Chapter 5

More Intelligent Vehicles: Extending the Capabilities

The intelligent controller defined in the previous chapter, although capable of controlling an automated vehicle's path using "local" information, is far from optimal in the sense that the resulting "safe" path is not always the shortest or fastest path, nor a good choice for improving the overall throughput of a segment of the automated highway. Also not seen in the simulation examples, a "pinch maneuver" condition where two vehicles changes lanes to occupy the same spot on the highway (See Section 5.2.2) is possible when only the immediate neighborhoods of the vehicles are considered. In this chapter, we will extend our assumptions on the capabilities of the intelligent vehicles. These changes can be grouped into three different classes: extending the capabilities of the sensors, creating more complete (and complex) decision rules, *i.e.*, extending the capabilities of the "teachers," and adding more sensor/teacher modules. All these changes are described in the following sections.

Extensions on sensor capabilities require more complex and expensive hardware implementations, and extending the capabilities of the teachers leads to more complex evaluation rules for teacher modules as we describe in the following sections. These include new definitions of front and side sensors, more rules for headway and side sensor modules, as well as interaction of lateral and longitudinal automata. Two new sensor modules are defined in this chapter, namely the pinch and lane sensor-module pairs. The assumptions and physical implementations of these sensors as well as the 'flag' structure used in the decision modules are described.

5.1. More Complex Sensors and Sensor Modules

The front sensor of an automated vehicle is defined as a range sensor in Chapter 4. Without changing our assumptions on the sensor hardware, we will extend the decision capability of the front sensor module by defining multiple headway ranges and by using consecutive range data to obtain rate of change of the headway distance as well as its current value. For the side sensors, we assumed only that these were capable of detecting a vehicle in a predefined range. Extensions on the side sensor modules in this chapter require more capable and therefore complex sensor structures.

5.1.1 Front Sensor

Again assuming that the sensor mounted in the front of the vehicle can detect the headway distance, reward-penalty rules in the headway module are redefined as shown in Table 5.1. The

cases where the output of the sensor has an overriding characteristic are indicated by asterisks. There are now three different range values used in the feedback decision. The value *fsr* defines the sensor range, or this value can be visualized as the limit of the region of interest. Any measurements equal or greater than this value results in a reward response for all longitudinal and lateral actions. As seen in Figure 5.1, two additional parameters d_1 and d_2 with $d_1 < d_2 < fsr$ are used to define a new "buffer region" where the sensor modules response also depends on the rate of change of the headway distance.

	Sensor Status					
	Vehicle in region A	Vehicle in region B (d_1 < dist. < d_2)		Vehicle in region C (d_2 < dist. < *fsr*)		No vehicle in range
Current Action	(dist. < d_1)	Vehicle is approaching	Vehicle is NOT approaching	Vehicle is approaching	Vehicle is NOT approaching	(dist. ≥ *fsr*)
SiL	1	1*	0	1*	0	0
SL	0**	0	0	0	0	0
SR	0**	0	0	0	0	0
SM	1	1	0	0	0	0
ACC	1	1	1	1	0	0
DEC	0	0*	0	0	0	0
Asterisks are used to distinguish the cases where additional functions need to be used (See Section 5.3.3 for details).						

Table 5.1. Output of the *Front Sensor* block (Also see Figure 5.1).

The lateral action SiL is "discouraged" when the vehicle in front is dangerously close. The parameter d_1 defines this safety region. The same action also receives penalty when the headway distance is larger than d_1, but less than the safe distance defined by parameter *fsr*. However, for the region between these two values, the action is penalized only when the rate of change of the headway distance is negative, *i.e.*, if the headway distance decreases. All other lateral actions receive reward from the front sensor module.

Longitudinal action ACC is discouraged when a vehicle enters the front sensor range, and is approaching. If the headway distance is under d_2, this action is penalized regardless of the rate of change of the measured distance. A vehicle should not increase speed while sensing another vehicle in proximity. The differentiation of feedback response for the range [*d2 fsr*] enables the automated vehicle to 'catch up' with the vehicle in front. If the headway distance is too close, or if the vehicle in front is approaching, the action SM also receives penalty.

Note that if we choose the parameters as $d_1 = d_2 = fsr$, the reward-penalty structure of Table 5.1 becomes the one previously given in Table 4.4. By defining these new regions where the feedback response is different for different longitudinal actions, we hope to obtain a smoother response, and thus remove the oscillations in the headway distance while following a slowly moving vehicle (See examples in Figure 4.8 and 4.9). The reward response of the front sensor module for longitudinal action ACC in region C facilitates speed recovery.

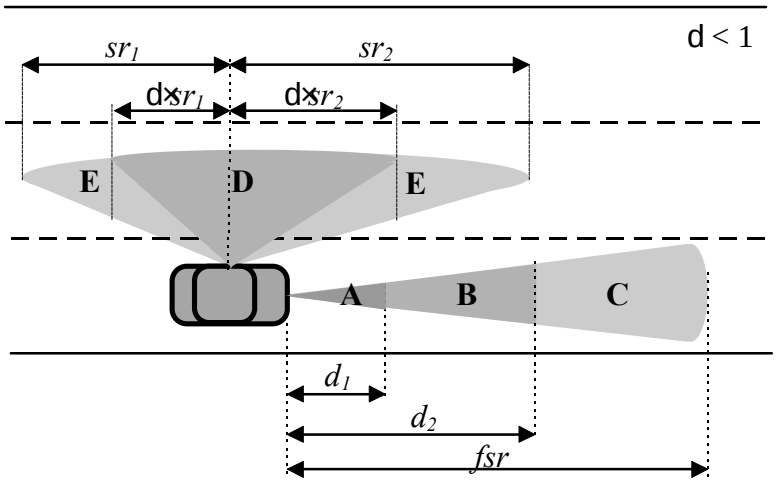


Figure 5.1. Extended definitions of front and side sensor ranges.

Again, consider the situation given in Figure 4.7 where the automated vehicle follows a slow-moving vehicle. The initial headway distance is still 30m, the sensor parameters are chosen as shown in Table 5.2. The automated vehicle has a desired (and initial) speed of 85kmh while the other cruises at 80kmh. All other parameters are set to the same values as in the second and third examples in Chapter 4 (Table 4.8). The regulation layer changes velocity after receiving the same action 25 times consecutively, and the processing speed is 25 iterations per second.

Simulation Number	**Figure**	**Parameters (m)**		
		d1	***d2***	***fsr***
1	5.2	10	20	30
2	5.3	10	15	20
3	5.4	12	15	18
4	5.5	14	15	16
5	5.6	12	15	16

Table 5.2. Front sensor parameters for simulations.

The vehicle speeds and the headway distances for the five simulations with the above parameters are given in Figures 5.2-5.6. Only the sensor range definitions are changed to see the effect of various settings. As seen in Figures 5.2 and 5.3, it is possible to prevent the oscillations in the headway distance. Usually, the headway distance stabilizes at a value between d_1 and d_2 when the automated vehicle approaches a slow-moving vehicle. As seen in Figure 5.2, the regulation layer changed the speed 5 times to match the speed of the vehicle in front, in

approximately 6 to 7 seconds. Again, note that a maximum of one change per second was possible in these simulations.

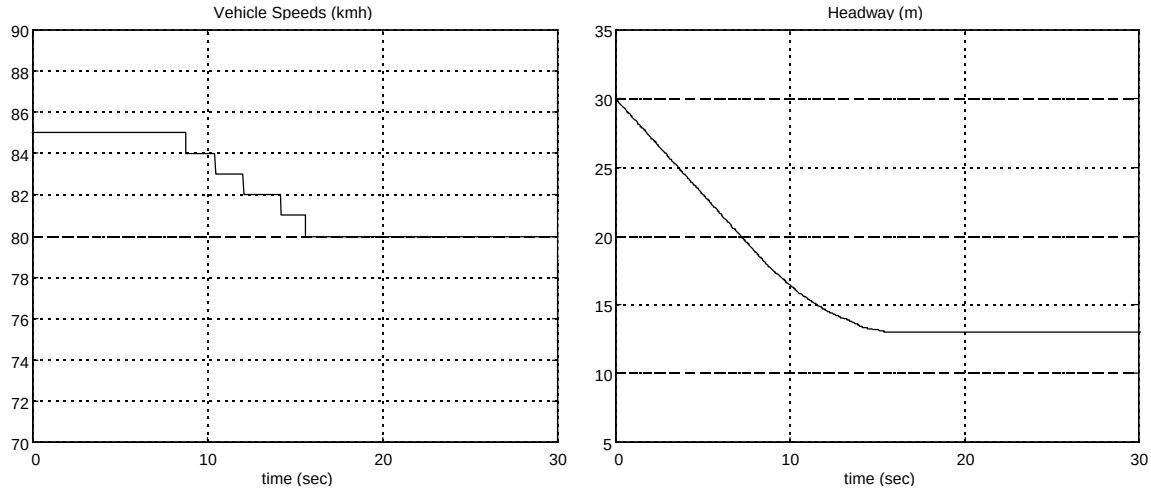


Figure 5.2. Headway distance and speed of an autonomous vehicle following another slow-moving vehicle: sensor parameters are d_1 = 10m, d_2 = 20m, and *fsr* = 30m.

