Chapter 4

Application of Learning Automata to Intelligent Vehicle Control

As stated in Section 2.2, vehicle control is one of the most important issues in ITS, and especially, in AHS. Designing a system that can safely control a vehicle's actions while contributing to the optimal solution of the congestion problem is difficult. Besides the control problems at the regulation layer, there is a need for extensive research on the planning layer of the control structure described by Varaiya [Varaiya 93], or at the tactical level of the driving task [Sukthankar96c]. When the design of a vehicle capable of carrying out tasks such as vehicle following at high speeds, automatic lane tracking, and lane changing is complete, we must also have a control/decision structure that can intelligently make decisions in order to operate the vehicle in a safe way.

The task of creating intelligent systems that we can rely on consequently brings the idea of "artificial intelligence" to mind. ITS community is well aware of the fact that the implementation of such a complex (and maybe global) system requires investigation of several different methods and the simultaneous applications of many. Several emerging methods, such as cellular automata, self-organization, neural networks, fuzzy logic, and hybrid systems applications are being mentioned [TRB95, Godbole95, Ho96, Deshpande96]. Initial research on intelligent vehicle control indicates that a planning-regulation system that can guarantee optimal operation with a sound theoretical background has not yet been developed, and it may be vital to the implementation of an automated highway system [Varaiya93, Lasky93].

In this chapter, we introduce a decision/control method for intelligent vehicles. Considering the complexity of an automated highway system or an intelligent vehicle-highway system, classical control methods are found to be insufficient to provide a fully automated, collision-free environment [Varaiya93]. Although we may not solve the "whole problem" using a single method, we attempt to find a way to make intelligent decisions here. Our approach to the problem of vehicle control makes the use of Learning Automata techniques described in Chapter 3. The learning algorithms used in this application are introduced and discussed separately in Chapter 6 for reasons of clarity. We visualize the planning layer (See Section 2.3) of an intelligent vehicle as an automaton (or automata group) in a nonstationary environment[1]. The aim here is to design an automata system that can learn the best possible action (or action pairs: one for lateral,

[1] The interpretation of the term 'environment' is twofold, as we explain later.

one for longitudinal) based on the data received from on-board sensors, and possibly some form of vehicle-to-vehicle and roadside-to-vehicle communications.

The significance of this system is that the controller we define will be useful as a backup system or the primary system in controlling the path of a vehicle in the case of communication loss with the higher layer in the hierarchy of a full AHS as well as during transition from fully automated to manual control. Since the implementation of a fully automated highway system is not imminent, we will try to concentrate on autonomous vehicles throughout this work.

4.1. The Model

For our model, we assume that an intelligent vehicle is capable of two sets of lateral and longitudinal actions. Lateral actions are *shift-to-left-lane* (SL), *shift-to-right-lane* (SR) and *stay-in-lane* (SiL). Longitudinal actions are *accelerate* (ACC), *decelerate* (DEC) and *keep-same-speed* (SM). The actions *stay-in-lane* and *keep-same-speed* are "idle actions," and can also be treated as a single action *cruise.* There are nine possible action pairs provided that speed deviations during lane changes are allowed (Table 4.1).

ACC SL	***ACC SR***	***ACC SiL***
DEC SL	***DEC SR***	***DEC SiL***
SM SL	***SM SR***	***SM SiL***

Table 4.1. Possible action pairs (shaded regions indicate the idle actions that can be combined into a single action).

An autonomous vehicle must be able to 'sense' the environment around itself. In the simplest case, it is to be equipped with at least one sensor looking at the direction of possible vehicle moves. Furthermore, an autonomous vehicle must also have the knowledge of the rate of its own displacement. Therefore, we assume that there are four different sensors on board the vehicle. These *are* the *headway sensor*, two *side sensors*, and a *speed sensor*. The headway sensor is a distance measuring device which returns the headway distance to the object in front of the vehicle. An implementation of such a device is a laser radar. Side sensors are assumed to be able to detect the presence of a vehicle traveling in the immediately adjacent lane. Their outputs are binary. Infrared or sonar detectors are currently used for this type of sensor. The speed sensor is simply an encoder returning the current wheel speed of the vehicle.

Each sensor is connected to its associated module. Sensor modules evaluate the sensor signals in the light of the current automata actions, and send a response to the automata (Figure 4.1). We visualize each sensor module as a teacher in a nonstationary automata environment (or a multi-environment system). The detailed descriptions of these sensor modules are given in the next section.

Our basic model for planning and coordination of lane changing and speed control is shown in Figure 4.1. The response of the environment is a combination of the outputs of all four

teacher blocks. The details of the mapping F is given in Section 4.4. The mapping F from sensor module outputs to the input b of the automata can be a binary function (for a P-model environment), a linear combination of four teacher outputs, or a more complex function — as is the case for this application. An alternative and possibly more ideal model would use a linear combination of teacher outputs with adjustable weight factors (*e.g.*, S-model environment).

