Car Platoon System: Literature Review and Implementation in Two New Approaches

Hongming Wang[1]

***Abstract*— This paper is mainly concerned with the study of control methodologies of vehicle platooning. A literature review is presented first to help understand some novel control strategies for platoon systems in recent years. To formalize the problem to solve, a model with three cars driving down the same road is established. Then two new approaches, namely, the fuzzy logic approach and the feedforward neural network approach are proposed to control this car platoon system to reach the desired state specified by the user, which serves as the main contribution. Finally, these two approaches are implemented in the simulation experiments to show the validity and effectiveness with a comparison study.**

I. Introduction

With the development of car industry, there are more and more cars driving on the road. The daily traffic load becomes so high that it is necessary to study on this issues and consider about possible solutions.

Platoon based system is one of the significant research directions that can dramatically improve the capacity of road [1]. In [4], the researchers designed a decentralized algorithm for the car-following control (CFC) system in the scenarios of multi-lane traffic to reduce the size of the jam.

Apart from the concerns of improving capacity of the road, [21] emphasized the importance of string stability, which indicates whether the amplitude of vehicles would become large as the evolution of the system.

Spacing policies are found that have strong relationship with string stability in [22], which leads more scholars to pay attention to the study of spacing policies choosing based on different requirements of the system. According to [10], there are two widely used spacing policies, one is constant spacing and the other is called traditional constant highway policy. Different assumptions of the spacing policy will lead to different control strategy and controller design [3].

In the meantime, different approaches to control the platoon system is also a research hotspot. The most of efforts in this paper will be put in this area. First we will review two novel approaches proposed in recent years in terms of related literature, then I will present two original approaches to control a platoon system with three cars.

The remainder of the paper is organized as follows. Section II reviews control methodologies of the vehicle platoon system in terms of consensus control as well as event-triggered control. In section III, firstly, the model of an automatous car platoon system is presented and the problem to solve is formulated. Then a fuzzy logic control approach for the system is proposed. After that, a feedforward neural network approach (FNN) is discussed and designed. Furthermore, the data generated by the first approach are used to train and validate the networks to enable the FNN approach to control the system. In section IV, two test cases are run in both of the two approaches of control. Based on the simulation results, a comparative study is conducted to analyze the outcome of both approaches. Finally, Section V presents the conclusion.

[1]Any questions are welcome, feel free to email me at `h539wang@uwaterloo.ca`

II. Literature Review

In this section, we will focus on two sorts of control strategies for the vehicle platoon system. The first one is called consensus control, which casts the car platoon system into a framework of consensus seeking. The second category named event-triggered control introduces related work of establishing an event-based control scheme on the platoon system model.

A. Consensus Control for Platooning System

In [13], the researchers proposed a method to control the platoon system from a perspective of the networked system. Specifically, the platoon system of a size N is regarded as a network where: (1) every vehicle is represented as a node of the network called a “agent”; (2) the communication link between vehicles using for the transmission of the information such as velocity, distance can be modelled as edges in the network; (3) the set of vehicles (nodes) $\mathcal{V}_N$ and the set of communication links (edges) $\mathcal{E}_N$ together are encoded into the topology of the network $\mathcal{G}_N = (\mathcal{V}_N, \mathcal{E}_N, \mathcal{A}_N)$, where $\mathcal{A}_N = [a_{N,ij}]$ serves as the weighted adjacency matrix of the network topology.

Within this framework, [14] formulizes the dynamics of the leading vehicle of the vehicle platoon as

$$\dot{r}_0 = v_0(t) \tag{1}$$

where $r_0(t)$ and $v_0(t)$ denotes the position and velocity of the pilot vehicle at time t respectively. For the i-th following vehicle $(i = 1, 2, \dots, N-1)$, its dynamics can be described as

$$\begin{aligned} \dot{r}_i &= v_i \\ \dot{v}_i &= u_i \end{aligned} \tag{2}$$

where u_i denotes the acceleration of the i-th follower.

The goal of the system control is to keep the velocity of following vehicles equal to the leader's speed and also maintain the spacing policy between the vehicles nearby. Now the definition of spacing is given by

$$d_i(t) = r_{i-1}(t) - r_i(t) - L_i \tag{3}$$

where L_i denotes the i-th vehicle's length. So the control system's objective is to reach the desired steady state which is described as

$$\begin{aligned} r_i(t) &\to \frac{1}{d_i}\left\{\sum_{j=0}^{N} a_{ij}\cdot(r_j(t)+e_{ij})\right\}, \\ v_i(t) &\to v_0 \end{aligned} \tag{4}$$

where v_0 is the desired velocity, and e_{ij} is the spacing errors between the i-th and the j-th vehicle. Further rewrite the equation (4) in the form of consensus target, we have

$$\begin{aligned} r_i(t) &\to r_0(t)+d_{i1}, \\ v_i(t) &\to v_0, \end{aligned} \tag{5}$$

where $r_0(t)$ denotes the desired trajectory of the pilot vehicle.

Inspired by the work done by [16], the researchers in [13] proposed distributed platooning control protocol for each vehicle to reach the goal of (5) so that the consensus of system is analytical guaranteed, which is given by

$$\begin{aligned} u_i = &-b[v_i(t)-V_0]+\frac{1}{d_i}\sum_{j=0}^{N}k_{ij}a_{ij}\cdot(\tau_i(t)v_0) \\ &-\frac{1}{d_i}\sum_{j=0}^{N}k_{ij}a_{ij}\cdot(r_i(t)-r_j(t-\tau_i(t))-h_{ij}v_0) \end{aligned} \tag{6}$$

where k_{ij} and b are constant positive parameters represent rigidity and damping of the system. Besides, $\tau_i(t)$ is the cumulative transmission delay for the $i-$th vehicle.

B. Event-trigger Platoon System

In the scenarios where the communication between vehicles is not always available or the resources of computation of are limited, we need to implement a new control strategies to the platoon system [6] to reduce unnecessary acceleration or deceleration, as well as communication needs, which is called the event-triggered control scheme refined by [17], [18].

As for the platoon system using event-triggered control strategies, the measurement generated by vehicular sensors is transmitted to the control system only if the expectation of a certain function which takes the current measurement values as the input parameters exceeds the a specific threshold [19] so that an control event occurs to trigger the activation of the control system.

To start with, the spacing error of the $i-$th follower is defined as

$$\delta_i = r_{i-1}-r_i-\delta_{\text{desired}}-L_i. \tag{7}$$

Note that the error evaluation function is based on δ_i. Then following the work of [20] and [6], an event generator is built in between the sensor and the controller. Instead of sampling the state of the $i-$th vehicle to make control decisions to response periodically, in this case, the regularly sampled state are fed to the event generator with a period of h. And then the event generator takes the sampled state at kh to determine whether the state sampled newly (not the state it held at kh) should be directly sent to the controller.

It is proved that [11] the following threshold is appropriate for the vehicle platoon system to ensure the stability of each individual vehicle and the asymptotic error of velocity approaching to zero for all of the vehicles, which is given by:

$$\begin{aligned} &[E\{\rho_s e(i_k h)\}]^T\Omega[E\{\rho_s e(i_k h)\}] < \\ &\mu[E\{\rho_s x(t_k h)\}]^T\Omega[E\{\rho_s x(t_k h)\}], \end{aligned} \tag{8}$$

where $e(i_k h)$ denotes the error between the sampling instant of the least transmitted to the sampling instant corresponding to the computation being conducted, that is

$$e(i_k h) = e(t_k h + lh) = x(t_k h) - x(i_k h). \tag{9}$$

For other meanings of notations, see the work of [11] and [6] for details.