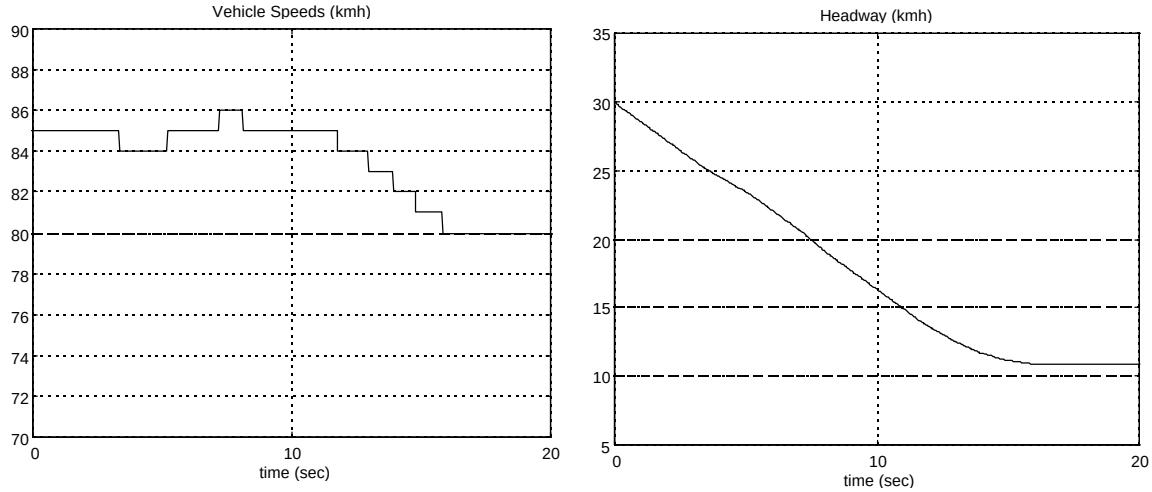


Figure 5.3. Headway distance and speed of an autonomous vehicle following another slow-moving vehicle: sensor parameters are d_1 = 10m, d_2 = 15m, and *fsr* = 20m.

Figure 5.4 shows another vehicle with much shorter sensor ranges than the previous ones. Again starting at the same initial conditions, the vehicle is able to match the speed of the vehicle in front, and keep a safe distance. Since the parameter *fsr* is relatively shorter than previous simulation runs, the vehicle had less time to adjust its speed — starting at only 18m. As a result of very rapid changes (only action DEC is permitted after t = 10sec since the headway is less than d_2 and the vehicle is approaching), the headway decreased under d_1 = 10m and the vehicle speed

below 80kmh. For approximately 6 seconds, the automated vehicle cruised at 79kmh in order to permit the headway to increase to a safe distance. At t ≈ 30sec, right after the headway distance becomes equal to d_2, the speed is matched. In this case, the region [d_2 fsr] is where the steady-state value is obtained. After t = 40sec, the vehicle ahead changes its speed to 83kmh, then back to 80kmh. As seen in Figure 5.4, this speed change is quickly matched, and the headway distance reaches its steady-state a few seconds after the vehicle in front reaches its cruising speed of 80kmh (at $t \approx$ 60sec).

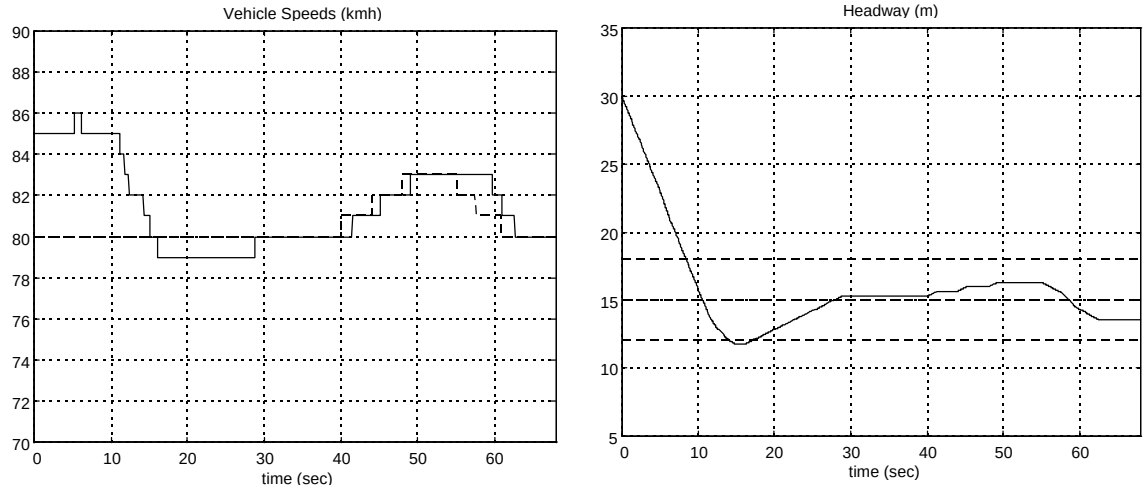


Figure 5.4. Headway distance and speed of an autonomous vehicle following another slow-moving vehicle: sensor parameters are d_1 = 12m, d_2 = 15m, and fsr = 18m.

When we attempt to decrease the sensor regions further to [d_1 d_2] = [14 15] and [d_2 fsr] = [15 16], oscillations resulted as seen in Figure 5.5. The regions where different actions are encouraged based on the distance and rate of change of distance are so small that before an intelligent decision can be made, the value of the headway distance moves out of the 'buffer region.' Again, it is important to note that an ACC or DEC action is fired every second in the best case. For a much shorter memory buffer as well as a faster iteration speed, the response will be quicker, and therefore these sensor values may be feasible for non-oscillatory response.

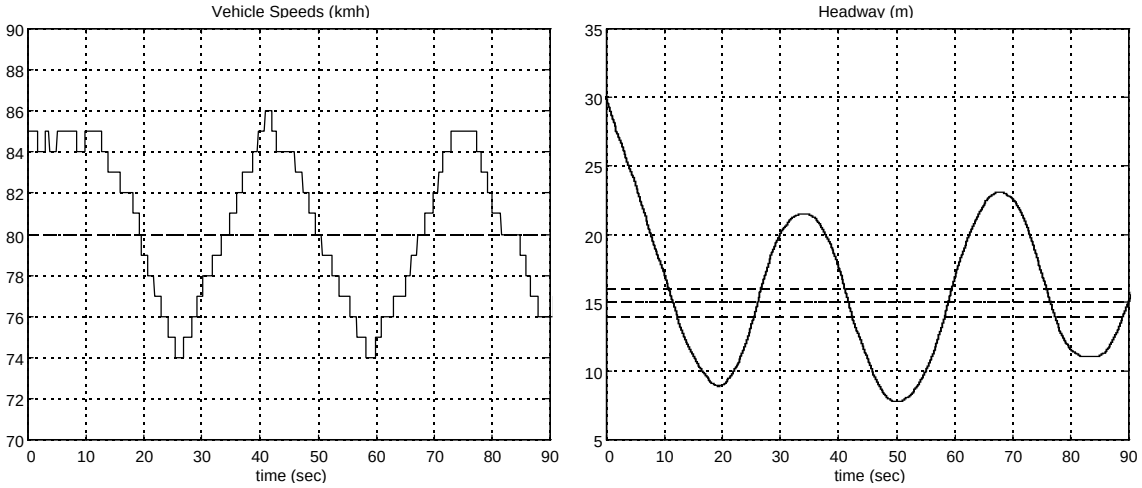


Figure 5.5. Headway distance and speed of an autonomous vehicle following another slow-moving vehicle: sensor parameters are d_1 = 14m, d_2 = 15m, and fsr = 16m.

Keeping the same processing speed and memory vector, we try to solve the problem by enlarging the lower sensor region, *i.e.*, by decreasing the value of d_1 to 12m. The results are shown in Figure 5.6. The response is not as 'damped' as in simulations 1 and 2, but a steady-state value is reached. Again, a change in the speed of the vehicle in the front (at $t \approx 64$sec) is quickly matched with a minimal deviation in the headway distance.

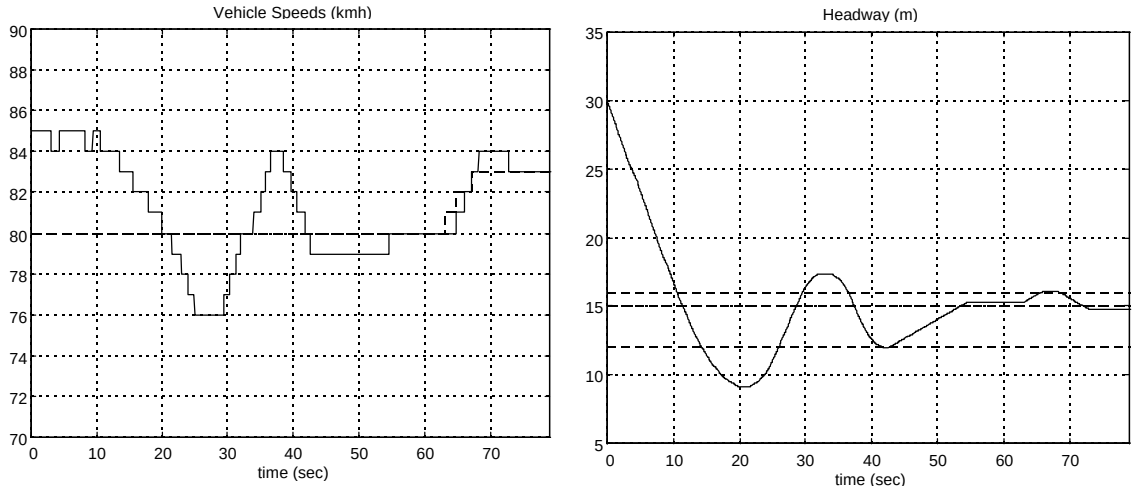


Figure 5.6. Headway distance and speed of an autonomous vehicle following another slow-moving vehicle: sensor parameters are d_1 = 12m, d_2 = 15m, and fsr = 16m.

