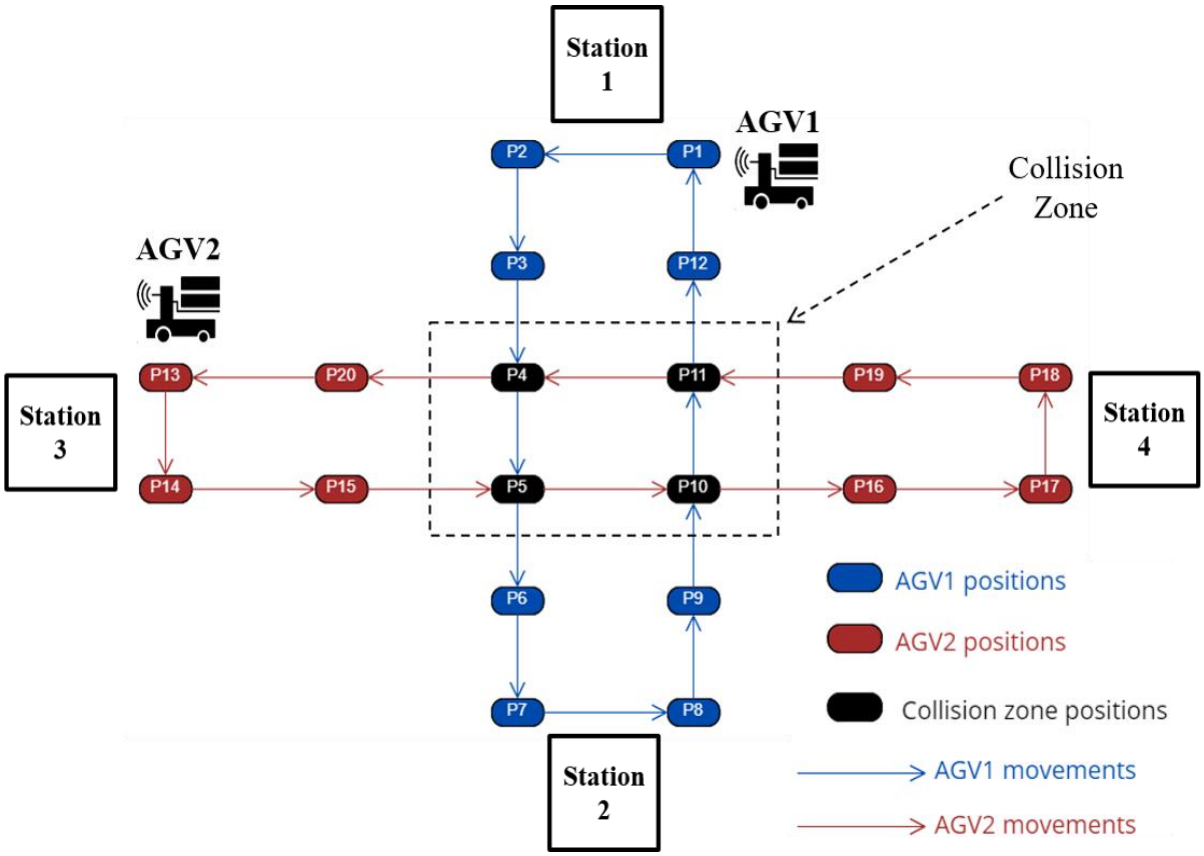

Figure 5: Working platform of unidirectional AGVs.

The blue arrows in Figure 5 indicate the direction of movement of AGV1, while the red arrows indicate the direction of movement of AGV2. While AGV1 moves on the blue and black nodes and transports parts from Station1 to Station2, AGV2 moves on the red and black nodes and transports parts from Station3 to Station4. The nodes shared by the two vehicles are indicated in black and this area is called the collision zone. It is defined that AGV1 starts its movement from point P1, follows the nodes on its route, returns to point P1 and completes its mission. AGV2 is defined to start its movement from point P13 and return to point P13 by following the nodes on its route and complete its task. The vehicles are expected to perform their movements simultaneously without being at the nodes in the collision zone.

4.1.1. Modeling of Unidirectional AGV System with Finite State Machines

In order to prevent collisions in the AGV system shown in Figure 5, a decentralized control study is carried out. In this method, control mechanisms are created separately for both AGVs and the movements that the vehicles need to perform are controlled. Q-learning method, one of the reinforcement learning algorithms, is used for the control of the vehicles in the study. First of all, the modeling of the AGVs with FSM is performed. Based on these models, the working platform of the vehicles is created and states, next states and rewards are determined according to the actions that could be taken. The Q table to be used for AGV control is updated using the Bellman equation and the maximum Q value in the actions corresponding to the states in the Q table obtained is assigned as the action that should occur in that state.

The model shown in Figure 6 is created for AGV1 in a unidirectional working environment. While creating this model, three states are defined for each location of AGV1. These states are when AGV2 is outside the collision zone, when AGV2 is near the collision zone and when AGV2 is inside the collision zone. Since there are 12 positions for the movement of AGV1, the total number of states is determined as 36. The state changes of the system are provided by transitions. In unidirectional operation, three transitions are determined for AGV1. The first transition indicates that AGV1 is waiting, the second transition indicates that AGV1 moves one node forward, and the third transition indicates that AGV2 changes its state in the collision zone.