III. Control Strategies in Two New Approaches

The chapter above summarizes several research directions related to platoon control of vehicles. And in this chapter, we will focus on how to construct a model for a car platoon system. Besides, control strategies in two approaches, using fuzzy logic and neural network, will be discussed in detail. Note that the mathematical symbols used in Section II will be redefined or deprecated.

A. Modelling and Problem Formulation

In order to describe the problem clearly, firstly, the system to be implemented in this paper will be illustrated in a brief manner. Meanwhile, I will reformulate the requirements and constraints inside the system into mathematical representations.

Suppose there is a one-dimensional space $\mathcal{S} = [0,\infty)$, which can be regarded as a one-way road. In the road, there are three cars in total, denoted by c_1 (also called the pilot car) ,c_2, c_3, they are moving towards the same direction with the same initial speed v_{ini}. Their coordinate positions are p_1, p_2, p_3 respectively, where

$$p_1 > p_2 > p_3$$

is always satisfied. Assume c_3 's initial position is the origin of road $\mathcal{S}$:

$$p_{3_{\text{ini}}} = 0.$$

With regard to the initial positions for c_1 and c_2, we have

$$p_{1_{\text{ini}}} = p_{2_{\text{ini}}} + d_{\text{initial}_1}, d_{\text{initial}_1} > 0 \tag{10}$$

and

$$p_{2_{\text{ini}}} = p_{3_{\text{ini}}} + d_{\text{initial}_2}, d_{\text{initial}_2} > 0, \tag{11}$$

where d_{initial_1} is the distance between c_1 and c_2, and d_{initial_2} is the distance between c_2 and c_3. Both d_{initial_1} and d_{initial_2} are specified by the user.

Now, let us talk about the goal of the control system. Once the system starts running, the c_1 will adjust its acceleration a_1 to let its velocity v_1 approach the desired velocity v_{desired}

specified by the user until the desired state $v_1 = v_{\text{desired}}$ is reached,

$$a_1 = f(v_1, v_{\text{desired}}) = \begin{cases} \text{accelerate} & \text{if } v_1 < v_{\text{desired}} \\ 0 & \text{if } v_1 = v_{\text{desired}} \text{ the goal}_1 \\ \text{brake} & \text{if } v_1 > v_{\text{desired}} \end{cases} \tag{12}$$

where $f(\cdot)$ is the control strategy need to be determined for c_1.

The goal for c_2 is no other than adjusting its acceleration a_2 based on the observation of the dynamic distance d_1 from the pilot car c_1 and its velocity v_1 to match the desired distance d_{desired} specified by the user.

$$\begin{aligned} &a_2 = g(d_1, d_{\text{desired}}, v_2, v_1) \\ &\text{goal}_2\text{: } d_1 = d_{\text{desired}} \text{ and } v_2 = v_1, \end{aligned} \tag{13}$$

where $g(\cdot)$ denotes the control method need to be determined for c_2. Similarly, for c_2 we have the same goal except that the front car of c_3 is c_2 instead of c_1,

$$\begin{aligned} &a_3 = g(d_2, d_{\text{desired}}, v_3, v_2). \\ &\text{goal}_3\text{: } d_2 = d_{\text{desired}} \text{ and } v_3 = v_2. \end{aligned} \tag{14}$$

Note that c_2 and c_3 share the same rules of control $g(\cdot)$ from equation (13), (14). From the above description, the communication topology can be constructed as is shown in Fig. 1.

$$\begin{tikzcd} \bullet_{c_3} & \bullet_{c_2} \arrow[l, "{d_2,\, v_2}"'] & \bullet_{c_1} \arrow[l, "{d_1,\, v_1}"'] \end{tikzcd}$$

Fig. 1: The communication topology of the system.

Combine the initial states of three cars together with equations (12), (13), (14), we can obtain the initial and desired state of the system:

$$\begin{aligned} \text{Initial state: } & a_1 = a_2 = a_3 = 0, \\ & v_1 = v_2 = v_3 = v_{\text{ini}}, \\ & d_1 = d_{\text{initial}_1},\ d_2 = d_{\text{initial}_2}. \end{aligned} \tag{15}$$

$$\begin{aligned} \text{desired state: } & v_3 = v_2 = v_1 = v_{\text{desired}}, \\ & d_1 = d_2 = d_{\text{desired}}. \end{aligned} \tag{16}$$

The aforementioned discussion presents an overview of the system from an external perspective. Now we will exploit the dynamics of system internally. From the definition of acceleration, we have the update rule of every car's velocity:

$$\begin{aligned} v_{1_{\text{new}}} &= v_{1_{\text{old}}} + a_1 \Delta t \\ v_{2_{\text{new}}} &= v_{2_{\text{old}}} + a_2 \Delta t \\ v_{3_{\text{new}}} &= v_{3_{\text{old}}} + a_3 \Delta t \end{aligned} \tag{17}$$

where the time interval Δt is equal to one unit time of the system. With changing velocity and nonzero acceleration taken into account, the overall displacement of each car follows that

$$p_i(t + \Delta t) = p_i(t) + v_i(t)\Delta t + \frac{1}{2} a_i^2 (\Delta t) \tag{18}$$

from t to $t + \Delta t$, where $i = 1, 2, 3$.

Now the system I am to work on is introduced quite clearly, next in order to design the control algorithms as the main contributions of this paper, I will present two approaches for that.

B. Fuzzy Logic Approach

In this subsection, an approach using fuzzy logic to control the system will be presented. Specifically, Sugeno fuzzy model is used as the fuzzy inference system for the car platoon system.

The general processes of applying fuzzy logic control to the car platoon system are as follows: Firstly, value-based measurements such as velocity, acceleration, distance are fuzzified as corresponding fuzzy variables. Secondly, design a proper membership function for each fuzzy variables so that variables with any values can be mapped into one of the five linguistic states. Thirdly, the rules of rule base fuzzy for inference are summarized and presented. Lastly, the inference system aggregates the output of each rules and defuzzify the acceleration fuzzy variable using the computation result of normalized weighted arithmetic mean, then the control decision is generated and influence the velocity of corresponding car. Fig. 2 shows the illustration of the inference system:

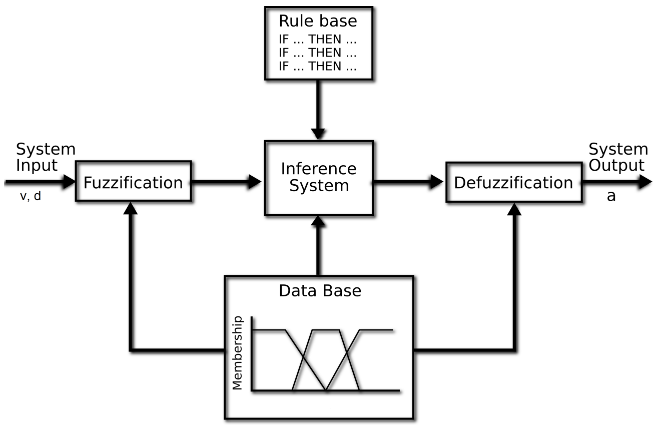


Fig. 2: General processes of fuzzy inference control system.