From the simulation results above, we see that there is a trade off between the time to reach a steady-state value for headway distance and the permitted region for the steady-state values. For faster processing speeds (which decrease the time to reach a decision in the planning

layer) and shorter memory buffer (which results in a shorter decision time in the regulation layer), the regions may be smaller, and therefore the problem less insignificant.

The definition of the front sensor module in Section 4.2 is the limiting case for the extended version given here. As d_1 and *fsr* approach d_2, the definition given here becomes similar to the one given in Chapter 4, and oscillatory responses are observed.

The idea of having multiple parameters defining the regions [d_1 d_2] and [d_2 *fsr*] is similar to the idea of defining a boundary layer around a switching surface for sliding mode controllers in order to stop chattering [Slotine91]. In sliding mode control, additional integral terms are added to the 'driving function' for the predefined boundary layer. Here, we force the speed to change according to value and rate of change of the headway distance. Since distance is the 'integral' of the velocity, the resulting effect is similar. Of course, the control methodology is much more discrete, and also less structured, than in a sliding mode controller. In the design methodology of the sliding mode controllers, either the width of the boundary layer or the function parameters are predefined and the rest adjusted accordingly [Kachroo96c]. Similarly, in our case, the choice of processing parameters and the sensor regions are sequential, in the sense that we either define the sensor regions first, and then choose the learning parameters, or vice versa.

The region [d_1 d_2] is effective when the automated vehicle has to slow down; [d_2 *fsr*] is mainly important in the case of acceleration while closely following another vehicle. This second region also improves the vehicle's braking time in the previous case.

Also note that the length of the memory vector is 25, *i.e.*, at the very best, only one velocity adjustment per second is permitted. More continuous regulation layer modeling is of course possible, but not considered here. Since the controlled variable is the distance, not the speed, and a more realistic model would also include delays, we did not investigate continuous models. The aim here is to show the importance of sensor range definitions, and to introduce an intelligent vehicle control method based on learning.

5.1.2 Side Sensors

In the previous chapter, the side sensors are described as being capable of detecting vehicles and obstacles in adjacent lanes. This may be achieved using relatively easy then the method we will now describe. By using multiple sensor arrays, and combining different technologies such as sonar, IR, and visual sensing, it may be possible to not only detect the presence, but also the relative movement of vehicles in the adjacent lanes. Current side sensor implementations include radar, vision, and IR technologies. Vehicle control systems based on the inter-vehicle communications do not consider the use of side sensors since all vehicles are assumed to be controlled by a higher level in the hierarchy or all will be capable of relaying their position information to other vehicles. Autonomous navigation implementations have not yet reached the level of capability where inter-vehicle collisions is a consideration.

Millimeter-wave radar sensing [Zoratti95] and short-range vehicle-to-vehicle communications — a more global solution to the problem — are currently under development for prediction of inter-vehicle collisions [Asher96]. Forward looking sensors [Liu89] or binocular vision systems [Weber96] could also provide some information necessary for lateral movements [Liu89].

Assuming that there is a sensing structure — possibly an array of sensors— that is able to give information about the relative speed of the neighboring vehicles, the side sensor's detection area is divided into two sections as seen in Figure 5.1. An immediate neighborhood is now defined. In this region (D) the output of the sensor module depends not only on the detection of vehicles, but also on measurement of their relative longitudinal movements. The penalty-reward structure for the side sensor modules is given in Table 5.3.

	Sensor Status (Left /Right)		
Actions	Vehicle in region D	Vehicle in region E	
		Approaching	NOT approaching
SiL	0 / 0	0 / 0	0 / 0
SL	1 / 0	1 / 0	0 / 0
SR	0 / 1	0 / 1	0 / 0

Table 5.3. Output of the left and right sensor modules.

Longitudinal actions are not shown in the Table 5.3 since there are not affected by the side sensor modules. The regions D and E are defined as shown in Figure 5.1; the parameters sr_1, and sr_2 are used to define the detection range of the sensors (or sensor arrays). The parameter d differentiates the region D from the region E. If a vehicle is detected in the region E, and it is not (longitudinally) approaching the automated vehicle, the penalty response to the lane changing actions are suppressed. This distinction gives more flexibility to the automated vehicle as we describe with the following simulation examples.

Figure 5.7 shows five automated vehicles with sensor ranges $sr_1 = sr_2 = 12$m, $d_1 = 8$m, $d_2 = 11$m, $fsr = 15$m (the side sensor module of Chapter 4 is used in this simulation). The vehicle in lane 3 wants to merge into the platoon, however the distance between the first two vehicles in the platoon is only 20m. It slows down in order to change lanes (the behavior forcing this change is given in Section 5.3.1). The right side sensor module always returns a penalty response for the action SR, and consequently the vehicle is unable to change lanes. It slowly moves the to the back of the platoon.

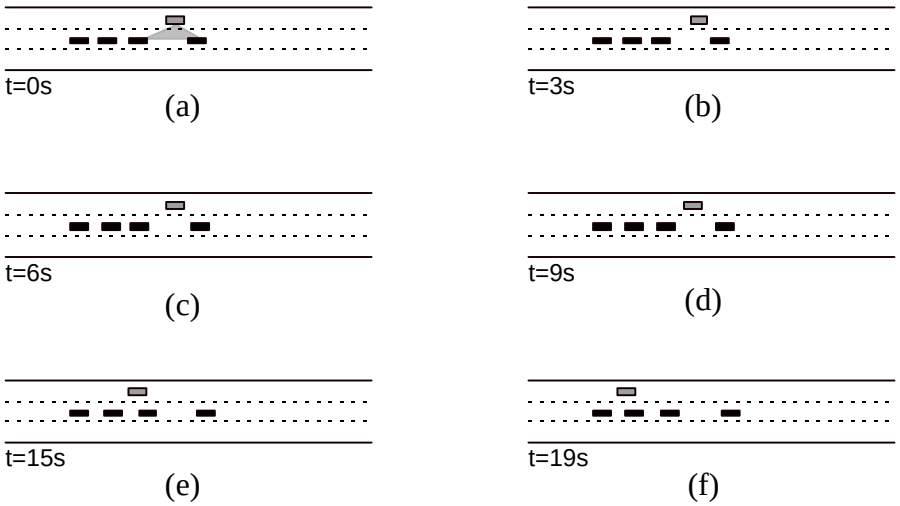


Figure 5.7. Positions of five automated vehicles: gray colored vehicle attempts to shift to the right lane by slowing down.

However, if we employ the extended side sensor module for the same scenario, the vehicle is able to change lanes. As seen in Figure 5.8, the vehicle in lane 3 slows down to change lanes, and shifts lanes at $t \approx 8.5$sec.

In this simulation, the side sensor ranges are $sr_1 = sr_2 = 12$m with the additional definition of $\mathsf{d} = 0.6\overline{6}$, *i.e.*, if the detected range is between 8m and 12m and the neighboring vehicle is *not* approaching, the action SR becomes acceptable. Using this decision structure, the gray colored vehicle shifts to lane 2, and adjusts its speed accordingly. Figure 5.9 shows the headway distances for the platoon. Before the lane change, the headway distance are steady with minor deviations due to the random speed changes (See Figure 5.10a). At $t \approx 8.5$sec, the lane shift occurs and the vehicle "entering" the platoon is cruising at a lower speed (See Figure 5.10). Consequently, the distance between the second and third vehicles decreases; the vehicle which is now the third in the platoon detects a sudden change in headway distance. As a result, this and other trailing vehicles slow down to adjust their headway distances. The effect can be seen in Figures 5.9 and 5.10b. The "ripple" is moving backward through the chain of vehicles. All vehicles are able to keep their headway distances within the acceptable range while returning to their desired speeds of 80kmh at $t = 30$sec. The steady-state response is due to the new definition of the front sensor module given in the previous section.

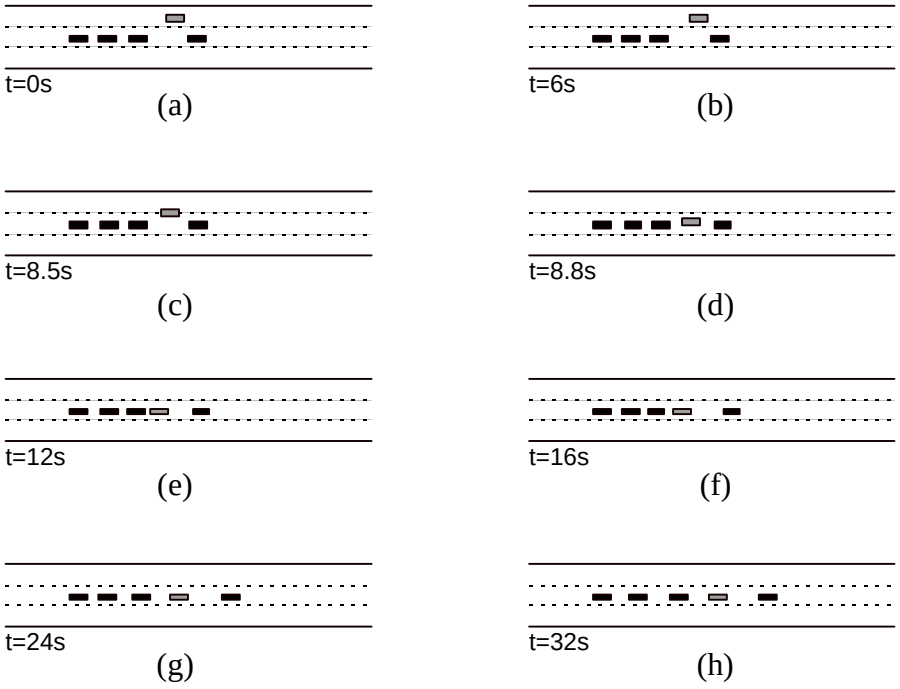


Figure 5.8. Positions of five automated vehicles: gray colored vehicle attempts to shift to the right lane by slowing down.