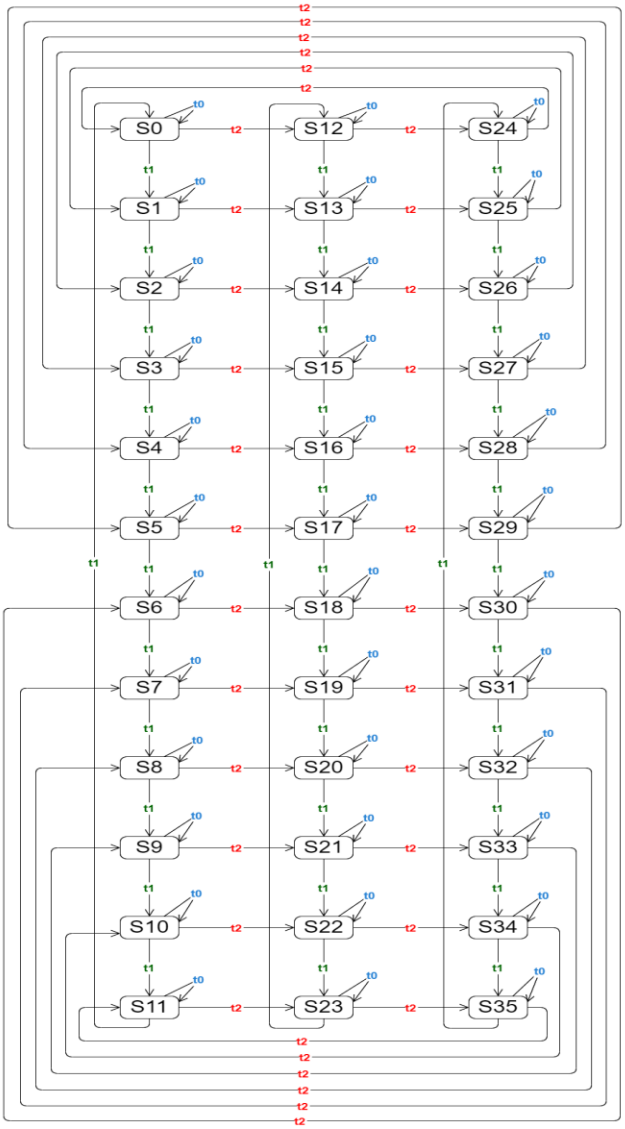

Figure 6: Finite state machine model for unidirectional AGV.

In the model shown in Figure 6, states S0 to S11 indicate that AGV1 is located at positions between P1 and P12, respectively, while AGV2 is located outside the collision zone. States between S12 and S23 likewise represent AGV1 being located at positions between P1 and P12, while AGV2 is located at positions P15 or P19 near the collision zone. States between S24 and S35 indicate that AGV1 is located at positions between P1 and P12, while AGV2 is located within the collision zone. In the model in Figure 6, the transition t0 indicates that AGV1 is waiting for an action, the transition t1 indicates that AGV1 moves one node forward, and the transition t2 indicates that AGV2 changes state in the collision zone.

The model in Figure 6 used for AGV1 can also be used for AGV2, which has a similar type of operation and the same number of nodes. In the route between P13 and P20 shown in Figure 5, there are 12 nodes where the movement of AGV2 takes place. In this model, states between S0 and S11 indicate that AGV2 is located between P13 and P20 respectively, while AGV1 is outside the collision zone. States between S12 and S23 likewise represent AGV2 being located at positions between P13 and P20, while AGV1 is located at P3 or P9 near the collision zone. States between S24 and S35 indicate that AGV2 is located between P13 and P20, while AGV1 is located within the collision zone. In the model in Figure 6, the transition t0 indicates the action that AGV2 is waiting for, the transition t1 indicates the action that AGV2 moves one node forward, and the transition t2 indicates that AGV1 changes state in the collision zone.

4.1.2. Q-learning Algorithm for Unidirectional AGV System

Q-learning algorithm, which is one of the reinforcement learning methods, is created according to the FSM model in Figure 6 for the control of AGV1 and AGV2 with unidirectional mobility. The rewards used in this algorithm are determined as shown in Table 1.

Table 1: Rewards for unidirectional AGVs.

AGV Actions	Reward / Punishment
Waiting with no vehicles in or near the collision zone	-1
Action that the vehicle cannot perform	-12
Entering the collision zone with a vehicle inside	-15
Waiting for the other vehicle to leave the area while there is a vehicle in the collision zone	3
Moving forward when no other vehicle is nearby	3
Complete the movement and return to the starting point	10

In Table 1, the action specified as the action that the vehicle cannot perform is t2. The action t2 given in the model is an action that the other vehicle cannot control because it is an action that shows the change of state in the collision zone, and it is intended to prevent this action from occurring by keeping the punishment value high. However, since the state may change in the collision area by the other vehicle, this action is defined in the model and used in the Q-learning algorithm.

For the Q-learning algorithm, first, a reinforcement learning environment is created in accordance with the reward/punishment values given in Table 1 and the model given in Figure 6. In this environment, a Q-learning algorithm is created separately for each of the two AGVs and the Q tables are determined. For the cases of arriving near the collision zone at the same time, AGV1 is given priority. According to this priority, in case both vehicles are near the collision zone at the same time, the AGV1 will move and enter the collision zone, while the AGV2 will wait for the AGV1 to leave the zone. Once AGV1 leaves the collision zone, AGV2 will move. For the Q-learning algorithm to perform these operations, the hyperparameters must be selected appropriately. The main hyperparameters to be chosen in the Q-learning algorithm are the learning rate, discount factor and epsilon. The choice of these parameters represents how well the model can learn. The learning rate and discount factor are mentioned in Section 2. The epsilon value is a parameter that determines the probability of an agent choosing random actions. The epsilon takes a value between 0 and 1 and the larger the epsilon value, the more likely the agent is to choose random actions. This can lead to less utilization of the learned knowledge and therefore lower