In the next step, the detail of each step will be showed one by one.

The measurements needed to be redefined to be used for inference system are the c_i's current velocity (*VEL*$_i$, $i = 1, 2, 3$), relative distance from the front car c_{i-1} (*DIS*$_i$, $i = 2, 3$), and acceleration (*ACC*$_i$, $i = 1, 2, 3$). Each type of fuzzy variables has multiple possible values.

Car c_i's velocity *VEL*$_i$ ($i = 1, 2, 3$) takes five fuzzy states: very slow (*VS*), slow (*SL*), just right (*JR*), fast (*FS*), very fast (*VF*). The membership functions of *VEL*$_i$ are defined in the support set $[v_o - 16, v_o + 16]$, where v_o denotes the target velocity. Note that for *VEL*$_1$, $v_o = v_{\text{desired}}$; for *VEL*$_2$, $v_o = v_1$; for *VEL*$_3$, $v_o = v_2$. The types of these membership functions are either triangular or trapezoidal, which are described by:

$$\begin{aligned}\mu_{\text{VS}}(v) &= 1 && \text{for } \textit{VEL}_i = (-\infty, v_o - 16) \\ &= \text{linear}[1.0, 0] && \text{for } \textit{VEL}_i = [v_o - 16, v_o - 8] \\ &= 0 && \text{Otherwise.}\end{aligned} \tag{19}$$

$$\begin{aligned}\mu_{\text{SL}}(v) &= \text{linear}[0, 1.0] && \text{for } \textit{VEL}_i = [v_o - 16, v_o - 8] \\ &= \text{linear}[1.0, 0] && \text{for } \textit{VEL}_i = [v_o - 8, v_o] \\ &= 0 && \text{Otherwise.}\end{aligned} \tag{20}$$

$$\begin{aligned}\mu_{\text{JR}}(v) &= \text{linear}[0, 1.0] && \text{for } \textit{VEL}_i = [v_o - 8, v_o] \\ &= \text{linear}[1.0, 0] && \text{for } \textit{VEL}_i = [v_o, v_o + 8] \\ &= 0 && \text{Otherwise.}\end{aligned} \tag{21}$$

$$\begin{aligned}\mu_{\text{FS}}(v) &= \text{linear}[0, 1.0] && \text{for } \textit{VEL}_i = [v_o, v_o + 8] \\ &= \text{linear}[1.0, 0] && \text{for } \textit{VEL}_i = [v_o + 8, v_o + 16] \\ &= 0 && \text{Otherwise.}\end{aligned} \tag{22}$$

$$\begin{aligned}\mu_{\text{VF}}(v) &= \text{linear}[0, 1.0] && \text{for } \textit{VEL}_i = [v_o + 8, v_o + 16] \\ &= 1 && \text{for } \textit{VEL}_i = [v_o + 16, \infty) \\ &= 0 && \text{Otherwise.}\end{aligned} \tag{23}$$

In Fig. 3 I draw all these membership functions in the same graph:

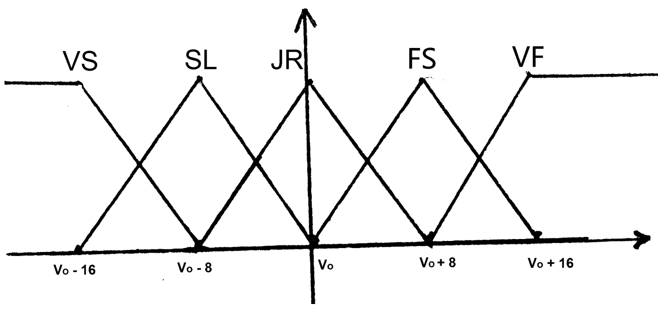


Fig. 3: Membership functions for all states of VEL_i.

The membership functions of $\textit{DIS}_i$ $(i = 2, 3)$ are defined based on proportion to the desired length d_{desired}. Because in the project description that Professor Karray sent to me, it is suggested that one benchmark for testing should be the case that $d_{\text{initial}_1} = 3d_{\text{desired}}$ and $d_{\text{initial}_2} = 5d_{\text{desired}}$. So I was inspired by that and not chose a value-based way to distinguish fuzzy states.

Car c_i's $\textit{DIS}_i$ $(i = 2, 3)$ takes five fuzzy states: very far (TF), far (FR), just right (JE), close (CL), very close (VC). The membership functions of $\textit{DIS}_i$ are defined in the support set $[0, \infty)$ pixels, where the term *pixel* is length unit of the screen. Related membership functions are described by:

$$\begin{aligned}\nu_{\text{TF}}(d) &= 1 && \text{for } \textit{DIS}_i = [4d_{\text{desired}}, \infty) \\ &= \text{linear}[0, 1.0] && \text{for } \textit{DIS}_i = [2d_{\text{desired}}, 4d_{\text{desired}}] \\ &= 0 && \text{Otherwise.}\end{aligned} \tag{24}$$

$$\begin{aligned}\nu_{\text{FR}}(d) &= \text{linear}[0, 1.0] && \text{for } \textit{DIS}_i = [d_{\text{desired}}, 2d_{\text{desired}}] \\ &= \text{linear}[1.0, 0] && \text{for } \textit{DIS}_i = [2d_{\text{desired}}, 4d_{\text{desired}}] \\ &= 0 && \text{Otherwise.}\end{aligned} \tag{25}$$

$$\begin{aligned}\nu_{\text{JE}}(d) &= \text{linear}[0, 1.0] && \text{for } \textit{DIS}_i = [0.75d_{\text{desired}}, d_{\text{desired}}] \\ &= \text{linear}[1.0, 0] && \text{for } \textit{DIS}_i = [d_{\text{desired}}, 2d_{\text{desired}}] \\ &= 0 && \text{Otherwise.}\end{aligned} \tag{26}$$

$$\begin{aligned}\nu_{\text{CL}}(d) &= \text{linear}[0, 1.0] && \text{for } \textit{DIS}_i = [0.5d_{\text{desired}}, 0.75d_{\text{desired}}] \\ &= \text{linear}[1.0, 0] && \text{for } \textit{DIS}_i = [0.75d_{\text{desired}}, d_{\text{desired}}] \\ &= 0 && \text{Otherwise.}\end{aligned} \tag{27}$$

$$\begin{aligned}\nu_{\text{VC}}(d) &= \text{linear}[1.0, 0] && \text{for } \textit{DIS}_i = [0.5d_{\text{desired}}, 0.75d_{\text{desired}}] \\ &= 1 && \text{for } \textit{DIS}_i = [0, 0.5d_{\text{desired}}] \\ &= 0 && \text{Otherwise.}\end{aligned} \tag{28}$$

In Fig. 4 I draw all these membership functions in the same graph:

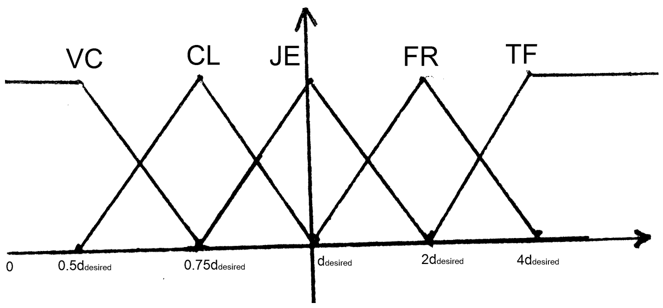


Fig. 4: Membership functions for all states of DIS_i.