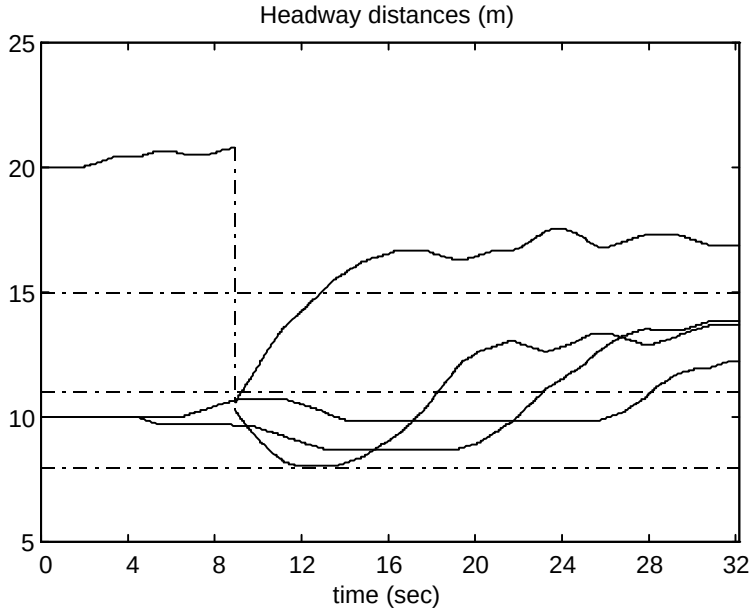
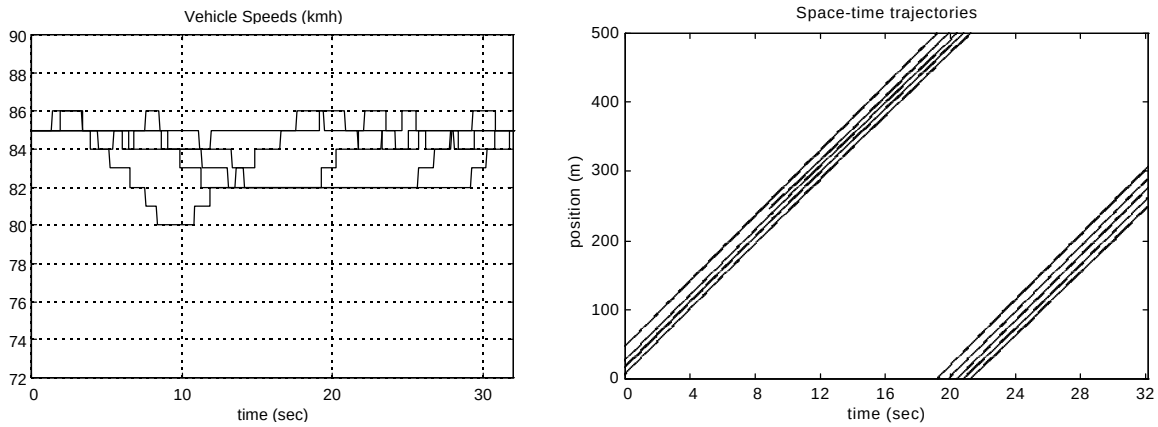


Figure 5.9. Headway distances in the platoon; lane change occurs at $t \approx 8.5$sec.



(a) (b)

Figure 5.10. (a) Speed and (b) space-time trajectory of five automated vehicles (Space-time trajectory for vehicle 2 is plotted after the lane change).

In the simulation example given here, the sensor ranges are chosen to illustrate the advantage of the new definition and the new front sensors' ability to stabilize the headway distance. However, with such small (side sensor) range definitions, the lane shift creates 'ripples' in the otherwise stable platoon. The main reason for defining multiple regions for the side sensors is actually to avoid sudden jumps of one vehicle in front of another. The sensor ranges must be carefully chosen, possibly using adaptive techniques mentioned in Chapter 1.

5.2 Additional Sensors and Inter-vehicle Communications

In addition to the changes made to the previously defined sensors, we now introduce two new modules which are more difficult to implement, and relatively more global than the previous definitions. The first module is visualized as a lane detector; the second as a pinch situation detector. While there are many possibilities for lane detector implementation, sensing a dangerous situation that may occur during lane changes is probably beyond the local sensing abilities of an automated vehicle.

5.2.1. Lane Detection

In the previous chapter, we assumed that the side sensor modules were inherently capable of sensing that the vehicle is in the leftmost or rightmost lane. Here, we will separate this function as a new module, and assume that not only the barriers and/or roadside are detected, but the vehicle can also sense which lane it is in.

A physical implementation of this module could be a vision system. Several AHS programs in US, Europe, and Japan use camera images for lateral and longitudinal control (see [Pomerlau96], [Malik95], [Özgüner96], and [Construction96] for recent examples). Extracting the lane information from such an image is becoming more and more trivial. Also, the magnetic markers used for lateral control may again provide position information. These solutions are still based on 'local' sensing; another alternative is to 'feed' the present and desired lane information

to vehicles via broadcast communications, as envisioned by the PATH program (e.g., [Hedrick96] and [Lygeros96]).

Once an automated vehicle knows its (lateral) position, and assuming that it can obtain the information on the best choice of lane for its desired speed and path, the decision for lateral movement is relatively straight-forward. Although current practical implementations do not yet consider such computationally complicated[1] maneuvers, research on control of lane changing maneuvers is slowly emerging [O'Brien96, Wonshik95, Hessburg95]. The difficult part in path planning is to find the optimal lane position for a specific vehicle.

For our purposes we will assume that an automated vehicle can sense its present lane, and that it has some idea about where it should be. Based on these two values, the action that leads to necessary lane shifts is encouraged by the teacher module shown in Figure 5.11. The flag structure described in Section 5.3.1 is used to force necessary longitudinal actions if the vehicle has trouble shifting into the desired lane.

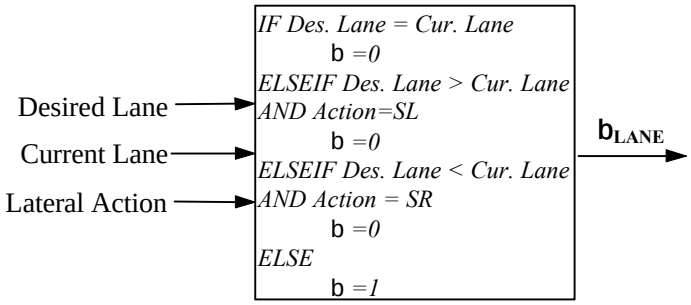


Figure 5.11. The lane module.

5.2.2. Pinch Condition Detection

Just like a driver who checks his rearview mirrors and looks to his side before changing lanes, it is imperative for an automated vehicle to make sure that the next lane is not "claimed" by another vehicle. The problem occurs when two vehicles one lane apart shift to the same spot in the lane between them (Figure 5.12). This situation is defined as "pinch condition." Besides the side sensors returning local data, vehicles "signaling" to shift lanes must be checked in order to make sure that a pinch condition does not occur.

[1] 'Computational difficulty' here describes the complexity of a good decision-making process for lane maneuvers; the low-level control of a lane maneuver is possible for most of the current test vehicles.

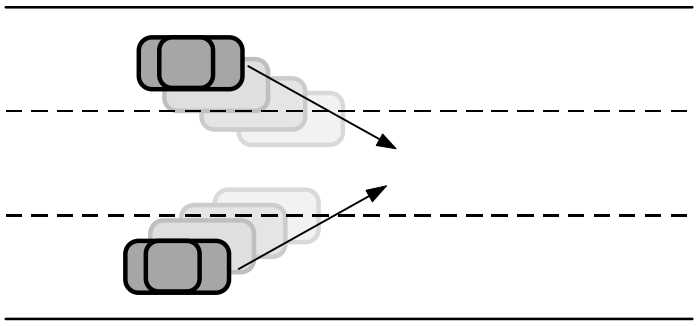


Figure 5.12. Pinch condition: two vehicles decides to shift to the same lane simultaneously.

In the simulations, we use the memory vector to check for other vehicles' intention to shift lanes. If the number of memory locations containing SR or SL is more that half the size of the vector for vehicle, it is assumed that the vehicle is likely to shift lanes (Figure 5.13). Since there are three lateral actions, a 50% distribution in the memory vector most likely indicates that an action will soon be "fired." If such an 'intention' signal is received from a neighboring vehicle, the pinch module sends a penalty response to the lateral automaton for the action that may cause a problem. In a sense, pinch module in an automated vehicle is driven by the memory vector of neighboring vehicles. In a real implementation, this corresponds to a signaling vehicle which may be detected by a vision system, vehicle-to-vehicle communications indicating intended actions, or a roadside-to-vehicle communication relaying the position of the vehicles. The last possibility is a global solution to the problem.