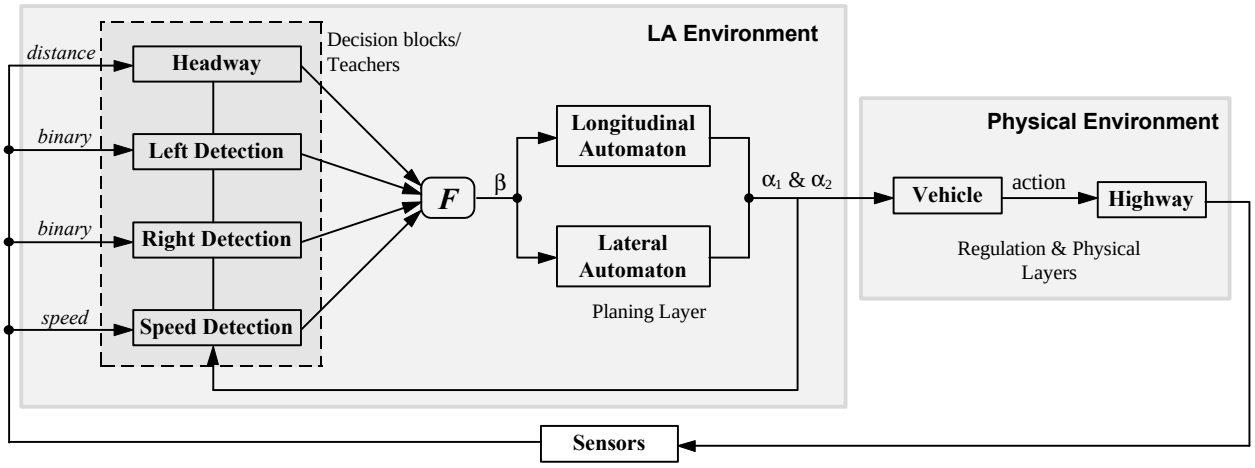


Figure 4.1 Automata in a multi-teacher environment connected to the physical layers.

It is important to differentiate between "automaton environment" and the "physical environment." The action a of an automaton is a signal to the regulation layer which defines the current choice. It is the regulation layer's responsibility to interpret this signal. When an action is carried out, it of course affects the physical environment. The sensors in turn sense the changes in the environment, and the feedback loop is closed with the sensor modules and the signal b.

The regulation layer is not expected to carry out the action chosen immediately. This is not even possible for lateral actions. To smooth the system output, the regulation layer carries out an action if it is recommended m times consecutively by the automaton, where m is a predefined parameter less than or equal to the number of iterations per second (Figure 4.2). In other words, the length of the *memory vector* is m. Whenever this vector (or buffer) is filled with the same action, the action is fired. This may of course be changed to "k times in the last m choices" or a more sophisticated decision rule.

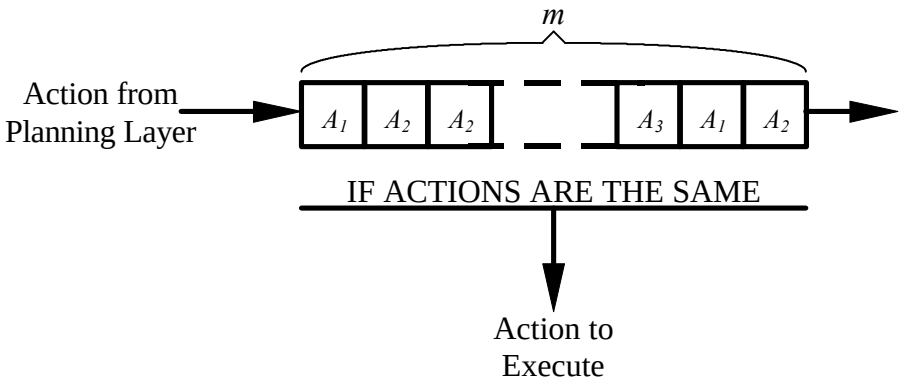


Figure 4.2. Memory vector/buffer in regulation layer.

When an action is carried out, the action probabilities in the controlling automaton are initialized to 1/*r*, where *r* is the number of actions. Although not necessary, this initialization enables the learning automaton to adapt to a new situation faster. One of the reinforcement schemes described in Chapter 6 is designed to speed up the adaptation process when the probability of the best/desired action is close to zero. When this new nonlinear scheme and/or very fast update rates are used, probability vector initialization is not necessary. When an action is executed, the memory vectors are initialized also; all locations are filled with the idle actions SiL or SM. A minimum processing speed of 25, and a maximum of 200 iterations per second are assumed. This value is related only to the computations; the sensor data feeds may have a different (and constant) rate. The upper limit of 200Hz is due to the communication requirements considered in the current AHS research [Lasky93].

The discussion of nonstationary environments in Chapter 3 is based on the changing penalty probabilities of actions. In this application, the action probabilities of the learning automata are functions of the status of the physical environment (*e.g.*, a decreasing headway distance will result in a penalty response from the front sensor module if the chosen action is for ACC). The realization of a deterministic mathematical model of this physical environment may be impossible, if not extremely difficult. Therefore, simulation is the only way of demonstrating the expediency and/or absolute optimality for a nonstationary automata environment resulting from a changing physical environment.

4.2. Sensor/Teacher Modules

The four teacher modules listed above are simple decision blocks that calculate the response associated with the corresponding sensor, based on the last chosen action. Table 4.2 below describes the output of these decision blocks for side sensors.

	Sensor Status (Left /Right)	
Actions[2]	Vehicle in sensor range or no adjacent lane	No vehicle in sensor range and adjacent lane exists
SiL	0 / 0	0 / 0
SL	1 / 0	0 / 0
SR	0 / 1	0 / 0
SM	#	#
ACC	#	#
DEC	#	#

Table 4.2. Output of the left/right sensor modules.

As seen in Table 4.2, a penalty response from the left sensor module is received only when the action is SL and there is a vehicle in the left sensor's range or the vehicle is already traveling at the left or rightmost lane. Similarly, the action SR is penalized if there is a vehicle on the right lane. All other situations result in a reward response from this sensor module. The longitudinal automaton do not use side sensors for reinforcement. Initial simulations define the range of the side sensors using two parameters sr_1 and sr_2 as shown in Figure 4.3. It is assumed that the side sensor's range covers only the adjacent lane, and the sensor is mounted right at the middle of the vehicle's side.