Since Sugeno-type is integrated to the inference system, the control output part is designed to keep constant within one state. So we have car c_i's $\textit{ACC}_i$ $(i = 1, 2, 3)$ takes five fuzzy states, and each state corresponds to a constant value: brake hard (BH, $a_i = -4$), brake (BR, $a_i = -2$), none (NO, $a_i = 0$), accelerate (AC, $a_i = 2$), accelerate hard (AH, $a_i = 4$).

In order to make control rules for the system, we need to analyze their scopes. From the equation (12), (13), (14) and the analysis of Section II, we know that the control strategies for c_1 is different from c_2 and c_3, where as c_2 and c_3 share the same rules. So the rules of fuzzy logic for the pilot car c_1 are designed separately. The following three rules are used as the rule base for controlling c_1:

	If	VEL_1	is	VS	then	ACC_1	is AH
else	if	VEL_1	is	SL	then	ACC_1	is AC
else	if	VEL_1	is	JR	then	ACC_1	is NO
else	if	VEL_1	is	FS	then	ACC_1	is BR
else	if	VEL_1	is	VF	then	ACC_1	is BH

Rule base for c_2 and c_3 is a little bit more complicated, because we also need to take the variable $\textit{DIS}_i$ $(i = 2, 3)$ into account. These rules are showed in the table below:

TABLE I: Control rules for the following car $c_i(i = 2, 3)$.

$\downarrow$VEL$_i$ $\searrow$ACC$_i$ $\rightarrow^{\text{DIS}_i}$	VC	CL	JE	FR	TF
VS	NO	NO	AC	AC	AH
SL	NO	NO	AC	AC	AH
JR	BH	BR	NO	AC	AH
FS	BH	BR	BR	NO	NO
VF	BH	BH	BH	BR	NO

In contrast with Mamdani fuzzy inference which mainly uses *min* operation on the grades of membership funcions for fuzzy implication, here we apply fuzzy operation *OR* (*OR* = *max*) to calculate the length of firing strength w_j of each corresponding rule, if there are multiple input variables, which is describe by:

$$w_{ij} = \textit{ORMethod}(\mu_{ij}(v), \nu_{ij}(d)), (i = 2, 3). \tag{29}$$

And for pilot car c_1, there is just one single input variable, which is describe by:

$$w_{1j} = \mu_{1j}(v), \tag{30}$$

where j ($j \in \mathcal{N}$) in equations (30) and (29) denotes a specific rule which may contribute to the output of the system.

Then by the calculation of normalized weighted average we can obtain the output of the inference system for each car:

$$a_{i_{\text{output}}} = \frac{\sum_{j\in\mathcal{N}} w_i z_i}{\gamma \sum_{j\in\mathcal{N}} w_i} = \sum_{j\in\mathcal{N}} w_i z_i \tag{31}$$

where $z_i(i = 1, 2, 3)$ is the output level of acceleration for c_i. In this paper, we choose a zero-order Sugeno such that z_i is a constant value in the set $\{-4, -2, 0, 2, 4\}$, which is specified by:

$$z_i = \begin{cases} -4 & \text{if ACC}_i \text{ is BH} \\ -2 & \text{if ACC}_i \text{ is BR} \\ 0 & \text{if ACC}_i \text{ is NO} \\ 2 & \text{if ACC}_i \text{ is AC} \\ 4 & \text{if ACC}_i \text{ is AH} \end{cases}. \tag{32}$$

And γ in (31) is the tuning parameter such that

$$\gamma \sum_{j\in\mathcal{N}} w_i = 1 \tag{33}$$

is guaranteed, such that after aggregation, the output will not exceed the stable range.

The main framework of the approach using fuzzy logic is as stated above. Now let us consider about some issues and their solutions related to the performance of the system that had encountered during the simulation in the early stage.

In several previous versions of control strategies for c_2 and c_3, their velocity and their front cars' velocities were not originally taken into account for control rules, which followed the project description. However, in practical simulations, I found that except the pilot car c_1 reaches the expect state, other 2 cars would run back and forth around c_1, which is not allowed by traffic laws and would cause severe accidents. And what is worse, the amplitudes of both cars would become larger and larger. In other words, the system is unstable even theoretically. Therefore, in order to fix this phenomenon, the constraints of related velocities are added to c_2 and c_3's rule bases. As a result, the system becomes stable, the ultimate distances between cars are within the a small range around the desired distance. In addition, during the process of control, each car will never contact with other cars physically.

Another issue is that jitter occurred periodically in the control output $a_{2_{\text{output}}}$ and $a_{3_{\text{output}}}$ so that d_1 and d_2 would escape the desired distance d_{desired} abruptly in every time interval with fixed length. To investigate the source of this problem, the control rules of the cases where VEL$_i$ is close to JR and DIS$_i$ ($i = 2, 3$) is nearly JE for c_2 and c_3 are checked carefully. I found that the jitter happened right after the time when d_1 or d_2 exceeded the desired distance d_{desired} (it was also sure before that moment $v_i < v_{i-1}(i = 2, 3)$ a bit). Thus the code segment for that case (the wiring strength of which is denoted by w_i*) is optimized and the form of the pseudocode is given by:

```
**if** $v_{i-1} - 8 \le v_i < v_{i-1}$ and $d_{\text{desired}} \le d_i < 2d_{\text{desired}}$ **then**
    $z_i$ = 2
    **if** $v_o - v_i < \varepsilon$ **then**
...     $w_i* = \eta\ max(\mu_{\text{SL}}(v_i), \nu_{\text{FR}}(d_i))$
```

where ε is the threshold to enter into the optimization algorithm, normally $\varepsilon < 0.1$. η denotes coefficient to soften the control output eliminate the jitter.

C. Neural Network Approach

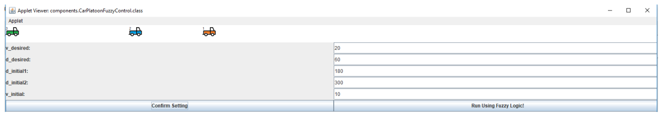
Up to now, the approach using fuzzy logic to control the system is completely described. And its simulation results went very well, which will be presented in Section IV. After the quality of the output of the fuzzy system is ensured to be high, it is proper to begin neural network training.

Firstly, we must figure out how many types of neural networks in total we should train. The answer is 2. Similarly as the previous approach, we know that there are two categories of control strategies, one is for pilot car c_1, the other one is for following cars c_2 and c_3. So we should train one neural network for c_1, called NN$_1$; and one neural network for c_2, called NN$_2$.

For NN$_1$, it takes two parameters as the input, v_1 and v_{desired}. Its output is the single a_1. For NN$_2$, it takes four parameters as the input, v_i, v_{i-1}, $d_{(i-1)}$, d_{desired}, and a_i as the only output.

To increase the accuracy of the neural network, one valid method is to increase to quantity of training data and test data at the same time. Thus instead of extracting 1,000 data points as suggested, I will feed a quantity of 5,000 data points to NN$_1$, and 10,000 data points to NN$_2$. The reason why the data set of NN$_2$ is twice as large as that of NN$_1$'s is because c_2 and c_3 share the same control strategies and they will share the same neural network NN$_2$. So the data points generated by c_3 are also eligible for training and testing NN$_2$. In every simulation, c_2 and c_3 together will produce twice as large quantity of data as c_1 does.