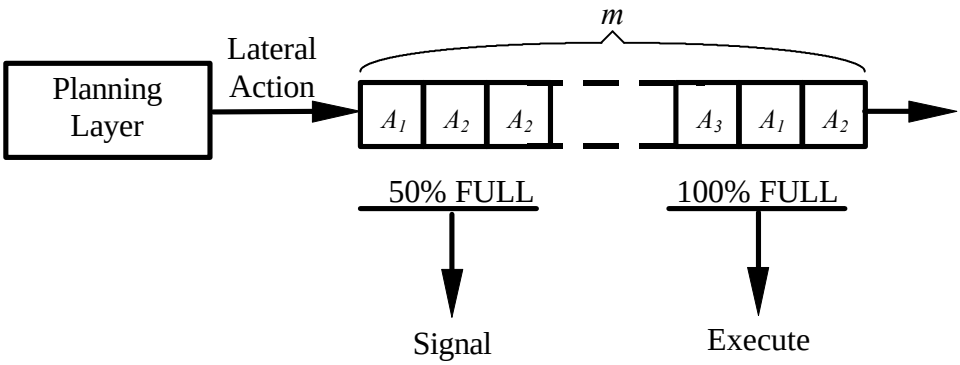


Figure 5.13. Memory vector/buffer: If an action fills half of the buffer, it is "signaled."

Figure 5.14 shows two vehicles cruising at the same speed. Initially, vehicle 1 is in the rightmost lane, vehicle two in the leftmost lane. However, vehicle 1 needs to shift to the leftmost lane, while the other wants to change to the rightmost lane. Figure 5.15 shows the lane and speed of the two vehicles versus time. Assuming that their initial positions in Figure 5.14 correspond to time $t = 0$, both vehicles are unable to change lanes until $t \cong 3$sec, then vehicle 1 shifts to the middle lane.

The reason for the delay is the pinch module in both vehicles. Without a detection method for this condition, the vehicles would change to the middle lane at approximately the same time,

and the result would be a collision. As seen in Figure 15.6, both vehicles lateral memory vector is initially filled with the idle action SiL. Since both vehicles are forced to chose the necessary lateral actions by the lane module, the percentages of the actions SL and SR in the memory buffers (of vehicle 1 and 2, respectively) increase. The percentage of the action SL in the memory buffer of vehicle 1 reaches 50% slightly faster than that of action SR in the vehicle 2 (Figure 5.16). As a result, the lateral action SR of the vehicle 2 starts receiving penalties earlier than vehicle 1, and the probability of this action decreases. The result is also a decrease in the percentage of the action SR in the memory buffer. The percentage of vehicle 1's SL action also drops (until $n = 40$, or $t = 1.6$sec), but never under 50%. Once the intention signal from vehicle 2 is stopped, action SL fills the memory buffer, and is fired at $n = 71$ ($t \approx 2.84$sec). The memory vector is then reset to *all-SiL*.

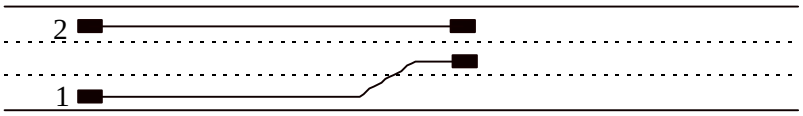
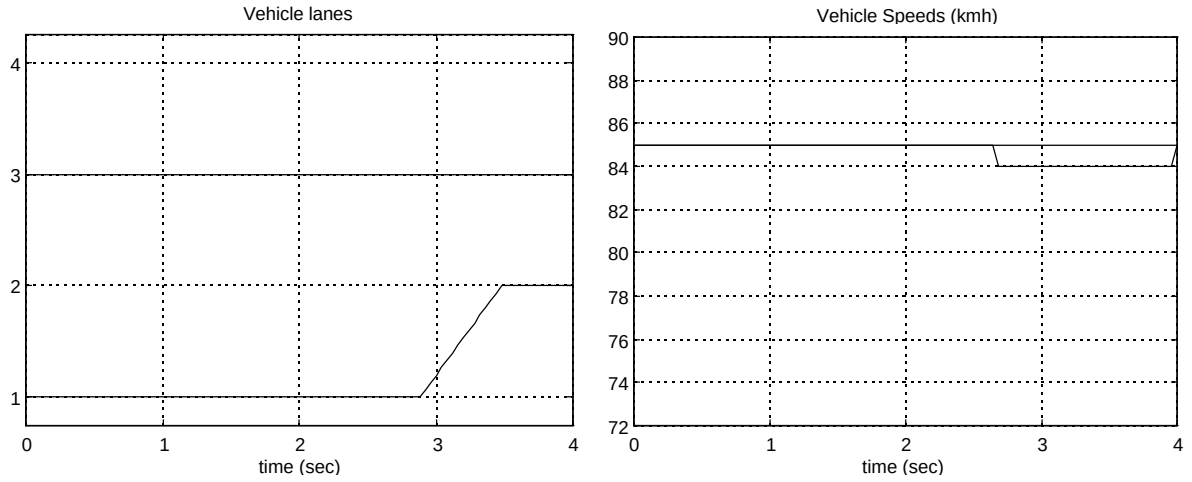


Figure 5.14. Initial positions and trajectories of two vehicles.



(a) (b)

Figure 5.15. Lane (a) and speed (b) of two vehicles from $t = 0$ to $t = 4$sec.

This new module enforces safe lane transitions; either one of the vehicles temporarily suppress its desire to shift lanes (*i.e.*, a penalty response from the pinch module overrides all other reward responses from lane and side sensor modules by ORing), or both vehicles are hindered from changing lanes to achieve their desired paths.

As seen in Figure 5.14, the two vehicles won't be able to 'exchange lanes' as long as they cruise at the same speed. Some bottleneck situation might also occur in a pinch situation where both vehicles have similar action ratios in their memory buffers. To overcome this problem, more complex decision rules need to be defined. These are explained in the next section.

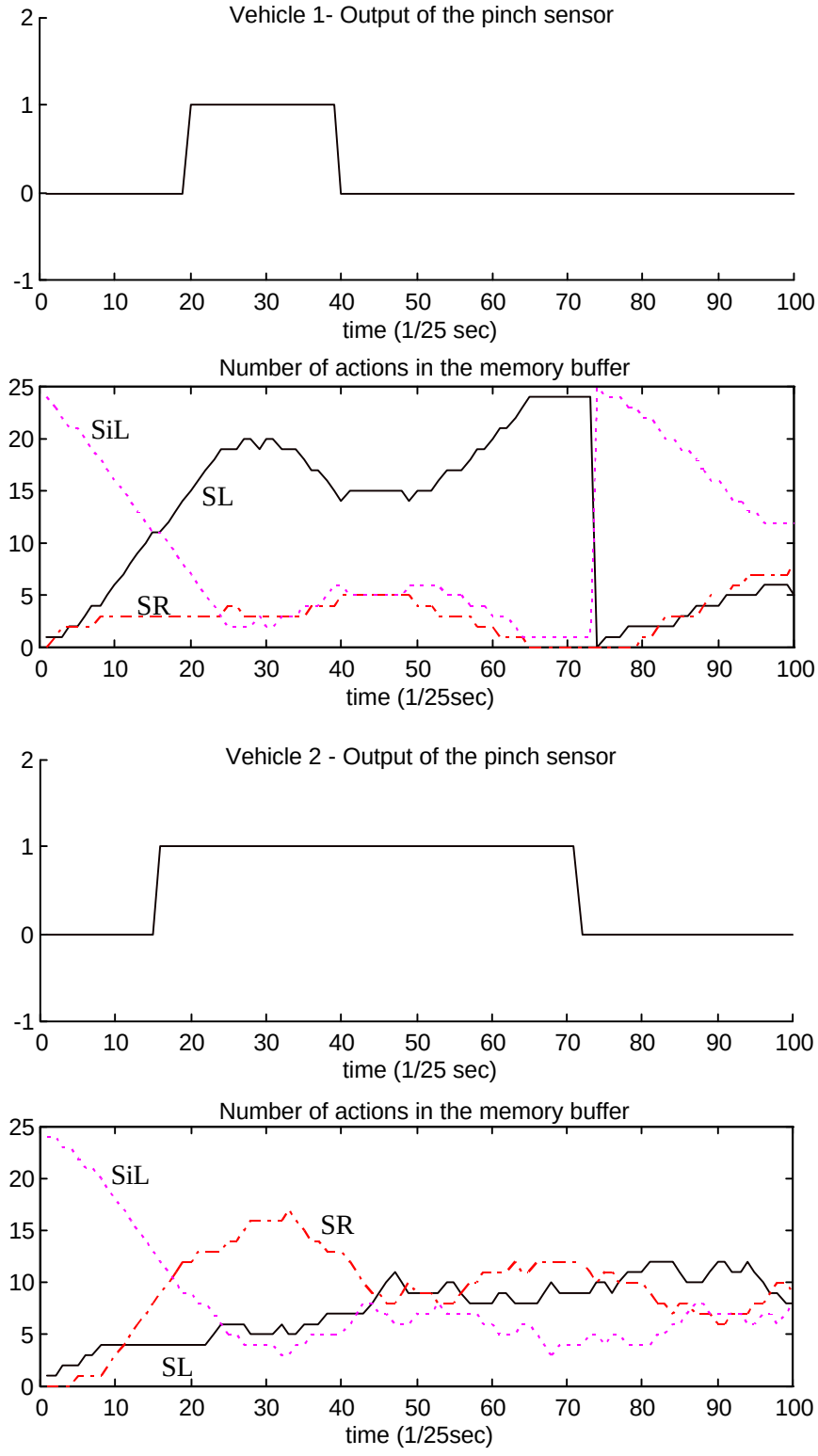


Figure 5.16. The output of the pinch module and the actions in the memory buffer versus time for vehicle 1 (top) and vehicle 2 (bottom).