performance. That is, if the epsilon value is chosen too high, the agent will make more explorations, but these explorations may ignore the learned knowledge and lead to less accurate actions. If the epsilon value is too small, the agent will be less likely to choose random actions, reducing the opportunity to test and optimize its learned knowledge. Therefore, it is important to find a balance in the choice of the epsilon value. The choice of hyperparameters can be determined by trial and error or by various algorithms. One of these selection algorithms is the grid search algorithm. The grid search algorithm is a method used to optimize the performance of a machine learning model. It is a search algorithm that tries different values for various hyperparameters of the model and determines which hyperparameters give the best performance. In this study, the discount factor value is chosen as a constant 0.95 and a grid search algorithm is used to select the learning rate and epsilon hyperparameters. In this algorithm, a Q-learning algorithm is created for nine learning rates varying between 0.1 and 0.9 in 0.1 increments and nine epsilon values varying in the same way and Q tables are analyzed. As a result of the examination of the Q tables, it is determined that the model learned the environment well at values where the learning rate is 0.1 and epsilon is 0.4 for both AGVs and hyperparameters are selected. According to these hyperparameters, the Q-table for AGV1 is shown in Figure 7 and the Q-table for AGV2 is shown in Figure 8.

QTable for Unidirectional AGV1

States	t0	t1	t2
s0	64.23	68.66	53.23
s1	64.66	69.12	53.66
s2	65.12	69.6	54.12
s3	65.6	70.1	54.6
s4	66.1	70.64	55.1
s5	66.64	71.19	55.64
s6	67.19	71.78	56.19
s7	67.78	72.4	56.78
s8	68.4	73.06	57.4
s9	69.06	73.74	58.06
s10	69.74	74.47	58.74
s11	70.47	75.23	59.47
s12	64.23	68.66	45
s13	64.66	69.12	45
s14	65.12	69.6	45
s15	65.6	70.1	45
s16	66.1	70.64	45
s17	66.64	71.19	45
s18	67.19	71.78	45
s19	67.78	72.4	45
s20	68.4	73.06	45
s21	69.06	73.74	51
s22	69.74	74.47	51.32
s23	70.47	75.23	51.65
s24	55.01	60	51.96
s25	56	60	53.65
s26	60	42	54.12
s27	56	60	54.6
28	56	60	55.1
s29	56	60	55.64
s30	56	60	56.19
s31	56	60	56.78
s32	60	48	57.4
s33	62	66.32	58.06
s34	62.32	66.65	58.74
s35	62.65	67	59.47

Actions

Figure 7: Q table for unidirectional AGV1.

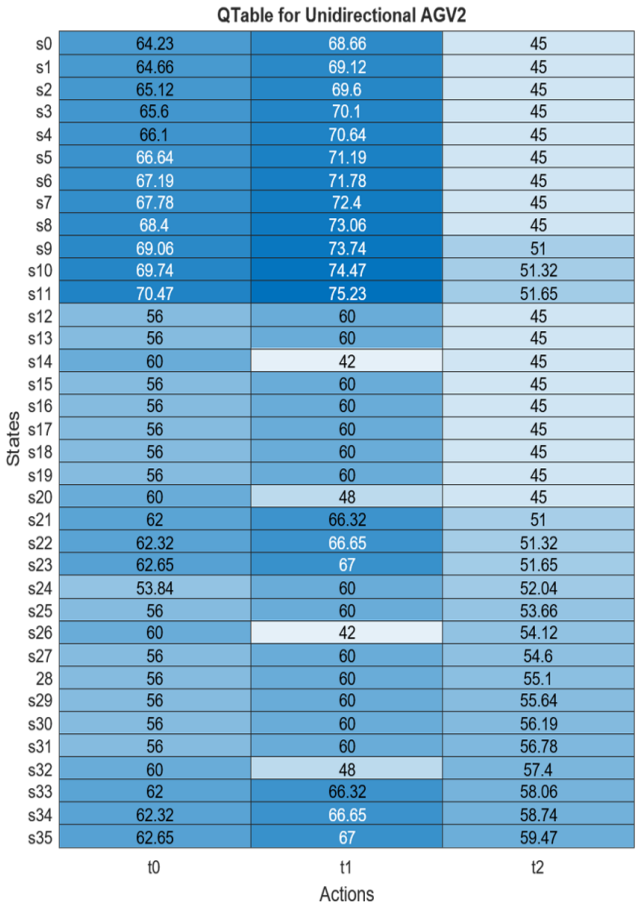

Figure 8: Q table for unidirectional AGV2.

The rows of the Q tables in Figure 7 and Figure 8 show the states of the vehicles and the columns show their actions. The column corresponding to the maximum value in each row in this table represents the action to be performed in that state. In this way, the control method is developed for both AGVs.

4.2. Bidirectional AGV System

In AGVs with bidirectional movement capability, vehicles perform backward movement in addition to stopping at nodes and forward line following movement between nodes. For vehicles with this type of operation, the actions of moving forward, moving backward and stopping are defined. Figure 9 shows the working platform of OYAs with bidirectional operation type.