The output of the side sensors are assumed to be binary, indicating the existence of a vehicle on the adjacent lane. However, the side sensor module may also use the distance of the detected vehicle from the sensor source for more intelligent decisions. This type of design may result in a more expensive implementation, as it requires distance measurement and possibly additional sensors.

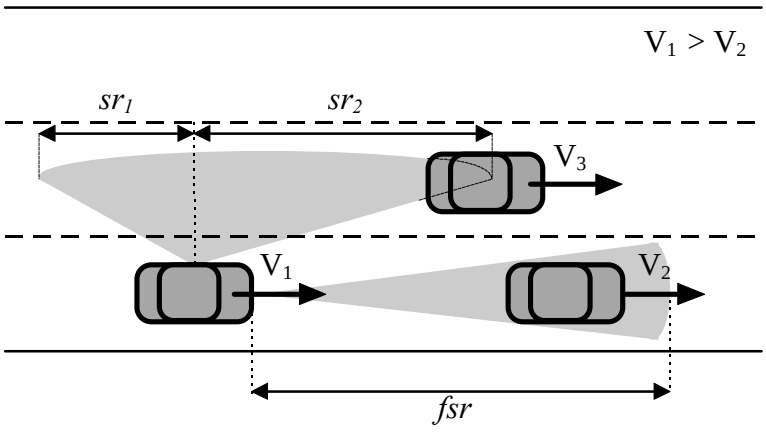


Figure 4.3. Sensor ranges for an autonomous vehicle.

[2] The actions are *shift-to-left-lane* (SL), *shift-to-right-lane* (SR), *stay-in-lane* (SiL), *accelerate* (ACC), *decelerate* (DEC) and *keep-same-speed* (SM).

Assuming that the front sensor is capable of detecting the headway distance, we define the headway module as shown in Table 4.3. If the sensor 'sees' a vehicle at a close distance (< *fsr*), a penalty response is sent to the automaton for actions *stay-in-lane*, *accelerate*, and *keep-same-speed*. All other actions (shifting lane or decelerating) may serve to avoid a collision, and therefore, are encouraged. Then, an implemented headway module's task is simply to compare the time-of-flight of the echo to the predefined time interval corresponding to distance *fsr*. Also note that although we show the limiting value *fsr* as the sensor range in Figure 4.3, the actual sensor range and this value do not have to be equal. The output of this module affects both automata as seen in Table 4.3.

The speed sensor of the autonomous vehicle is assumed to be an encoder connected to the wheel shaft. The output of the encoder can be used to detect the wheel speed, which is approximately equal to vehicle speed (or equal when cruising). The speed module's task is simply to compare the actual speed to the desired speed, and based on the action chosen, send a feedback to the longitudinal automata. The responses depending on the speed and the longitudinal actions are given in Table 4.4.

	Sensor Status	
Actions	Vehicle in range (*headway* < *fsr*)	No vehicle in range (*headway* ≥ *fsr*)
SiL	1	0
SL	0	0
SR	0	0
SM	1	0
ACC	1	0
DEC	0*	0

Table 4.3. Output of the *Front Sensor* block.

	Sensor Status		
Actions	*dev* < *-pdif*	*-pdif* < *dev* < *pdif*	*dev* > *pdif*
SiL	#	#	#
SL	#	#	#
SR	#	#	#
SM	1	0	1
ACC	0	0	1
DEC	1	0	0
dev" actual speed - desired speed *pdif" permitted difference between actual and desired speeds*			

Table 4.4. Output of the *Speed Sensor* Module.

When the actual vehicle speed differs from the desired speed by a large amount, then only the action decreasing the speed deviation receives a reward; others are penalized. This forces the vehicle to slow down or speed up in order to match the desired speed. The value of the parameter *pdif* (permitted difference between actual and desired speeds) is predefined.

The values of the sensor limits *sr1*, *sr2*, *fsr*, and *pdif* define the capabilities of the sensors (in the case front and side sensors) as well as the "behavior" of the vehicle. Sensitivity to the headway distance and/or to speed fluctuations are given by the predefined parameters and are key to the behavior of the autonomous vehicle. For example, a vehicle with shorter side sensor range will tend to "jump in front of vehicle in the next lane" more often than a vehicle equipped with more capable sensors.

Now that we have defined the sensor module outputs, the problem is to intelligently employ these signals for automata reinforcement. It is possible to treat all sensor modules as separate teachers with conflicting feedback responses for the automata. For example, consider the situation given in Figure 4.3. The longitudinal action will receive a penalty from the front sensor module due to the existence of a vehicle in front. If the actual speed of this vehicle is less than the desired speed, the speed module will try to force the vehicle to increase its speed. These two outputs conflict, and it is obvious that one must have priority over the other: no matter what its current speed is, a vehicle must slow down in order to avoid a collision if it senses another vehicle occupying the immediate space in front of itself. The next section describes our approach to the problem.

4.3. Nonlinear Combination of Multiple Teacher Signals

Action probabilities of the longitudinal and lateral automata are updated using the binary feedback from the four sensor modules described in the previous section. As seen in Table 4.5, lateral actions probabilities depend on three sensor outputs (front and side sensors), and longitudinal action probabilities are updated based on the output of the headway and speed sensor modules.