To determine how many valuable data points can be generated in a single simulation, the simulation program is run first with the parameters which meet the specification of the benchmark, showed as in Fig. 5. Note that several



v_desired: 20
d_desired: 60
d_initial1: 180
d_initial2: 300
v_initial: 10

Fig. 5: One possible parameter setting for the benchmark is: $v_{\text{desired}} = 20, d_{\text{desired}} = 60, d_{\text{initial}_1} = 180, d_{\text{initial}_2} = 300, v_{\text{desired}} = 10$

.

arrays should be maintained to store the value (including $v_i, a_i (i = 1, 2, 3)$ and $d_j (j = 1, 2)$ recorded at each step. After nearly 95 time steps, $v_1 = v_2 = v_3 = v_{\text{desired}} = 20$ and $d_2 \approx d_1 = 20$, we can assert that the system reaches the desired state.

This number of steps indicates that a quantity of 10,000 data points cannot be generated in a single simulation. Because after the desired state is reached, the control output is nearly zero, and a lot of intensive data points in this area are trivial to influence the behavior of the neural network.

To collect enough non-trivial data points with even distribution, a simple way is to run the simulation program multiple times with different parameters, group and merge all these data points into one file as the training and testing input for corresponding neural network. The following rules are used to generate parameters as the input of the simulation program to cover the proper cases as broad as possible:

$$\begin{aligned}
&v_{\text{desired}} \in [5, 30] \text{ in } \mathbb{R};\\
&d_{\text{desired}} \in [50, 100] \text{ in } \mathbb{R};\\
&d_{\text{initial}_1} = \alpha_1 d_{\text{desired}}, \alpha_1 \in [2, 5] \text{ in } \mathbb{R};\\
&d_{\text{initial}_2} = \alpha_2 d_{\text{desired}}, \alpha_2 \in [2, 5] \text{ in } \mathbb{R};\\
&v_{\text{desired}} \in [0, 30] \text{ in } \mathbb{R};
\end{aligned} \tag{34}$$

60 sets of parameters are generated using rules of (34), and are input to the simulation program. During the runtime, we observe the number of time steps needed n_{stable} to reach the desired state from the very beginning of every simulation for the corresponding set of parameters. From these sets of parameters, I select 40 of them whose simulations satisfy the requirement $n_{\text{stable}} \in [80, 120]$ as the data sets for NN_1 and NN_2. The simulation of every selected set of parameters will contribute a number of 125 data points generated since the very beginning to the data sets. In this way, we have $125 \times 40 \times 1 = 5000$ data points for training and testing NN_1, and $125 \times 40 \times 2 = 10000$ data points for training and testing NN_2.

As for the structure of the neural network, it is quite challenging to demonstrate that one design is the best of all. But we can focus on some detailed consideration to figure out a reasonable neural network design.

Feedforward neural network (FNN) allows the input signals travel in one direction, that is from input to output. It is quite a simple way to find a way of associating inputs with outputs. Theoretically, it is able to solve the function approximation problems with an appropriate accuracy, which applies to function mapping tasks in this paper.

On the other hand, Recurrent Neural Network (RNN) is also a powerful tool, which is very popular in research recently. This class of neural network has a mechanism that allows connections to form directed circles so that information can be passed to and stored in corresponding neurons. In addition, RNN has the ability of keeping memory of the previous input by using its neurons' memory units, which performs well in sequential tasks such as time series prediction. However, for the system in this paper, we assume that no delay will happen in the communication. Therefore, there seems to be no strong need to use RNN for this problem, and we will simply use the feedforward neural network as the basic architecture for NN_1 and NN_2.

With regard to the number of layers, my thinking is that to ensure the accuracy of the function approximation, at least one hidden layer is needed for both NN_1 and NN_2. Thus NN_1 and NN_2 both have three layers, namely, one input layer, one hidden layer, and one output layer.

For the choice of hidden activation function, given that the output of NN_i $(i = 1, 2)$ can be positive (accelerate) or negative (brake), I choose a rescaling version of *sigmoid* function called *hyperbolic tangent* function, denoted by "*tanh*" and defined as

$$tanh(x) = 2 \cdot \sigma(2x) - 1, \tag{35}$$

where $\sigma(x)$ denotes the *sigmoid* function, and is give by

$$\sigma(x) = \frac{e^x}{1 + e^x}. \tag{36}$$

Now it is time to determine the number of neurons in each layer. For NN_1, it is a 2-input 1-output system. And bias should also be added to the input layer and the hidden layer. So for the input layer, there are $2 + 1 = 3$ neurons; for the hidden layer, there are $3 + 1 = 4$ neurons (note that the number 3 in the left hand side is determined by evaluation of the complexity of $f(\cdot)$ in equation (12)); for the output layer, there is one neuron.

Similarly, for NN_2, it is a 4-input 1-output system. One bias neuron is added to the input layer, and the other one is added to the hidden layer. With these extra bias neurons taken into account, for the input layer, there are $4 + 1 = 5$ neurons; for the hidden layer, there are $7 + 1 = 8$ neurons (note that the number 7 in the left hand side is determined by evaluation of the complexity of $g(\cdot)$ in equations (13),(14)); for the output layer, there is one neuron.

Finally, we reach the stage of feeding data to the model to train and test NN_1 and NN_2. In order to be compatible with the simulation program based on Java applet, and reuse the code of previous work as much as possible, I choose a machine learning framework called "*Encog*", which is available in Java language environment. Moreover, it is worth mentioning that *Encog* provides a mechanism to automatically shuffle the data, and divide them into two sets , one for training and the other one for validation. In this paper, in order to specify 70% of the data points for training, and the other 30% for testing and validation, the only thing we need to do is typing one line of code:

```
model.holdBackValidation(0.3, true, 1001);
```

where the first parameter 0.3 specify 30% of the data points for testing and validation. Note that the last parameter 1001 is not a magic number but a fixed random seed which ensures the repeatability of the training and testing. In order to avoid the phenomena that the networks over-fit the training sets, a k-fold cross-validation of size 5 is performed. And it is easy to achieve this in one line with *Encog*:

```
bestMethod = (MLRegression)
                model.crossvalidate(5, true);
```

where the object called `bestMethod` contained the well-tuned weights of the neural network. And these weights are integrated to the structures of NN_1 and NN_2 respectively, see Fig. 6 and Fig. 7, which are served as the final models of NN_1 and NN_2. Note that the number of

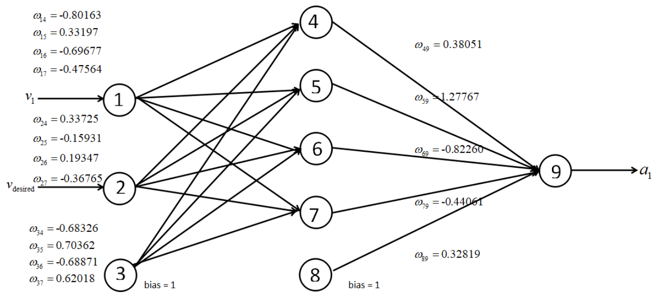


Fig. 6: Final FNN model with tuned weights for NN_1.

weights for NN_2 are too large to display, please see the file `NeuralNetworkModels.pdf` for the complete details.