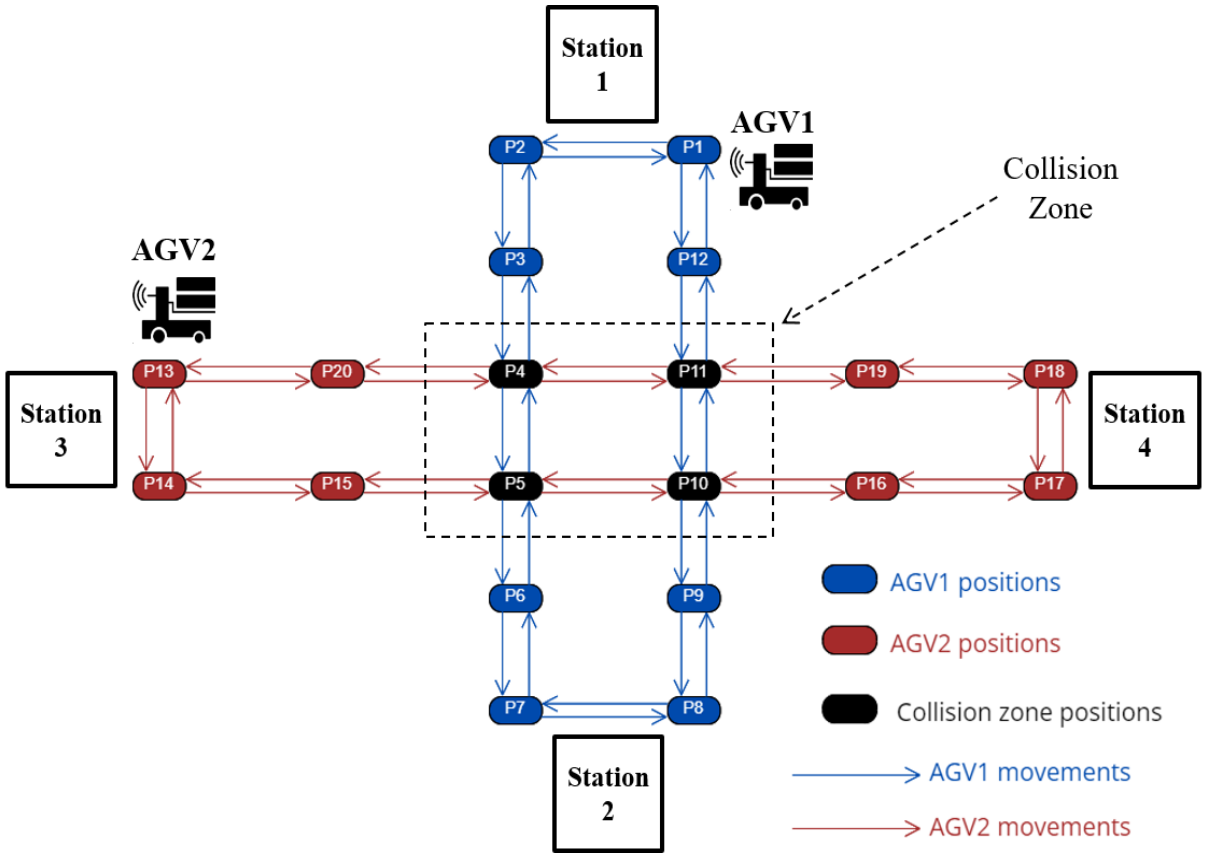

Figure 9: Working platform of bidirectional AGVs.

The blue arrows in Figure 9 indicate the direction of movement for AGV1, while the red arrows indicate the direction of movement for AGV2. While AGV1 moves on the blue and black nodes and transports parts from Station1 to Station2, AGV2 moves on the red and black nodes and transports parts from Station3 to Station4. The nodes shared by the two vehicles are indicated in black and this area is called the collision zone. It is defined that AGV1 starts its movement from point P1, follows the nodes on its route, returns to point P1 and completes its mission. AGV2 is defined as starting its movement from point P13, following the nodes on its route and returning to point P13 and completing its mission. The only difference of this type of operation from the unidirectional type is that the vehicles have the option to move backward. In both unidirectional and bidirectional AGVs, vehicles should not be in the collision zone at the same time.

4.2.1. Modeling of Bidirectional AGV System with Finite State Machines

In order to prevent collisions in the AGV system shown in Figure 9, a decentralized control study is carried out. In this method, as in the unidirectional type of operation, separate control mechanisms are created for both AGVs and the movements that the vehicles need to perform are controlled. The Q-learning method is used for the control of the vehicles in the study. First of all, the modeling of the AGVs with FSM is performed. Based on these models, the working platform of the vehicles is created and states, next states, and rewards are determined according to the actions that could be taken. The Q table to be used for AGV control is updated using the Bellman equation and the maximum Q value in the actions corresponding to the states in the Q table obtained is assigned as the action that should occur in that state.

For the AGV1 in bidirectional operation environment, the model shown in Figure 10 is created. While creating this model, three states are defined for each location of AGV1. These states are when AGV2 is outside the collision

zone, when AGV2 is near the collision zone and when AGV2 is inside the collision zone. Since there are 12 positions for the movement of OYA1, the total number of states is determined as 36. The state changes of the system are provided by transitions. In the bidirectional system, four transitions are determined for AGV1. The first transition indicates the action that AGV1 is waiting, the second transition indicates the action that AGV1 moves one node forward, the third transition indicates that AGV2 changes its state in the collision zone, and the fourth transition indicates the action that AGV1 moves one node backward.