	Modules			
Actions	Headway	Left	Right	Speed
SiL	?	0	0	-
SL	0	?	0	-
SR	0	0	?	-
SM	?	-	-	?
ACC	?	-	-	?
DEC	0 or 0*	-	-	?

'?' means reward (0) or penalty (1) response is possible.
'-' means that output is not used; can be treated as '0'.
'0*' has a higher priority (See discussion below).

Table 4.5. Action - Sensor Module matrix.

The probability of the lateral action SiL depends mainly on the output of the headway sensor. When this action is chosen, the probability is decreased when the headway module indicates a vehicle in the front sensor range. Side sensors always sends a reward response to this action no matter what the situation is. Therefore, we can obtain a combined environment response for this action by simply OR-ing the binary module outputs.

The same method is also valid for the two other lateral actions. Probabilities of actions SL and SR depend on the output of the left and right sensor modules, respectively. A decision to shift lane is penalized if the associated sensor indicates the presence of a vehicle in the adjacent lane. Therefore, by using a simple OR function, we can combine all three sensor module outputs to obtain a meaningful teacher response for the lateral automaton.

For the longitudinal actions, although there are only two sensors modules whose feedback is considered for reinforcement, the decision is more complicated. Longitudinal actions SM and ACC may receive a penalty or reward from the front sensor depending on the headway distance. The output of the speed module depends on the actual speed of the vehicle, and is used to force the vehicle to match the desired speed. Action DEC is considered as 'good' all the time by the headway sensor module. However, the reward response in the case where there is a vehicle in the front sensor range (indicated by 0*) is different than the normal reward response (indicated by 0): this reward response must "override" a possible penalty from other modules. For the safety of the vehicle, the output of this sensor must have a higher priority than that of the speed module. Therefore, a simple OR-gate is not sufficient; additional Boolean functions must be used. Possible action-sensor output scenarios are shown in Table 4.6.

	Sensor Module Output		
Actions	**Speed**	**Headway**	**Combined**
SM or ACC	0	0	0
	0	1	1
	1	0	1
	1	1	1
DEC	0	0	0
	0	0*	0
	1	0	1
	1	**0***	**0**

Table 4.6. Possible longitudinal action-sensor output scenarios.

Again, an OR-ing of the headway and speed module outputs will give the combined output in Table 4.6 except the case where the action is DEC and the headway module indicates the presence of a vehicle in front. Therefore, the output of an OR-gate must be "double-checked" for this condition. Since we are using OR gates, we can define "don't care" conditions ('-') as reward responses. Thus, it is possible to OR the outputs of all sensors. Then, the function *F* mapping individual sensor outputs to a combined feedback signal is defined as in Figure 4.4.

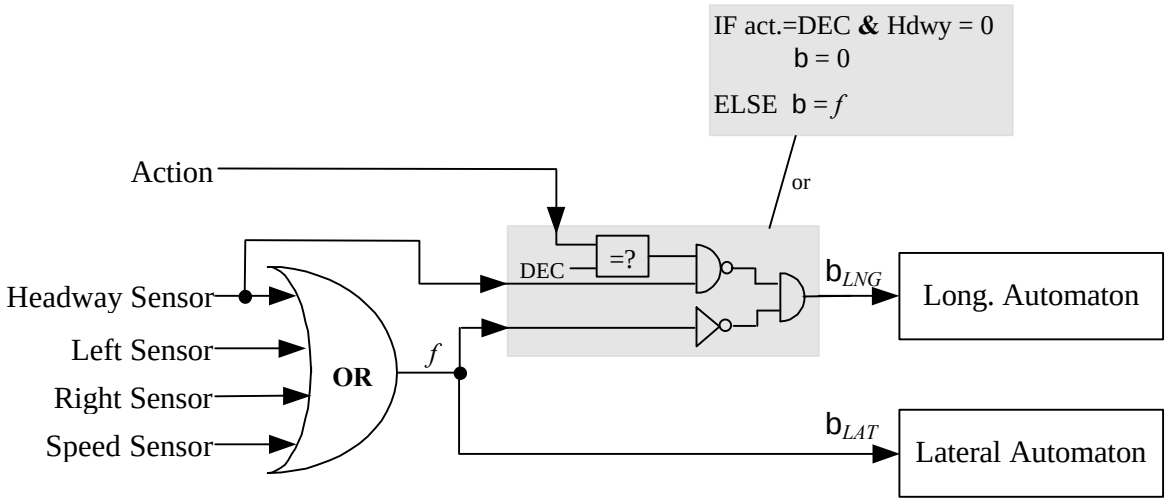


Figure 4.4. The definition of the mapping *F*.

Therefore, by adding one conditional rule to the OR-ed feedback signal, we obtain a combined environment response to be applied to the automata. This single response is used to update the action probabilities in both automata using the reinforcement schemes described in the next section.

4.4. The algorithms

As described in previous chapter, the environment response is used to update the action probabilities in an automaton. The update algorithm, called the reinforcement scheme, is the key to automata learning. In Chapter 3, we introduced the learning paradigm and related definitions. As we describe in more detail in Chapter 6, the reinforcement schemes are categorized according to their nature and the behavior of the automaton. There are four different reinforcement schemes used throughout this dissertation. These are:

- Linear reward-penalty L_{R-P} (with $a = b$),
- Multi-teacher general absolutely expedient (MGAE) scheme,
- Linear reward-penalty $L^{\neq}_{R-P}$ (with $a \neq b$),
- NL_H, an absolutely expedient nonlinear scheme.