Other than the discussion above which is based on the architecture of feedforward neural network, a comparative test is also run to show the performance of other types of neural network, such as radial basis function (RBF) neural network, NEAT neural network, support vector machines (SVM). Detailed structures of these networks are determined by *Encog* with optimization by default. Performance indicators used including training time[1], training error and validation error. These results are average values for 3 replicates.

TABLE II: Performance comparison of different approaches to implement NN_1.

↓Indicators →NN_1 approach	FNN	RBF	NEAT	SVM
training time (ms)	981	417	48005	4405
Training error	2.26E-4	1.83E-3	7.69E-5	1.76E-5
Validation error	1.78E-4	6.56E-3	6.71E-5	3.09E-5

[1]Here training time means the time of k-fold cross-validation of size 5.

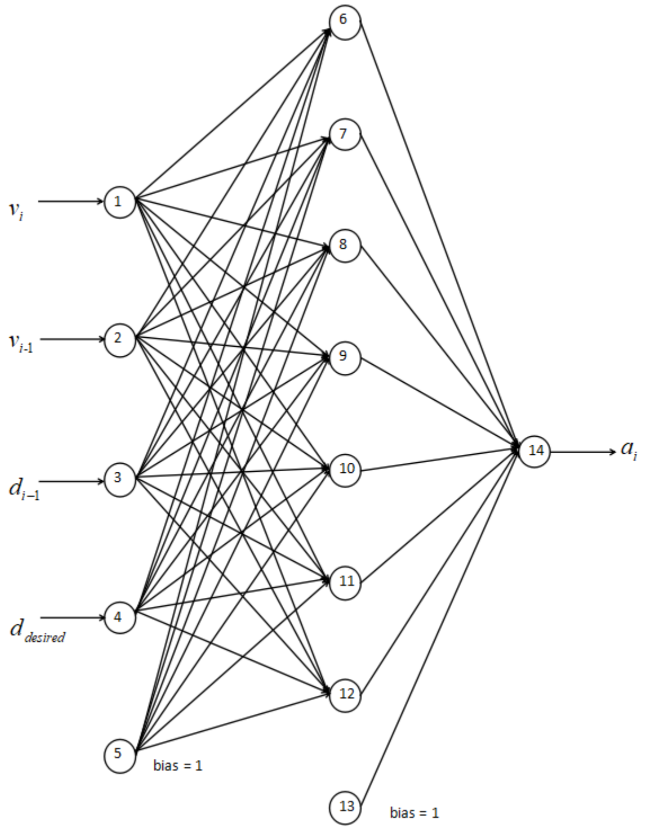


Fig. 7: Final FNN model for NN_2.

TABLE III: Performance comparison of different approaches to implement NN_2.

↓Indicators →NN_2 approach	FNN	RBF	NEAT	SVM
Training time (ms)	3561	6808	54650	24910
Training error	1.83E-2	1.76E-2	1.42E-2	9.45E-3
Validation error	1.80E-4	1.66E-2	1.44E-2	9.57E-3

From these two tables we can see that NN_1 and NN_2 can achieve the highest accuracy using SVM, while FNN takes the shortest time for training and validation, and also gets a proper accuracy, which makes no big difference in practical control prediction. So in Section IV we will continue using FNN-based neural networks to present the results of simulation.

IV. Simulation and Results

Simulation experiments of the car platoon system controlling have been carried out using fuzzy logic approach as well as feedforward neural network approach. Main results of them will be presented in this section.

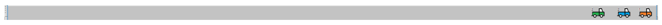
Roll-back display mechanism in GUI

Since some screenshots of Java applet GUI of simulations will be showed to illustrate the processes of stabilizing, so it is necessary to understand the mechanism of how to catch three running cars and display them in the PC screen with a finite width (screen width = 1600 pixels for my PC). I came up with a method to solve this issue perfectly, named roll-back display mechanism.

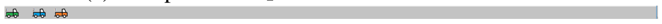
As a matter of fact, we are concerned only with changes of relative position of each car. And in practical simulations of the car platoon system, I find that if parameters are chosen according to equation (34), then $p_1 > p_2 > p_3$ is held at every time step. So we only need to display their relative positions in the GUI. Specifically, when the pilot car c_1 reaches the end of the screen on the other side, i.e. $p_i > 1600$, then the roll-back mechanism will relocate three cars' positions in the screen using the rule given by

```
if $p_i > 1600$ then
    diff = $p_3 \% 1600 - 0$
    $p_3 = 0, p_2 = p_2 - \text{diff}, p_1 = p_1 - \text{diff}$
    Display $c_3$ at $p_3$, $c_2$ at $p_2$, $c_1$ at $p_1$
```

Fig. 8a, 8b show an example of how the roll-back display mechanism works.



(a) The pilot car c_1 reaches the end of the screen.



(b) Then three cars' positions are relocated.

Fig. 8: Illustration of roll-back display mechanism.

Now the results of two sets of parameters will be specified as the initial conditions of the car platoon system. The first one meets the specification of benchmark, the other one is generated using the rule of equation (34). Both of them is neither from training set nor from validation set. They are specifically used for testing. The table below shows all of the elements of these two sets.

TABLE IV: Sets of initial conditions for testing.

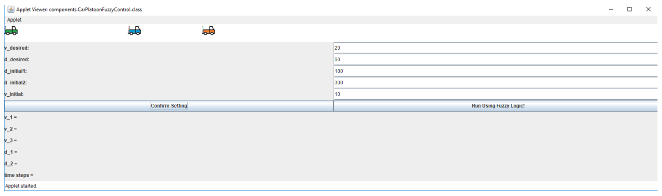
↓No.→Initial conditions	v_{desired}	d_{desired}	d_{initial_1}	d_{initial_2}	v_{ini}
1 (benchmark)	20	60	180	300	10
2	12	71	263	149	18

Testing of the fuzzy logic approach

Firstly, the fuzzy logic part of the simulation is run with the initial conditions of benchmark, as is shown in Fig. 9a, 9b, 9c, 9d. These figures present four representative states during the process of control. Note that the simulation program also traces system variables in real time, and their values are displayed in the lower part of GUI.

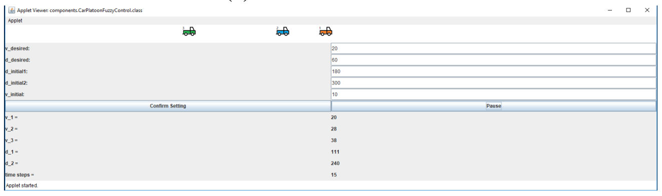
From these states, we can see that the system responses quickly and makes right decisions to approach the targeted state in a short time. The pilot car c_1 can reach its desired state before $t = 15$, and c_2 reaches the desired speed around $t = 30$. After that, only small adjustments are needed to make, and the system stabilizes around the desired state at $t = 97$ or so.

Fig. 10, 11 record the changes of v_i and d_{i-1} in a larger time span ($t \in [0, 124]$). These figures show that both v_i and d_{i-1} will after several moderate fluctuations and stabilize



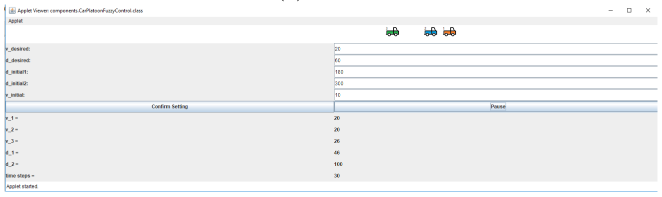
```
Applet Viewer: components.CarPlatoonFuzzyControl.class
Applet
v_desired:      20
d_desired:      60
d_initial1:     180
d_initial2:     300
v_initial:      10
Confirm Setting      Run Using Fuzzy Logic!
v_1 =
v_2 =
v_3 =
d_1 =
d_2 =
time steps =
Applet started.
```

(a) Initial state $t = 0$.