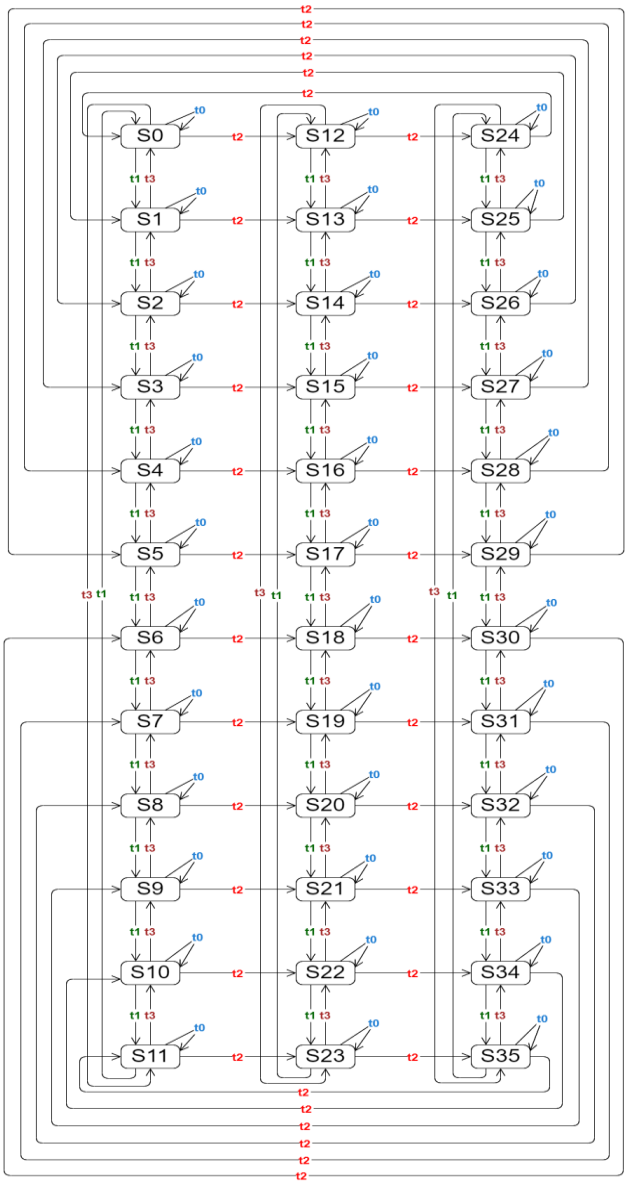

Figure 10: Finite state machine model for bidirectional AGV.

In the model shown in Figure 10, states between S0 and S11 indicate that AGV1 is located at positions between P1 and P12 respectively, while AGV2 is located outside the collision zone. States between S12 and S23 likewise represent AGV1 being located at positions between P1 and P12, while AGV2 is located at positions P15 or P19 near the collision zone. States between S24 and S35 indicate that AGV1 is located at positions between P1 and P12, while AGV2 is located within the collision zone. In the model in Figure 10, the transition t0 indicates the action that AGV1 is waiting for, the transition t1 indicates the action that AGV1 moves one node forward, the transition t2 indicates that AGV2 changes state in the collision zone, and the transition t3 indicates the action that AGV1 moves one node backward.

The model in Figure 10 used for AGV1 can also be used for AGV2, which has a similar type of operation and the same number of nodes. In the route between P13 and P20 shown in Figure 9, there are 12 nodes where the movement of AGV2 takes place. In this model, states between S0 and S11 indicate that AGV2 is located between P13 and P20 respectively, while AGV1 is outside the collision zone. States between S12 and S23 likewise represent AGV2 being located at positions between P13 and P20, while AGV1 is located at positions P3 or P9 near the collision zone. States between S24 and S35 indicate that AGV2 is located between P13 and P20, while AGV1 is located within the collision zone. In the model in Figure 11, the transition t0 indicates the action that AGV2 is waiting for, the transition t1 indicates the action that AGV2 moves one node forward, the transition t2 indicates that AGV1 changes state in the collision zone, and the transition t3 indicates the action that AGV2 moves one node backward.

4.2.2. Q-learning Algorithm for Bidirectional AGV System

The Q-learning algorithm is created according to the FSM model in Figure 10 for the control of AGV1 and AGV2 with bidirectional mobility. The rewards used in this algorithm are determined as shown in Table 2.

Table 2: Rewards for bidirectional AGVs.

AGV Actions	Reward / Punishment
Waiting with no vehicles in or near the collision zone	-1
Moving backward when no other vehicle is nearby	-3
Action that the vehicle cannot perform	-12
Entering the collision zone with a vehicle inside	-15
Waiting for the other vehicle to leave the area while there is a vehicle in the collision zone	3
Moving forward when no other vehicle is nearby	3
Exiting the collision zone by moving backward when there is a vehicle in the collision zone	3
Complete the movement and return to the starting point	10

The action that the vehicle cannot perform in Table 2 refers to action t2, as stated in the unidirectional study. The t2 action given in the model is an action that the other vehicle cannot control, as it is an action that shows the change of situation in the collision zone, and it is intended to prevent this action from occurring by keeping the penalty value high. However, since the state may change in the collision area by the other vehicle, this action is defined in the model and used in the Q-learning algorithm.