The first algorithm is a well-known linear reinforcement scheme, one of the first LA algorithms [Bush58]. The second algorithm is given by Baba [Baba83, Baba85], and also is valid for S-model environments. The last two reinforcement schemes are extensions of the first two, and have desirable characteristics for our application. They are the direct results of our attempts to create reinforcement schemes with desirable characteristics suitable for this study of learning automata applications to intelligent vehicle control. The detailed description of the algorithms can

be found in Chapter 6. Here, we will briefly mention their characteristics and relative properties. Table 4.7 gives the general description of the above mentioned schemes.

Scheme	Nature	Behavior	Advantages
L_{R-P}	- Linear - Equal learning parameters ($a = b$) - Nonprojectional	- Expedient [Narendra89]	
$L^{\neq}_{R-P}$	- Linear - Unequal learning parameters ($a \neq b$) - Nonprojectional	- Expedient, and optimal in an "ideal environment" (See Chapter 6)	-Faster convergence with same learning parameter *a.* - Optimal (equilibrium) solution is guaranteed in N-person nonzero-sum games
MGAE	- Nonlinear - Projectional - Applicable to *S*-model	- Absolutely expedient [Baba85]	
NL_H	- Nonlinear - Projectional - Applicable to *S*-model	- Absolute expedient (See Chapter 6)	- Faster convergence than MGAE

Table 4.7. Properties of the reinforcement schemes.

The second algorithm is slightly different than the first one, and has not been studied in detail previously. However, the convergence characteristics is better when the same learning parameter a associated with reward is used. Furthermore, $L^{\neq}_{R-P}$ is proven to guarantee convergence to an equilibrium point[3] in N-person games. This is a property we would like to use for analyzing the behavior of multiple autonomous vehicles. The last algorithm is an extension of the third one, again resulting in a faster convergence in probability updates. These two schemes are also proven to be absolutely expedient, meaning that the decision tends to a 'better' one at every time step.

4.5. Simulation Results

The first simulation example shows a single autonomous vehicle traveling faster than (nine) other vehicles on a 3-lane highway. All other vehicles are assumed to be cruising at fixed speed of 80kmh without changing lanes. The autonomous vehicle's desired speed is set at 85kmh. Sensors limits are fsr = 15m and $sr_1 = sr_2$ = 10m; the permitted speed deviation is 1kmh. Linear reward-penalty scheme $L^{\neq}_{R=P}$ is used for reinforcement (see Table 4.8 for all other parameter settings). At

[3] Or *Nash equilibrium*, where no player has a positive reason to change her strategy.

t = 0, the autonomous vehicle, traveling in lane 3, approaches the group of vehicles cruising at fixed speed of 80kmh (Figure 4.5a; the vehicles are traveling from left to right). It immediately changes lane to avoid a collision. It detects another slow moving vehicle in lane 2 at approximately t = 10sec and slows down to keep a 15-meter headway distance (see Figure 4.6a). At approximately t = 14sec, it shifts to lane 1. Immediately after the lane change, the speed is adjusted to desired value of 85kmh. At t = 34sec and t = 45sec, two shift-to-left-lane actions are carried out, again due to the presence of slow moving vehicles in front (Figures 4.5d-e). During these maneuvers, the vehicle speed is not decreased, because the probability of the lateral action SL reached 1 much faster than that of the longitudinal action DEC. The reason for this behavior is the fact that the lateral action SR receives penalty at all times due to the presence of another vehicle in the right lane, or due to the fact that the vehicle is in the rightmost lane.

After two shifts to the right lane (Figures 4.5f-h), the vehicle finds itself again in lane 1, traveling behind another vehicle. The speed is adjusted to match the speed of the vehicle in front (Figure 4.6). At this point, the only lateral action receiving reward is SiL, and this action is fired repeatedly. Longitudinal automata, on the other hand, will chose the action necessary to keep the headway distance over 15m. Whenever the headway distance is larger than this predefined limit, the vehicle will attempt to increase its speed to 85kmh. It cannot move out from the pocket created by two slow moving vehicles.