5.3. Higher Level Additions: More Intelligent Teachers

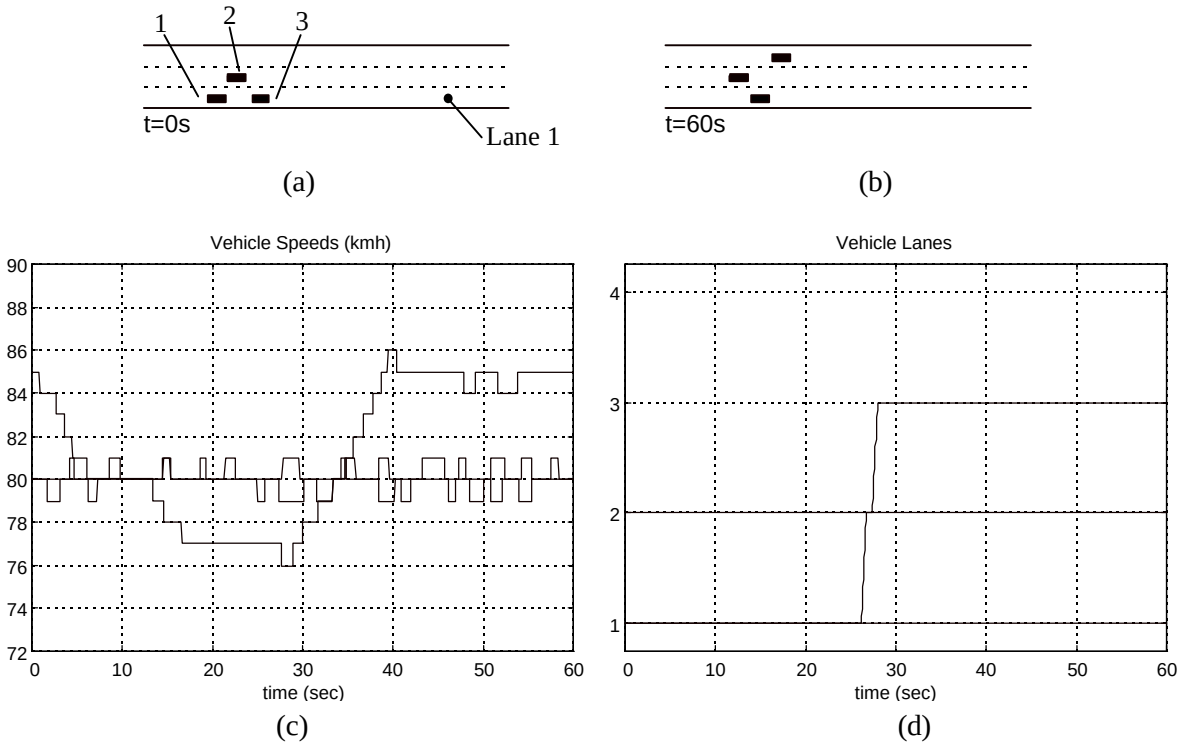
With the addition of new sensor modules and the extension of the previous ones, several changes need to be made to 'fuse' all sensor/teacher responses. In this section, we define the 'flags' used to temporarily alter the behavior of the vehicles in order to obtain a better path or to avoid bottleneck situations. The definition of the new mapping function and necessary changes are also given here.

5.3.1. Lane flag

The *lane flag* is defined in order to enable an automated vehicle to take an action if it cannot match its desired lane in a pre-specified time interval. A specific module keeps track of the time passed after a desired lane value is set (by on-board or roadside decision makers). If the vehicle cannot change to its desired lane in the predefined time interval, then the lane flag is set. The effect of this flag is to temporarily decrease the value of the desired speed by some fixed amount. As a result, the vehicle slows down and hopefully finds an opening to change lanes. Once the vehicle reaches its desired lane, the flag is reset and the vehicle slowly increases its speed to match the previous desired speed.

Consider the situation shown in Figure 5.17(a) where the velocities of the vehicles 1, 2 and 3 are 85, 80 and 80mph respectively. The desired speed for each vehicle is the same as its current speed. Vehicle 1 slows down to match the speed of vehicle 3 to avoid a collision, however it is unable to move away from the pocket created by vehicles 2 and 3. Assume that vehicle 1 "knows" that it has to be in lane 3 because its desired speed is faster than others. Also assume that an automated vehicle is capable of sensing its current lane. In the simulation run, the desired lane of vehicle 1 is changed to lane 3 at time t = 8sec. Since the vehicle is unable to shift lanes, after 4 seconds (the predefined time for lane changes) the lane flag is set, forcing the vehicle's desired speed to be redefined as 80-3 = 77mph. The amount of the change in the desired speed (3mph) is also predefined, and may be a function of current vehicle speed, average speed of other vehicles in the neighborhood, headway, and/or current lane and desired lane. Once the flag is set (at t = 12sec), the speed of the vehicle 1 starts decreasing to match its new desired value (Figure 5.17c). As a result, the vehicle finds an opening once its left sensor(s) clear vehicle 2, and shifts lane at $t \approx$ 27sec (Figure 5.17d). Upon reaching the desired lane, the flag is reset, and the speed again matches its previous desired value of 85mph (Figure 5.17c).

This lane flag is also used to get around the bottleneck situation given in Figure 5.14. For the vehicles to exchange lanes, they have to change their speeds to find an opening on their side. For this, the lane flag is designed to affect a vehicle that needs to shift to the right lane slightly differently than a vehicle that needs to shift left. The value of the 'adjustment' in desired speed with the lane flag is chosen differently to break the symmetry. Again, the temporary adjustment value for the desired speed may be a function of the present and desired lanes, solving the symmetry problem.



ch5m1.mpg
(0.273Mb; every frame represents one fifth of a second)
(e)

Figure 5.17. Initial (a) and final (b) situation, speed (c) and lane positions (d) of three automated vehicles traveling on a 3-lane 500-meter circular highway. The *mpeg* movie of the simulation is accessible via icon (e).

5.3.2. Speed flag

Similar to the method described above, another flag to temporarily change the desired lane is defined. If desired lane is not set, an adjoining lane is selected. For this flag, the predefined time interval must be larger than that of the lane flag since the time to reach the desired speed is much longer than the time required to change lanes.

A specific module keeps track of the time passed after the current speed deviates from the desired speed for the first time If the vehicle cannot adjust its speed in the predefined time interval, then the *speed flag* is set. The effect of this flag is to send a penalty response to the lateral action SiL. Consequently, the vehicle changes to the left or right lane, if there is an opening. Once the vehicle changes lane, the speed flag and the time counter are reset for the next interval.

Consider the situation in Figure 5.18(a) where an automated vehicle (#1) with desired speed 85mph is trailing another one with 80mph cruising speed. Vehicle 1 is able to slow down

and match the speed of vehicle 2, but it is unable to travel at its desired speed of 80mph (Figure 5.18(e)). The predefined time interval to match the desired speed is 10sec in this case. The count starts at t = 5.32sec (the '+' mark in Figure 5.18(f)), and the time expires at t = 15.32sec ('✳' in Figure 5.18(f)) while the automated vehicle is keeping a safe distance from the vehicle in front. Once the flag is set, the action SiL, which is normally encouraged to avoid unnecessary lane changes, starts receiving a penalty. The probability of one of the actions SR and SL, in this case SR^2, approaches one, and the action SR is fired at t = 18.96sec. The flag is reset when the vehicle completes the lane change maneuver at t = 19.52sec ('o' in Figure 5.18(f)). Immediately after the lane shift, the vehicle increases its speed to match the desired value.