For the Q-learning algorithm in vehicles with bidirectional operation type, a reinforcement learning environment is created in accordance with the reward/punishment values given in Table 2 and the model given in Figure 10. Separate Q-learning algorithms are created and Q-tables are determined for both AGVs in this environment. For the cases of arriving near the collision zone at the same time, the priority is again given to the AGV1. According to this priority, in case both vehicles are near the collision zone at the same time, the AGV1 will move and enter the collision zone, while the AGV2 will not move and wait for the AGV1 to leave the zone. Once AGV1 leaves the collision zone, AGV2 will move. Unlike the unidirectional system, even if there is a vehicle in the collision zone, the vehicle entering the zone will exit the zone by moving backwards. For the Q-learning algorithm to perform these operations, the hyperparameters should be chosen appropriately. As in the unidirectional study, the discount factor value is chosen as a constant 0.95 and a grid search algorithm is used to select the learning rate and epsilon hyperparameters. In this algorithm, a Q-learning algorithm is created for nine learning rates varying between 0.1

and 0.9 in 0.1 increments and nine epsilon values varying in the same way and Q tables are analyzed. As a result of examining the Q tables, it is determined that the model learned the environment well at values where the learning rate is 0.1 and epsilon is 0.4, as in the unidirectional study for both AGVs, and hyperparameters were selected. According to these hyperparameters, the Q-table for AGV1 is shown in Figure 11 and the Q-table for AGV2 is shown in Figure 12.

QTable for Bidirectional AGV1

States	t0	t1	t2	t3
s0	64.23	68.66	53.23	68.47
s1	64.66	69.12	53.66	62.23
s2	65.12	69.6	54.12	62.66
s3	65.6	70.1	54.6	63.12
s4	66.1	70.64	55.1	63.6
s5	66.64	71.19	55.64	64.1
s6	67.19	71.78	56.19	64.64
s7	67.78	72.4	56.78	65.19
s8	68.4	73.06	57.4	65.78
s9	69.06	73.74	58.06	66.4
s10	69.74	74.47	58.74	67.06
s11	70.47	75.23	59.47	67.74
s12	64.23	68.66	45	68.47
s13	64.66	69.12	45	62.23
s14	65.12	69.6	45	62.66
s15	65.6	70.1	45	63.12
s16	66.1	70.64	45	54.6
s17	66.64	71.19	45	55.1
s18	67.19	71.78	45	64.64
s19	67.78	72.4	45	65.19
s20	68.4	73.06	45	65.78
s21	69.06	73.74	45	66.4
s22	69.74	74.47	51.32	58.06
s23	70.47	75.23	51.65	58.74
s24	54.24	60	49.9	60.1
s25	56	60	53.66	54
s26	60	42	54.12	54
s27	56	42	54.6	60
28	56	60	55.1	42
s29	56	60	55.64	45
s30	56	60	56.19	54
s31	56	60	56.78	54
s32	60	42	57.4	54
s33	56	48.32	58.06	60
s34	62.32	66.66	58.74	42
s35	62.65	67.02	59.47	48.32

Actions

Figure 11: Q table for bidirectional AGV1.

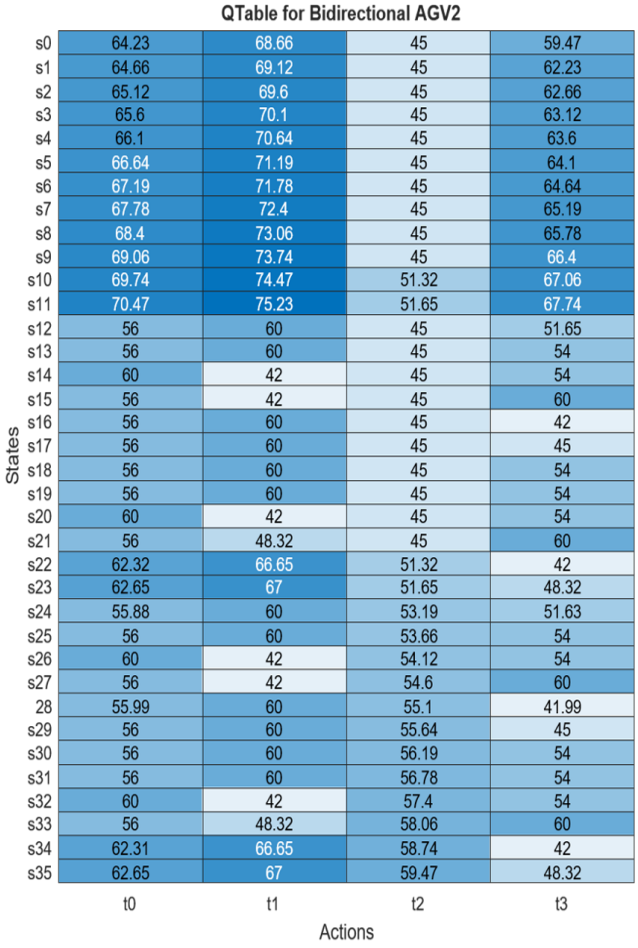

Figure 12: Q table for bidirectional AGV2.

In the Q tables shown in Figure 11 and Figure 12, the rows represent the states of the vehicles and the columns represent their actions. The column corresponding to the maximum value in each row in this table represents the action to be performed in that state. In this way, the control method is developed for both AGVs.

5. Results and Discussion

In order to validate the proposed method, simulation applications have been carried out for unidirectional and bidirectional AGVs. Using the Robotics System Toolbox in the Matlab&Simulink software, AGV modeling has been done and a working environment has been created. Q tables containing the actions that will control the vehicles are added as the block that decides the actions in the model and data are obtained through simulations. Figure 13 shows the conditions of the unidirectional AGVs in the collision zone.