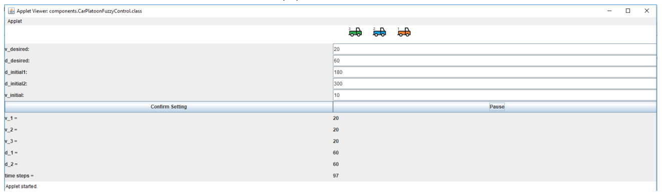
```
Applet Viewer: components.CarPlatoonFuzzyControl.class
Applet
v_desired:      20
d_desired:      60
d_initial1:     180
d_initial2:     300
v_initial:      10
Confirm Setting      Pause
v_1 =           20
v_2 =           28
v_3 =           38
d_1 =           111
d_2 =           240
time steps =    15
Applet started.
```

(b) $t = 15$.



```
Applet Viewer: components.CarPlatoonFuzzyControl.class
Applet
v_desired:      20
d_desired:      60
d_initial1:     180
d_initial2:     300
v_initial:      10
Confirm Setting      Pause
v_1 =           20
v_2 =           20
v_3 =           26
d_1 =           46
d_2 =           100
time steps =    30
Applet started.
```

(c) $t = 30$.



```
Applet Viewer: components.CarPlatoonFuzzyControl.class
Applet
v_desired:      20
d_desired:      60
d_initial1:     180
d_initial2:     300
v_initial:      10
Confirm Setting      Pause
v_1 =           20
v_2 =           20
v_3 =           20
d_1 =           60
d_2 =           60
time steps =    97
Applet started.
```

(d) $t = 97$, the desired state has been reached.

Fig. 9: Fuzzy logic control process of benchmark in GUI.

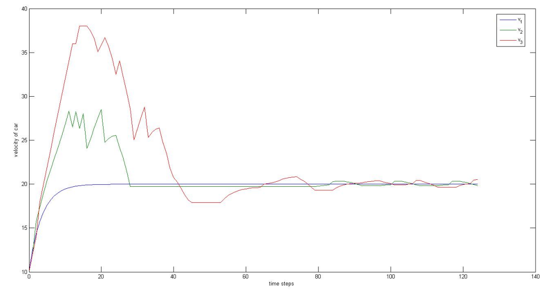


Fig. 10: The changing curves of v_i in the benchmark using the fuzzy logic approach.

around the desired values, which demonstrate the stability of the system from another angle.

Besides, the second test case is also presented in order to have a much clearer picture of this fuzzy logic approach. See Fig. 12, 13, the time span chosen for plotting is also ($t \in [0, 124]$).

Testing of the feedforward neural network approach

Secondly, the FNN part of the simulation is run with same sets of initial conditions. Again, we will run the benchmark first. Due to the limited space, results are presented briefly in the form of changing curves, see Fig. 14, 15.

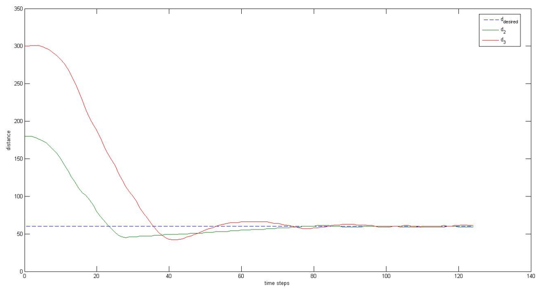


Fig. 11: The changing curves of d_{i-1} in the benchmark using the fuzzy logic approach.

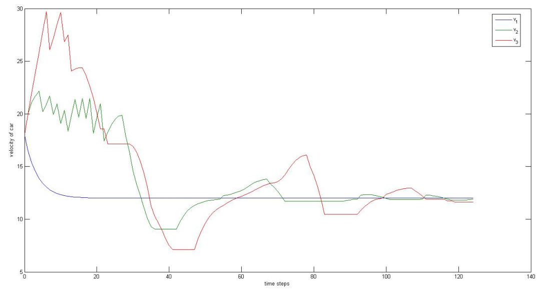


Fig. 12: The changing curves of v_i in the second test case using the fuzzy logic approach.

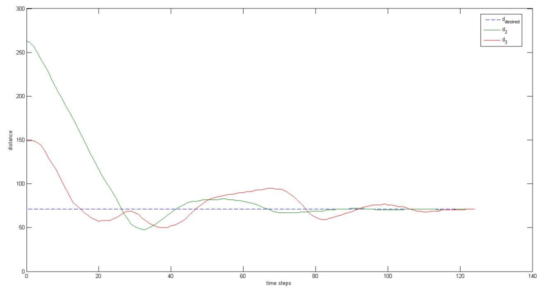


Fig. 13: The changing curves of d_{i-1} in the second test case using the fuzzy logic approach.

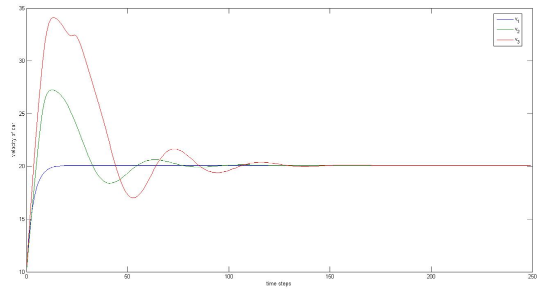


Fig. 14: The changing curves of v_i in the benchmark using the FNN approach.

Note that after multiple times of simulation, I find that the FNN approach takes more time to reach its steady state normally, so the time span is extended to $t \in [0, 250]$. Moreover, in Fig. 15 we can see that the steady state error of d_{i-1} is 4.

A remarkable feature of the changing curves of the FNN approach is relatively low volatility, as shown with the second test case run in Fig. 16, 17. We can see the curves of v_i and d_{i-1} change more gently. Also note that the steady state error of d_{i-1} is -3.

Comparison and Analysis

From the results above, there are several attractive aspects of control performance from two approaches that can be compared and analyzed.

First of all, running the simulation using the FNN approach multiple times with the same initial conditions will get a set of results (such as changing curves of d_{i-1}), and surprisingly, each of them slightly differs from the other. However, if we do the same thing using fuzzy logic approach, the output will always remain constant, only if the initial conditions are the same. The reason of these phenomena is that the neural network initializes its weights of connections with random values in a range from 0 to 1.

Secondly, from Fig. 10, 12 we can see the drastic rate of changes of the v_i (i.e. the changes of control output) clearly in the early stage of fuzzy logic control process. The degree of change of control output is so high that we can even see that there are many saw-teeth in the curves. In addition, it is also the main cause for the endless small vibration around the desired state in the later stage. In contrast, the curves in Fig. 14, 16 are very smooth. This indicates that the neural network is a more sleeker approach in terms of the change rate of the system outcomes. The reason is because the neural network approach figures out a way of curves fitting with nearly minimum error. The nature of fitting guarantees its sleekness.