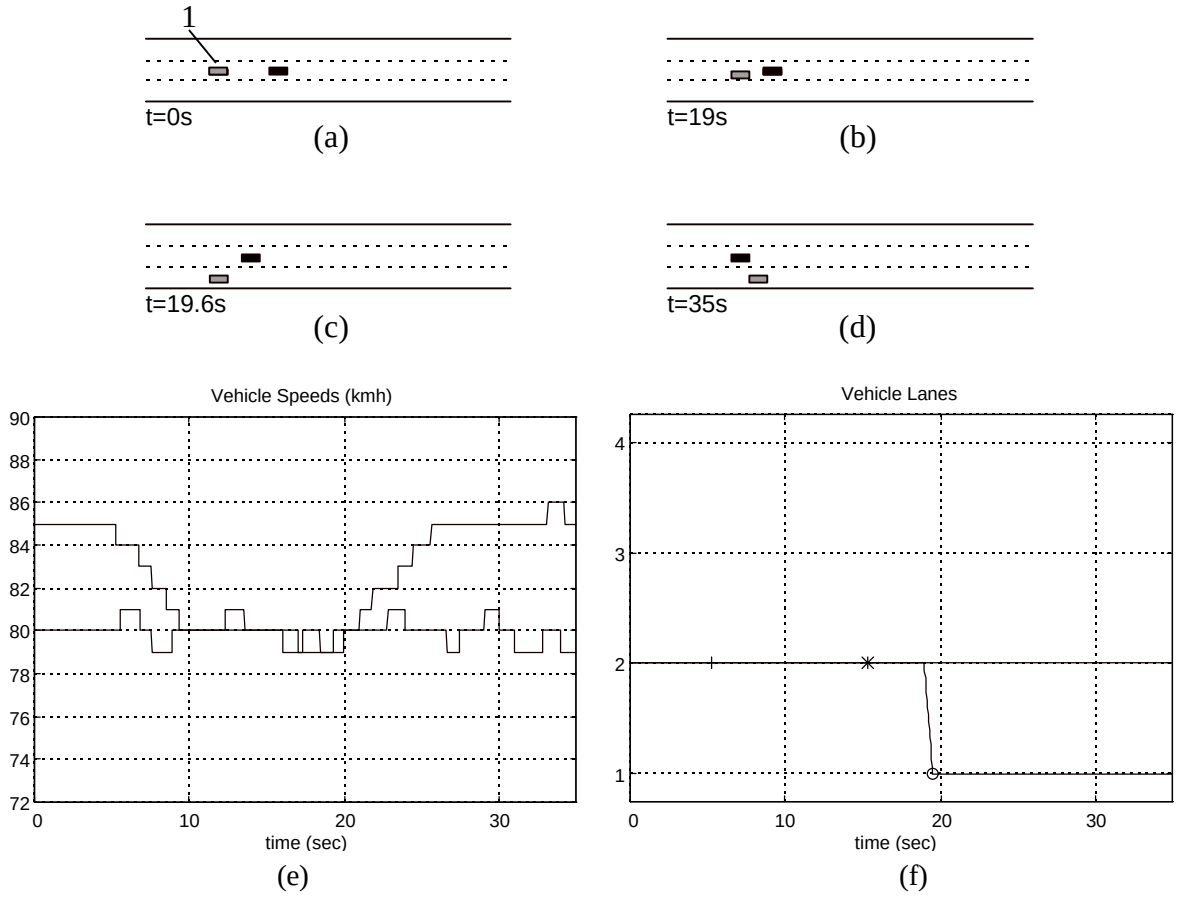


Figure 5.18. Snapshots (a-d), speed (e) and lane positions (f) of two automated vehicles traveling on a 3-lane 500-meter circular highway.

5.3.3. Additional rules

Addition of new modules and extending the capabilities of the previous sensors require changes be made to the function mapping multiple teacher outputs to a single automaton input. Although,

[2] Using the additional rule defined in Section 5.3.3.

the structure of the front sensor module is changed, the mapping that combines the teacher outputs for longitudinal actions does not change; the rules are exactly the same as the previous case given in Table 4.7 and Figure 4.4.

For the lateral actions, the changes in the front sensor and the addition of lane and pinch modules affect the mapping. The new function generating the automaton input is more complex than a single OR gate. For the lateral action SiL, the combined output is again an OR-ed combination of sensors except for one case. When the lane module sends a reward signal to the lateral automaton, but the headway module indicates an approaching vehicle, the action SiL must be penalized, as is the case with a simple OR function. However, in order to avoid unnecessary lane changes, the penalty response of the headway module is suppressed whenever the longitudinal action is DEC for headway distances not dangerously close (Table 5.4).

	Module Output				
Actions	**Right/Left**	**Lane**	**Headway**	**Pinch**	**Combined**
SiL		0	0		0
		0	1		1
	always	**0**	**1***	always	**1 or 0**
	0	1	0	0	1
		1	1		1
		1	**1***		1
SR or SL	0	0	0	0	0
	0	0	0	1	1
	0	0	0**	0	0
	0	0	0**	1	1
	0	1	0	0	1
	0	1	0	1	1
	0	**1**	**0****	**0**	**0**
	0	1	0**	1	1
	1	0	0	0	1
	1	0	0	1	1
	1	0	0**	0	1
	1	0	0**	1	1
	1	1	0	0	1
	1	1	0	1	1
	1	1	0**	0	1
	1	1	0**	1	1
Combined output depends on the longitudinal action (See text and Figure 5.19). All combined outputs for SR and SL are OR-ed side, lane and pinch module outputs except this one, if 0 and 0** are assumed to be the same binary value, FALSE.					

Table 5.4. Possible lateral action-sensor output combinations.

For actions SR and SL, again an additional *if-then* condition needs to be added for one case. Consider a situation where the side sensors and pinch modules indicate a reward for lane shifting actions SR and SL. The lane module, on the other hand, favors lateral action SiL because the vehicle is currently cruising in its desired lane. However, there is another vehicle in very close proximity of the vehicle and the front sensor module sends a reward signal to indicate the need for a lane change (if the headway distance is under the predefined limit d_1, we assume that the action DEC has failed to avoid collision). This reward signal must override the lane sensor's penalty signal (indicated with '$0^{\dagger\dagger}$' in Table 5.4). The resulting mappings from multiple teacher outputs to a single automata input are given in Figure 5.19.

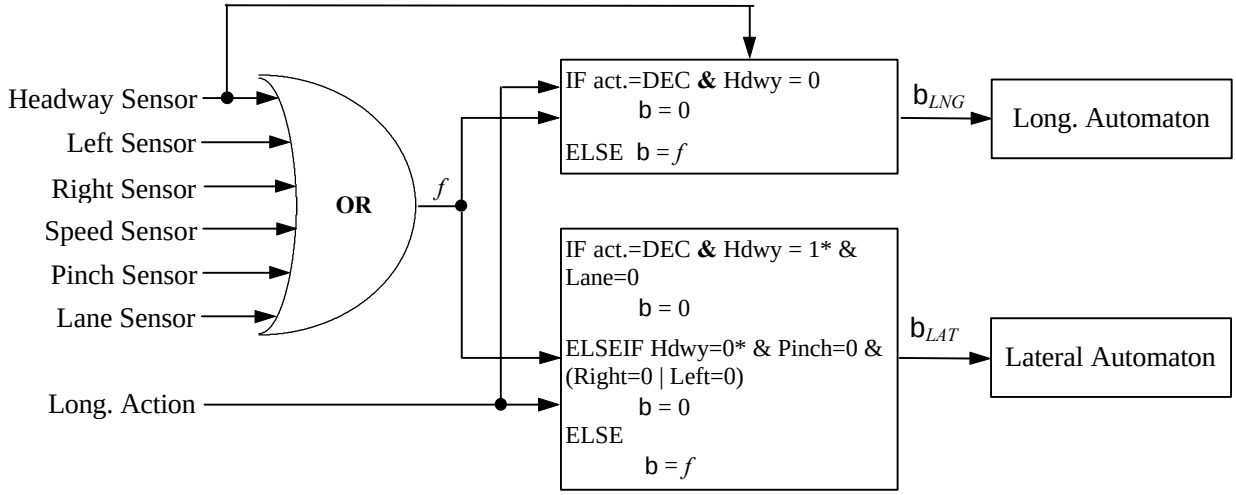


Figure 5.19. The new definition of the mapping *F*.

In addition to the rules described above, one more subroutine is defined to check the memory buffer interface to the regulation layer. Currently, if an automated vehicle senses a decreasing headway and nothing on the sides with no useful information from the lane module, it cannot decide whether to shift left or right. The reason for that is the lack of global information; the vehicle using only its local sensing devices cannot distinguish between the left or right lane, and the probability of both actions reach 0.5. To overcome the problem, a symmetry breaking rule which randomly chooses one of the actions is introduced. The overall structure of the intelligent planner is given in Section 5.4.1.

5.4. Results and Discussion

With all the additions described in this chapter, the planning layer of an automated vehicle is now more complex and robust than what we started with. The presentation of the final structure of the automata-teacher model, the discussion of the environment, the effect of parameters, and some final thoughts on the design will conclude this chapter.

5.4.1 Overall Structure

The final design of our automata-teacher planning layer consists of two automata — one for lateral, the other for longitudinal actions — a total of six teacher modules reporting on headway distance, speed deviation, left and right sensor detection, lane deviations and possible pinch conditions. The two 'flags' described in Section 5.3 are used to provide additional information to speed and lane detection modules (Figure 5.20). There are three additional rules that need to be implemented, besides the logical OR gate, in order to guarantee collision-free path control.

Furthermore, the automata are interconnected, not only through the physical environment, but also by the fact that the current output of the longitudinal automaton is used as a condition for the lateral automaton (Section 5.3.3). This is not necessarily important for collision avoidance, but it illustrates the fact that lateral and longitudinal actions cannot be treated as fully decoupled actions.

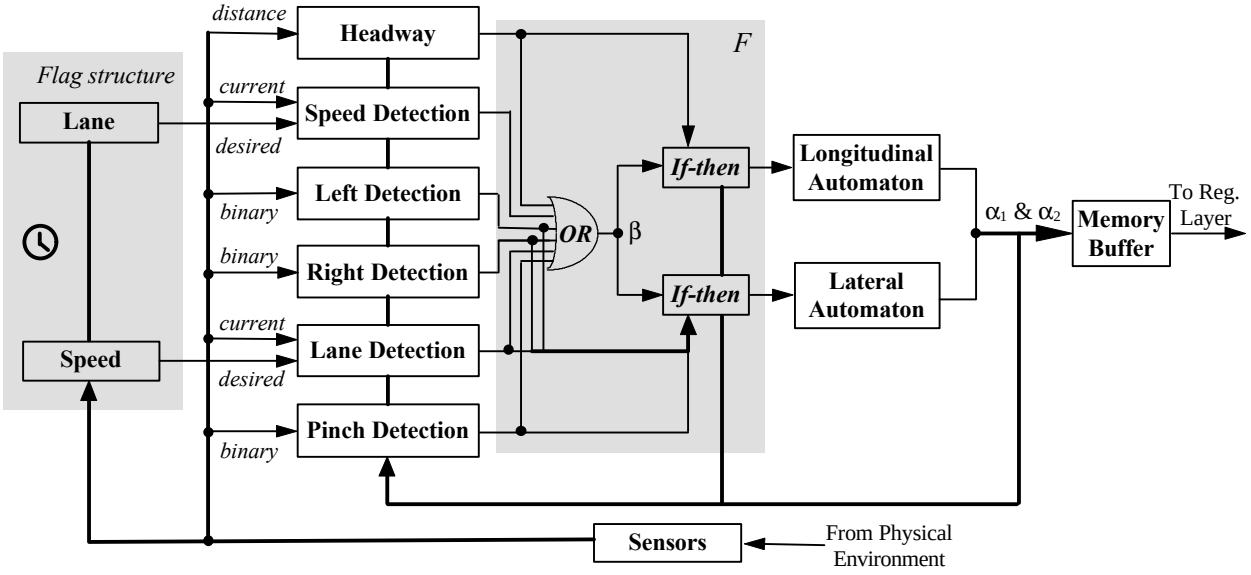


Figure 5.20. Automata in a multi-teacher environment with new definitions of mapping functions and flag structures.