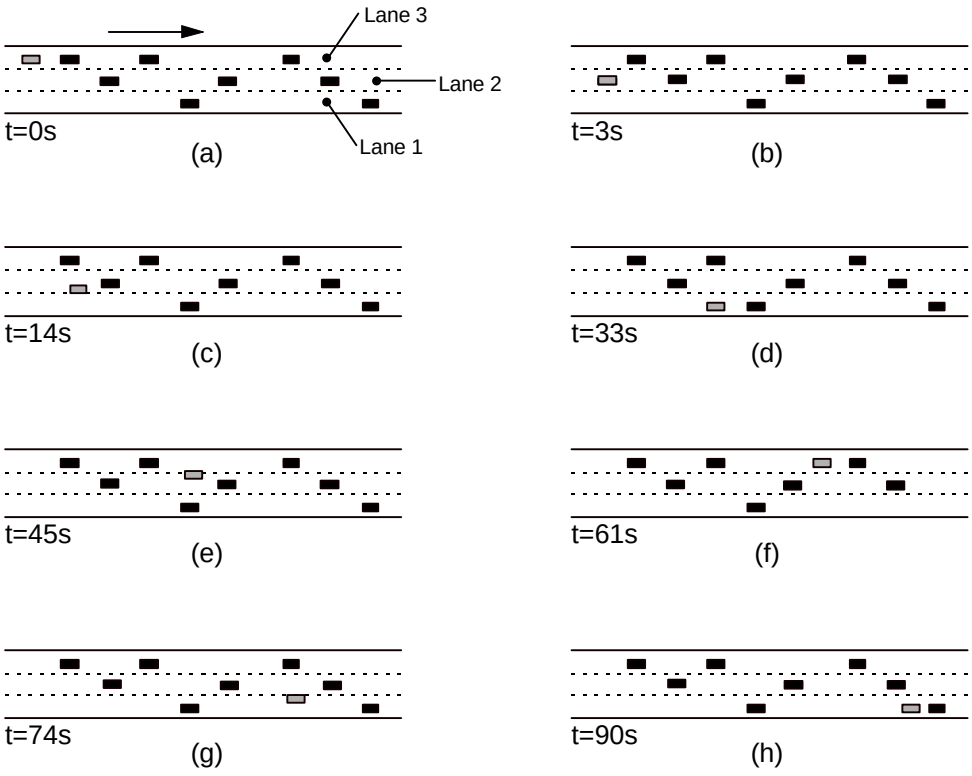


Figure 4.5. Positions of nine vehicles: gray colored vehicle is autonomous, black colored vehicles are cruising at constant speed.

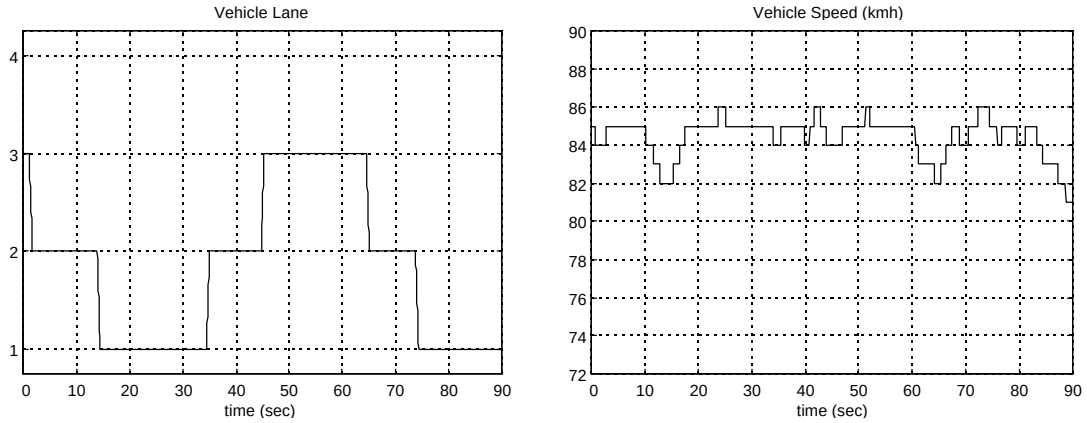


Figure 4.6. Automated vehicle's (a) lane position and (b) speed (See also Figure 4.5).

The second and third simulation results show the behavior of a vehicle following a slowly moving vehicle. Both simulations have been started with an initial headway distance of 30m (Figure 4.7). Front sensor range *fsr* is set to 15m. The memory vector size is the same as the processing speed (25), and this results in speed changes every second at the least. The only difference between two simulations is in the desired speeds; 81kmh in the second, and 85kmh in the third simulation. The headway distance and speed of the automated vehicles are shown in Figures 4.8 and 4.9.

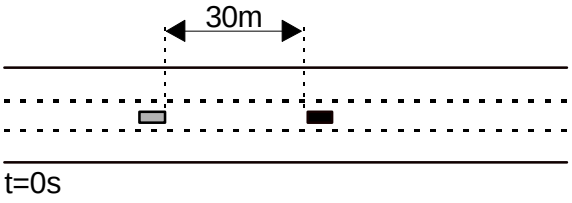
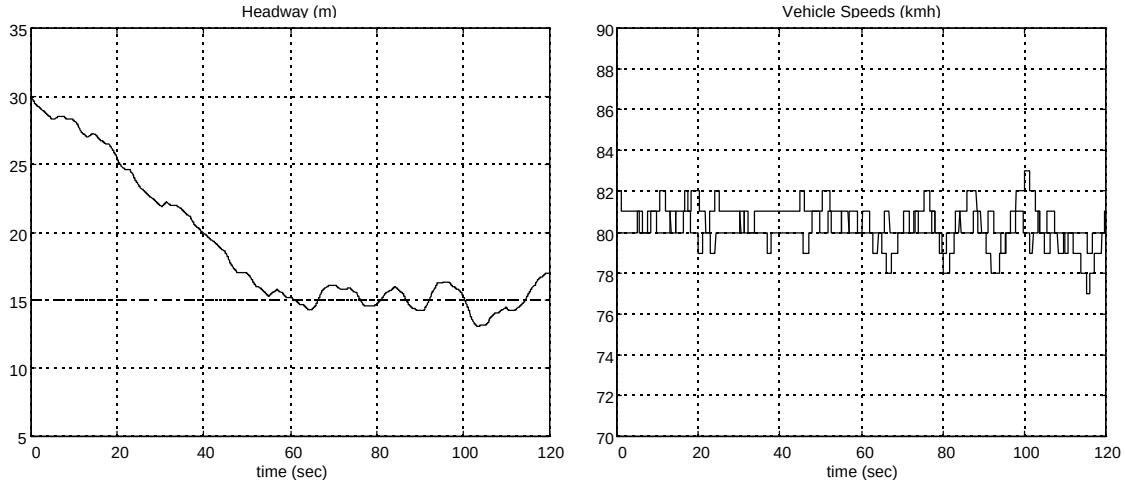
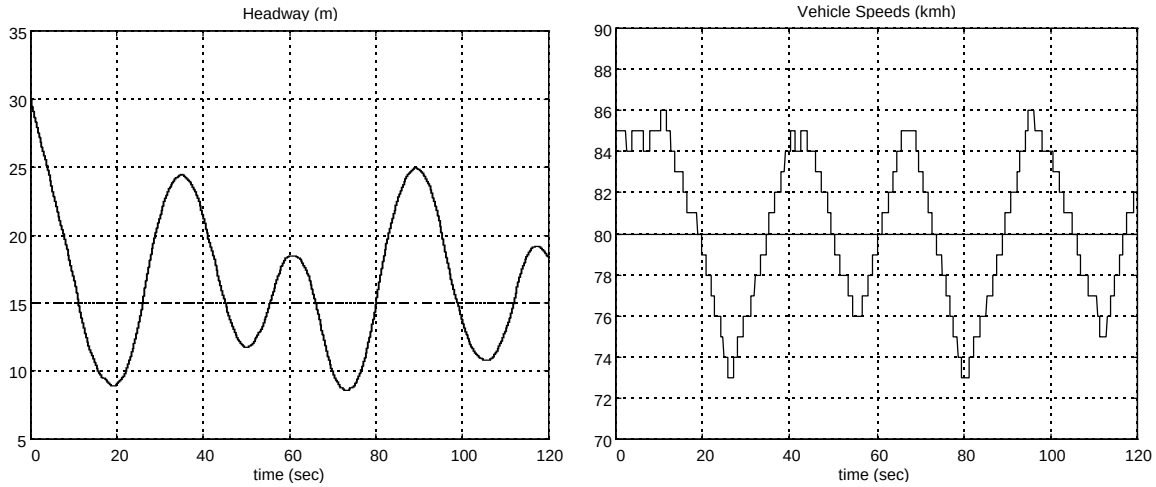