Though every time of running simulation using the FNN approach will come out a slight different result, the average time for the system to stabilize around the desired state is far longer than the one using fuzzy logic approach, usually more than twice the time. Therefore, in terms of rapidity, the FNN approach is not as good as the fuzzy logic approach.

With regard to accuracy, as is shown in Fig. 15, 17, there are non-negligible steady state errors, positive or negative, between the ultimate values of d_{i-1} and the desired distances. However, for the fuzzy logic approach, though it has small vibration around the desired state, the overall error can be ignored. In other words, the fuzzy logic approach is more accurate than the FNN approach to reach the targeted state.

V. Conclusion

In this study, a promising driving pattern called the car platoon system is investigated. First, two novel approaches to control the vehicle platoon are reviewed in terms of related literature. Then two original approaches, the fuzzy logic approach and the feedforward neural network approach, are designed to control the system to reach the desired

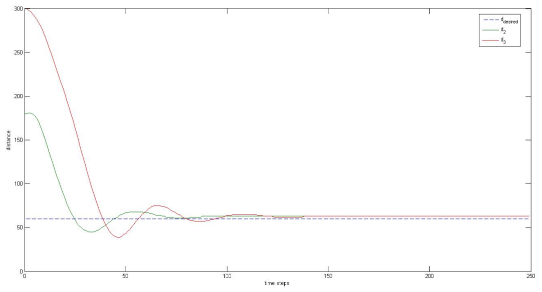


Fig. 15: The changing curves of d_{i-1} in the benchmark using the FNN approach.

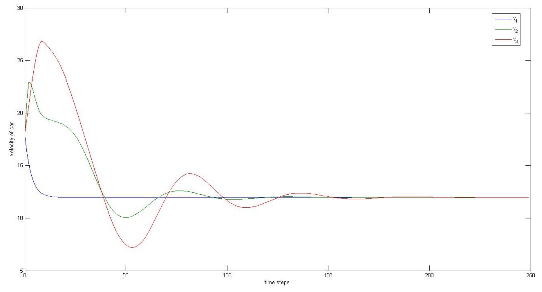


Fig. 16: The changing curves of v_i in the second test case using the FNN approach.

state. Thirdly, a set of comparative simulations are run to show the validity and effectiveness of proposed approaches. In addition, the results are analyzed in terms of control performance. Finally, based on the contribution of this paper, the most wanted challenge to be solved in the future is using the recurrent neural network to control the platoon system with time-varying delays.

REFERENCES

[1] Jia, Dongyao, et al. "A survey on platoon-based vehicular cyber-physical systems." IEEE Communications Surveys & Tutorials 18.1 (2016): 263-284.

[2] Demir, Ozan, and Jan Lunze. "Event-based synchronisation of multi-agent systems." IFAC Proceedings Volumes 45.9 (2012): 1-6.

[3] Li, Shengbo Eben, et al. "Platoon Control of Connected Vehicles from a Networked Control Perspective: Literature Review, Component Modeling, and Controller Synthesis." IEEE Transactions on Vehicular Technology (2017).

[4] Wang, Meng, et al. "Cooperative car-following control: Distributed algorithm and impact on moving jam features." IEEE Transactions on Intelligent Transportation Systems 17.5 (2016): 1459-1471.

[5] Zegers, Jeroen C., et al. "Consensus Control for Vehicular Platooning With Velocity Constraints." IEEE Transactions on Control Systems Technology (2017).

[6] Wei, Yue, Wang Liyuan, and Guo Ge. "Event-triggered platoon control of vehicles with time-varying delay and probabilistic faults." Mechanical Systems and Signal Processing 87 (2017): 96-117.

[7] Gong, Siyuan, Jinglai Shen, and Lili Du. "Constrained optimization and distributed computation based car following control of a connected and autonomous vehicle platoon." Transportation Research Part B: Methodological 94 (2016): 314-334.

[8] Morbidi, Fabio, Patrizio Colaneri, and Thomas Stanger. "Decentralized optimal control of a car platoon with guaranteed string stability." Control Conference (ECC), 2013 European. IEEE, 2013.

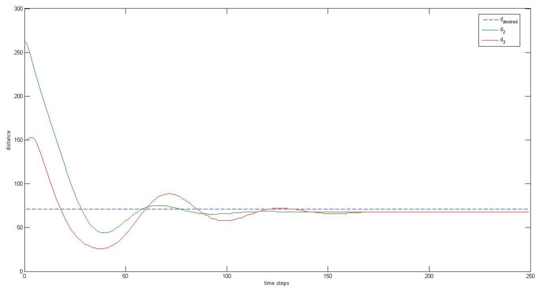


Fig. 17: The changing curves of d_{i-1} in the second test case using the FNN approach.

[9] Gao, Weinan, Zhong-Ping Jiang, and Kaan Ozbay. "Data-driven adaptive optimal control of connected vehicles." IEEE Transactions on Intelligent Transportation Systems 18.5 (2017): 1122-1133.

[10] Guo, Xianggui, et al. "Distributed adaptive sliding mode control strategy for vehicle-following systems with nonlinear acceleration uncertainties." IEEE Transactions on Vehicular Technology 66.2 (2017): 981-991.

[11] Yue, Wei, and Liyuan Wang. "Event-triggered autonomous platoon control against probabilistic sensor and actuator failures." Automatika (2017): 1-13.

[12] Zegers, Jeroen C., et al. "Consensus-based bi-directional CACC for vehicular platooning." American Control Conference (ACC), 2016. IEEE, 2016.

[13] di Bernardo, Mario, Alessandro Salvi, and Stefania Santini. "Distributed consensus strategy for platooning of vehicles in the presence of time-varying heterogeneous communication delays." IEEE Transactions on Intelligent Transportation Systems 16.1 (2015): 102-112.

[14] Montanaro, Umberto, et al. "Extended cooperative adaptive cruise control." Intelligent Vehicles Symposium Proceedings, 2014 IEEE. IEEE, 2014.

[15] Zheng, Yang, et al. "Influence of information flow topology on closed-loop stability of vehicle platoon with rigid formation." Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on. IEEE, 2014.

[16] Zhu, Wei, and Daizhan Cheng. "Leader-following consensus of second-order agents with multiple time-varying delays." Automatica 46.12 (2010): 1994-1999.

[17] Wang, Xiaofeng, and Michael D. Lemmon. "Event-triggering in distributed networked control systems." IEEE Transactions on Automatic Control 56.3 (2011): 586-601.

[18] Lunze, Jan, and Daniel Lehmann. "A state-feedback approach to event-based control." Automatica 46.1 (2010): 211-215.

[19] Seyboth, Georg S., Dimos V. Dimarogonas, and Karl H. Johansson. "Control of multi-agent systems via event-based communication." IFAC Proceedings Volumes 44.1 (2011): 10086-10091.

[20] Guo, Ge, and Shixi Wen. "Communication scheduling and control of a platoon of vehicles in VANETs." IEEE Transactions on Intelligent Transportation Systems 17.6 (2016): 1551-1563.

[21] Xu, Lijian, et al. "Communication information structures and contents for enhanced safety of highway vehicle platoons." IEEE Transactions on vehicular Technology 63.9 (2014): 4206-4220.

[22] Ghasemi, Ali, Reza Kazemi, and Shahram Azadi. "Stable decentralized control of a platoon of vehicles with heterogeneous information feedback." IEEE Transactions on Vehicular Technology 62.9 (2013): 4299-4308.