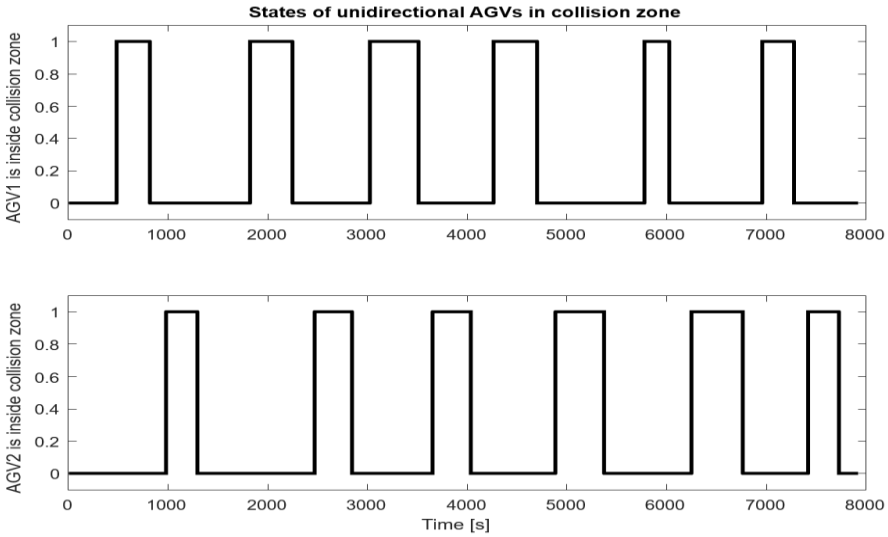

Figure 13: States of unidirectional AGVs in collision zone.

In the graph in Figure 13, the presence of vehicles in the collision zone is shown as 0 or 1. If the collision zone status of the vehicle is 1, the vehicle is inside the collision zone, and if it is 0, the vehicle is outside the collision zone. Therefore, when the collision zone status of the vehicles in Figure 13 is analyzed, it is seen that AGV1 and AGV2 are not in the collision zone at the same time throughout the operation. AGV1 always enters the collision zone before AGV2 and the vehicles are never in the collision zone in the same time period. Both vehicles move on their own routes for a certain period of time and no collision occurs during these movements. These results confirm that the method proposed in this study is suitable for unidirectional AGVs.

Figure 14 shows the collision occurrence of bidirectional AGVs in the collision zone within the scope of the study.

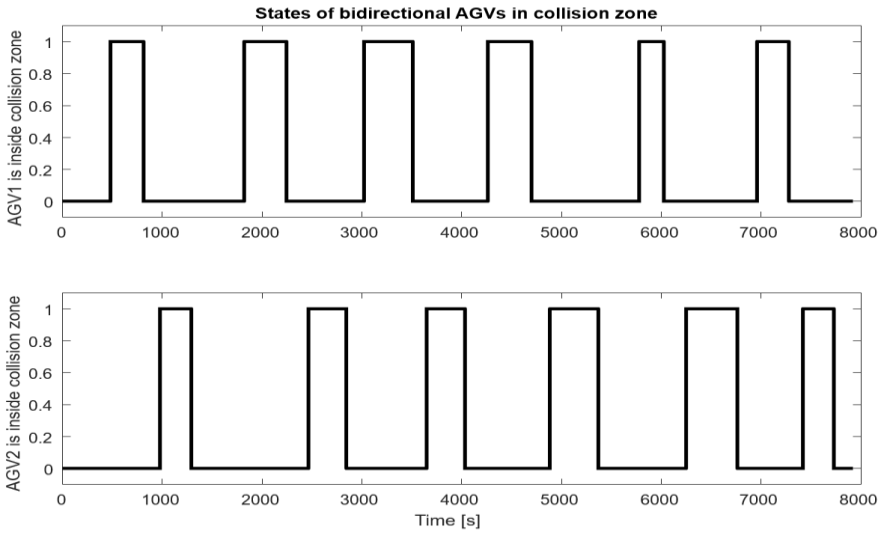

Figure 14: States of bidirectional AGVs in collision zone.

In the graph shown in Figure 14, the presence of vehicles in the collision zone is shown as 0 or 1. As in unidirectional operation, if the collision zone status of the vehicle is 1, the vehicle is inside the collision zone, and if it is 0, the vehicle is outside the collision zone. Therefore, when the collision zone status of the bidirectional vehicles in Figure 14 is analyzed, it is seen that AGV1 and AGV2 are not in the collision zone at the same time throughout the study. AGV1 always enters the collision zone before AGV2 and the vehicles are never in the collision zone in the same time period. Both vehicles move on their own routes for a certain period of time and no

collision occurs during these movements. These results confirm that the method proposed in this study is suitable for bidirectional AGVs.

6. Conclusion

In this study, an effective modeling and control method is proposed for collision avoidance of AGVs operating in an environment with a shared work zone and overlapping routes. For unidirectional and bidirectional vehicles, a decentralized node-based method is used to reduce the complexity of the model and simplify the control of the system. In the study, finite state machines are used for AGV modeling and Q-learning algorithm, one of the methods of reinforcement learning, is used for collision avoidance. The proposed method is validated with simulations of unidirectional and bidirectional vehicles. By using this method, AGVs operating in industrial environments can perform their movements and tasks without collisions. Future work is planned to improve the proposed methodology to provide a scalable control strategy for AGVs operating in environments with a larger number of vehicles and more complex conflicting routes.

References

[1] Cassandras, C. G., and Lafortune, S. (1999). Discrete event systems: The state of the art and new directions. *Applied and Computational Control, Signals, and Circuits:* Volume 1: 1-65.

[2] Fanti, M. P. (2002). A deadlock avoidance strategy for AGV systems modelled by coloured Petri nets. *In Sixth International Workshop on Discrete Event Systems*, 61-66.