Figure 4.7. An automated vehicle following a slow-moving vehicle in lane 2.



(a) (b)

Figure 4.8. (a) Headway distance and (b) speed of an autonomous vehicle following another slowly moving vehicle: the desired speed of the autonomous vehicle is 81kmh while the vehicle in front cruises at 80kmh.



(a) (b)

Figure 4.9. (a) Headway distance and (b) speed of an autonomous vehicle following another slowly moving vehicle: the desired speed of the autonomous vehicle is 85kmh while the vehicle in front cruises at 80kmh.

As seen in the above figures, both vehicles are able to avoid a collision[4]; the headway distance is never too close. However, the response of the automated vehicle is oscillatory. The longitudinal automaton sends the action ACC whenever the headway distance is larger than 15m

[4] This is, of course, possible provided that the iteration rate is fast enough or memory vector is relatively short.

to reach the desired speed of 85kmh. The action DEC is the choice when the distance is less than the predefined limit. Also, the oscillation amplitude is higher when the relative speed is higher. Although the oscillations are smaller for the desired speed setting closer to the speed of the vehicle in front, the behavior is unstable, as seen in Figure 4.7.

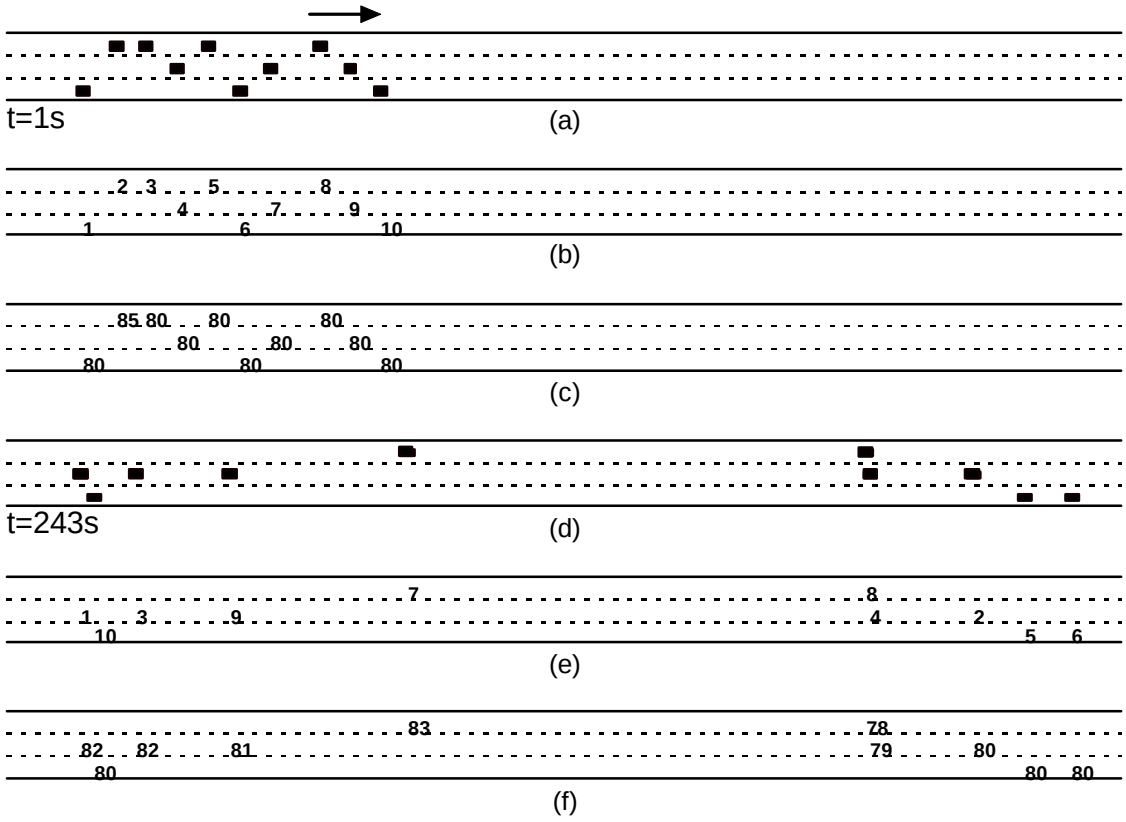
Sim. #	Learning Parameters		Sensor limits (m)			Proc. Speed	Memory buffer size		Figures
	a	b	fsr	sr2	sr1	(Hz)	lateral	long.	
1	0.15	0.10	15	10	10	25	25	25	4.5, 4.6
2	0.15	0.10	15	10	10	25	25	25	4.7, 4.8
3	0.15	0.10	15	10	10	25	25	25	4.7, 4.9
4	0.15	0.10	15	10	10	25	25	25	4.10

Table 4.8. Parameter settings for simulations.

Although not shown here, the behavior of the vehicles change with different parameters. The choice of sensor limits and learning coefficients as well as the iteration speed and the size of the regulation buffer are important factors.

Another important issue is the fact that the automated vehicles in simulations 2 and 3 chose to adjust their speed (back and forth) to keep a safe distance instead of shifting to lane 1 or lane 3 to evade the slow moving vehicle. Since the only lateral action receiving a penalty from the front sensor module is SiL, the probability of the two other actions SL and SR reach 0.5. This results in a memory buffer filled with (approximately) equal number of two different actions. The vehicle cannot decide which action to take.

Figure 4.10 shows the initial and final positions, numbers and speeds of ten autonomous vehicles. The vehicles are traveling from left to right on a 500m circular highway. Final speeds shown in Figure 4.10(f) are equal to desired speeds set initially. Simulation parameters are again given in Table 4.8.



ch4m1.mpg
(1.230Mb, every frame represents one fifth of a second)
(g)

Figure 4.10. Initial and final positions (a, d), numbers (b, e), and speeds (c, f) of ten automated vehicles traveling on a 3-lane 500-meter circular highway. The *mpeg* movie of the simulation is accessible via icon (g).

As seen in figure above and the *mpeg* movie, vehicles with higher speeds are slowly moving ahead of the others. All vehicles are able to avoid collision, either by slowing down or shifting lanes. In the end, all vehicles have reached their desired speeds, since they were all able to shift to an open lane, sooner or later during the simulation. However, the paths taken to overtake slower vehicles are not the shortest and/or quickest paths.

4.6 Discussion of the Results

The simulation results given in the previous section are indicative of several important issues. First of all, assuming that the regulation layer can respond to the planning layer's requests in time, and that the real dynamic behavior of this system as well as actual sensor are not drastically different than the simplified model here, our planning layer intelligent controller is capable of safely directing the vehicle(s). Of course, the processing speed and the learning parameters need to be adjusted to guarantee timely responses to environmental changes. Considering the current

technology in computing, the iteration speed is not a big concern. Learning parameters must be chosen large enough to guarantee fast learning for the desired iteration speed while avoiding values larger than 0.3 for linear schemes, and 0.1 for nonlinear schemes in order to avoid unnecessarily large action probability update of a non-optimal action probability to 1. The length of the memory vector is another factor in "firing" actions. This buffer is definitely needed for lateral actions, in order to avoid continuous lane changes, while its size can be decreased down to 1 for longitudinal actions provided that the speed changes are continuous.

Although there are no collisions with carefully selected parameters to obtain the necessary convergence speed, the path and/or behavior of the vehicles are far from optimal. The autonomous vehicle in simulation number 1 did not possess the information to help it avoid a pocket created by two slow moving vehicles. The decisions are based on local data, and therefore cannot guarantee global solutions for the vehicle, or the traffic as a whole for that matter. Conflicting decisions are expected.

The observed oscillations in the headway distance is due to the "discrete time" control of the speed, and the defined front sensor limit. For a fast processing speed and a more realistic speed controller design, the problem may be relatively insignificant. However, with the current definition of the headway module, the vehicles will keep decelerating even though the headway distance is "improving."

Furthermore, unnecessary lane changes can be observed in simulation number 4. This is again due to the fact that the vehicles do not have an evaluation method for the desired lane when such a decision is needed. This fact is also the cause of the behavior seen in simulations 2 and 3. When lateral actions SL and SR are both "good," the vehicle is unable to change lanes using the current method of firing actions when the memory vector/buffer is full (see Section 4.1).

Initial simulations show promising results, but there is a need for more elaborate sensor definitions, and for more complex decision rules. The next chapter will address these upgrades, and other related issues.

It is interesting to note that again a task performed easily by human beings proves to be very complicated for a "machine." Even a simple decision of shifting lanes in order to avoid being stuck in a packet of slow moving vehicles (simulation 1, Figure 4.5h) requires complicated sensory hardware and, possibly, a complex communication mechanism. However, a human driver can simply look farther ahead toward the adjacent lanes, and decide to slow down in order to find an opening. Research currently conducted on AHS aims to solve such problems with extensive vehicle-to-vehicle and/or vehicle-to roadside communications [Construction96, Lygeros94], multiple sensors and sensor fusion [Agogino96] and a hierarchical control structure [Varaiya91, Varaiya93, Lygeros95, Lygeros96]. When, the hierarchical system and all vehicles are "connected," they will collectively reach a solution. However, such an architecture consequently requires an expensive implementation. The problem of making decisions based on local information would exist even in a full AHS environment unless the system is fully hierarchical.