[3] Fanti, M. P., and Zhou, M. (2004). Deadlock control methods in automated manufacturing systems. *IEEE Transactions on systems, man, and cybernetics-part A: systems and humans*, 34(1): 5-22.

[4] Wu, N., and Zhou, M. (2005). Modeling and deadlock avoidance of automated manufacturing systems with multiple automated guided vehicles. *IEEE Transactions on Systems, Man, and Cybernetics*, Part B (Cybernetics), 35(6): 1193-1202.

[5] Manca, S., Fagiolini, A., and Pallottino, L. (2011). Decentralized coordination system for multiple agvs in a structured environment. *IFAC Proceedings Volumes*, 44(1): 6005-6010.

[6] Hernández Martínez, E. G., Pérez Sampieri, J. C., and Aranda Bricaire, E. (2013). Supervisory control of AGV's for flexible manufacturing cells. *Congreso Nacional de Control Automatico*, Ensenada-Baja California-Meksika, 366-371.

[7] Fanti, M. P., Mangini, A. M., Pedroncelli, G., and Ukovich, W. (2015). Decentralized deadlock-free control for AGV systems. *In 2015 American Control Conference* (ACC), 2414-2419.

[8] Fanti, M. P., Mangini, A. M., Pedroncelli, G., and Ukovich, W. (2018). A decentralized control strategy for the coordination of AGV systems. *Control Engineering Practice*, 70: 86-97.

[9] Wan, Y., Luo, J., Zhang, Q., Wu, W., Huang, Y., and Zhou, M. (2018). Controller design for avoiding collisions in automated guided vehicle systems via labeled petri nets. *IFAC-PapersOnLine*, 51(7): 139-144.

[10] Małopolski, W. (2018). A sustainable and conflict-free operation of AGVs in a square topology. *Computers & Industrial Engineering*, 126: 472-481.

[11] Zając, J., and Małopolski, W. (2021). Structural on-line control policy for collision and deadlock resolution in multi-AGV systems. *Journal of Manufacturing Systems*, 60: 80-92.

[12] Luo, J., Wan, Y., Wu, W., and Li, Z. (2019). Optimal Petri-net controller for avoiding collisions in a class of automated guided vehicle systems. *IEEE Transactions on Intelligent Transportation Systems*, 21(11): 4526-4537.

[13] Chen, X., Xing, Z., Feng, L., Zhang, T., Wu, W., and Hu, R. (2022). An ETCEN-based motion coordination strategy avoiding active and passive deadlocks for multi-AGV system. *IEEE Transactions on Automation Science and Engineering*, 20(2): 1364-1377.

[14] Chen, X., Wu, W., and Hu, R. (2022). A Novel Multi-AGV Coordination Strategy Based on the Combination of Nodes and Grids. *IEEE Robotics and Automation Letters*, 7(3): 6218-6225.

[15] Maza, S. (2023). Hybrid supervisory-based architecture for robust control of Bi-directional AGVs. *Computers in Industry*, 144: 103797.

[16] Jeon, S. M., Kim, K. H., and Kopfer, H. (2011). Routing automated guided vehicles in container terminals through the Q-learning technique. *Logistics Research*, 3: 19-27.

[17] Nagayoshi, M., Elderton, S. J., Sakakibara, K., and Tamaki, H. (2017). Reinforcement Learning Approach for Adaptive Negotiation-Rules Acquisition in AGV Transportation Systems. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 21(5): 948-957.

[18] Xue, T., Zeng, P., and Yu, H. (2018). A reinforcement learning method for multi-AGV scheduling in manufacturing. *In 2018 IEEE international conference on industrial technology (ICIT)*, 1557-1561.

[19] Hu, H., Jia, X., He, Q., Fu, S., and Liu, K. (2020). Deep reinforcement learning based AGVs real-time scheduling with mixed rule for flexible shop floor in industry 4.0. *Computers & Industrial Engineering*, 149: 106749.

[20] Jestel, C., Surmann, H., Stenzel, J., Urbann, O., and Brehler, M. (2021). Obtaining robust control and navigation policies for multi-robot navigation via deep reinforcement learning. *In 2021 7th International Conference on Automation, Robotics and Applications (ICARA)*, 48-54.

[21] Zhou, P., Lin, L., and Kim, K. H. (2023). Anisotropic Q-learning and waiting estimation based real-time routing for automated guided vehicles at container terminals. *Journal of Heuristics*: 1-22.

[22] Zhang, H., Luo, J., Lin, X., Tan, K., and Pan, C. (2021). Dispatching and path planning of automated guided vehicles based on petri nets and deep reinforcement learning. *In 2021 IEEE International Conference on Networking, Sensing and Control (ICNSC)* Vol. 1, 1-6.

[23] Choi, H. B., Kim, J. B., Ji, C. H., Ihsan, U., Han, Y. H., Oh, S. W., Kim, K. H. and Pyo, C. S. (2022). Marl-based optimal route control in multi-agv warehouses. *In 2022 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, 333-338.

[24] Sagar, K. V., and Jerald, J. (2022). Real-time automated guided vehicles scheduling with Markov decision process and double Q-learning algorithm. *Materials Today: Proceedings*, 64: 279-284.

[25] Zhang, Z., Chen, J., and Guo, Q. (2023). Application of Automated Guided Vehicles in Smart Automated Warehouse Systems: A Survey. *CMES-Computer Modeling in Engineering & Sciences*, 134(3).

[26] Hu, H., Yang, X., Xiao, S., and Wang, F. (2023). Anti-conflict AGV path planning in automated container terminals based on multi-agent reinforcement learning. *International Journal of Production Research*, 61(1): 65-80.