Although not important for our design purposes, we chose to disable speed changes during lane shifting. Once the lane change is completed, the probability update for longitudinal actions continues. The longitudinal vehicle dynamics are modeled as simply as they can be; however, we assert that realistic vehicle dynamics will not affect the overall system behavior since the regulation and physical layers will react to 'fired actions' fast enough. Although not instantaneous, speed changes of 1m/s are not hard to obtain. For lateral movements, we assumed that an automated vehicle completes a single lane change so that the maximum lateral acceleration is less than or equal to $0.5 m/sec^2$.

The automata use the algorithms given in Chapter 6 as reinforcement schemes. The learning parameters are adjusted to give fast enough learning while keeping them relatively low to avoid instantaneous learning.

5.4.2 Effect of Parameters

There are many parameters affecting the behavior of an automated vehicle. In this section, we will give simulation examples illustrating the effects of parameters such as the length of the memory buffer, the definition of sensor ranges, and the learning parameters. The sample runs given here are simulations of platoons; only longitudinal actions are considered for demonstration purposes. The effects of the learning parameters on lateral actions is similar. Values of the parameters for each simulation is given in Table 5.5. General linear reward-penalty scheme is used in all simulations.

Sim. #	Learning Param.		Sensor limits (m)			Proc. Speed (Hz)	Memory buffer size	Figures
	a	b	fsr	sr2	Sr1		longitudinal	
1	0.15	0.10	16	15	12	25	25	5.21(a), 5.22(a) 5.23
2	0.15	0.10	16	15	12	25	9	5.21(b), 5.22(b) 5.23
3	0.15	0.10	20	15	10	25	25	5.21(c), 5.22(c) 5.23
4	0.05	0.05	16	15	12	25	9	5.21(d), 5.22(d) 5.23
5	015	0.10	18	15	10	25	25	5.24a, 5.25a, 5.26
6	0.15	0.10	18	15	10	25	9	5.24b, 5.25b, 5.26
7	0.15	0.10	18	15	10	25	9	5.27a
8	0.15	0.10	18	15	10	25	9	5.27b

Table 5.5. Parameter settings for simulations.

Figures 5.21 and 5.22 show the average platoon speed and the distances between vehicles for four platoons of ten vehicles, each using a different set of parameters. Positions and speeds of the vehicles for $t = 0$ and $t = 29.6$sec are given in Figure 5.23. All the simulations (#1-4) are run separately, and then the data combined for illustrations. Each platoon corresponds to a simulation run. Figure 5.23 also includes the *mpeg* movie of the four runs. The initial positions and speeds of the vehicles are the same for all runs as is the processing speed of 25Hz. The desired speed for all vehicles is 80kmh.

As seen in Figure 5.23 the positions of the vehicles at $t = 29$sec and their speed are different for each simulation run. All values plotted in Figures 5.21, 5.22, and 5.23 are not

averaged over a large number of runs with the same parameters, but the result for each set of parameters shows characteristics of a particular choice of parameters.

Simulations 1 and 2 (Figures 5.21a-b and 5.22a-b) illustrate the effect of the length of the memory buffer on the vehicle following behavior of an automated vehicle. For the same initial separation and velocity, the velocities of the vehicles reach a steady-state much faster when the memory buffer is shorter. The separation between vehicles is more uniform with a larger memory buffer. This is due to the fact that the system with the shorter buffer reaches the permitted region [12m 18m] for the headway distance faster. Once the desired region is reached, the reaction to the change in the feedback response is also faster. As a result, the separation between vehicles is more uniform for a large memory buffer, however the response is not stable; the opposite is true for a shorter memory vector length.

Changing the sensor range while keeping other parameters constant also affects the behavior of the vehicles in the platoon, as expected. Simulations 1 and 3 differ only in the choice of the values for d_1 and *fsr.* As seen in Figures 5.21(a, c) and 5.22(a, c), by increasing the region around the switching value d_2, it is possible to obtain a steady-state speeds 15 sec after the start of the run. The average speed of the platoon is still under the desired value of 80kmh since the headway distance is still too small for the action ACC to fire for some of the vehicles. Eventually all vehicles will reach their desired speed provided that the headway distance is above the safe value. An example of this behavior is seen in Figures 5.21c and 5.22c: as soon as the headway distance grows larger than d_2 = 15m, at $t \cong 29$sec, one of the vehicles increases- its speed, and the effect on the average platoon speed is seen. Furthermore, it is important to note that the initial vehicle separations are already in the defined sensor range. As seen in Figure 5.23, the overall behavior of the platoon in simulation 3 is closer to the one in simulation 1 than in simulation 2.

Simulation 4 is given to illustrate the importance of the choice of learning parameters. In terms of uniformity of the headway distances, the result of this simulation is worse. This is of course due to relatively small learning parameters. Although the length of the memory vector is the same as in simulation 2, the responses to the headway module are slow, and therefore the headway and speed variations are large.

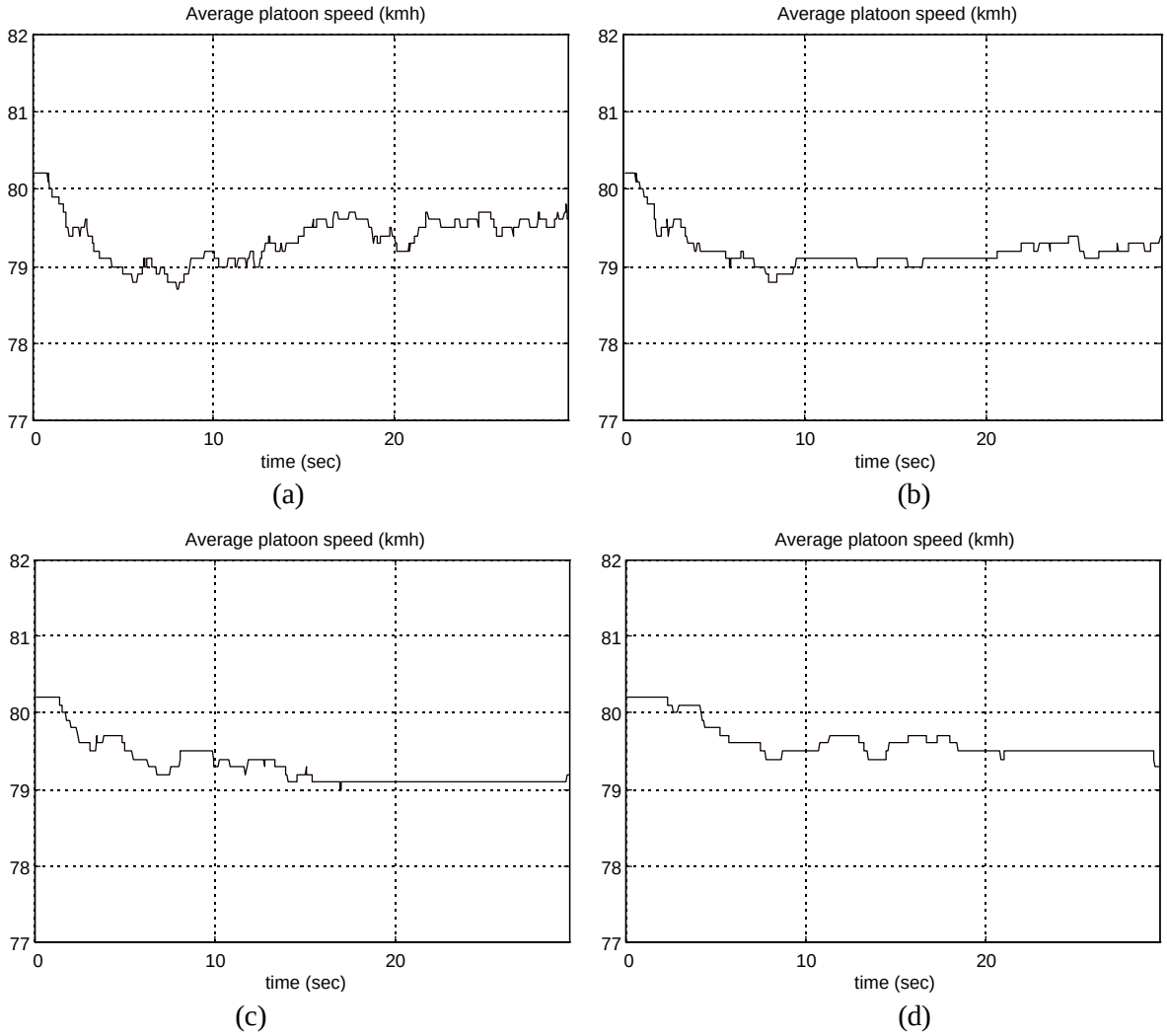


Figure 5.21. Average platoon speeds for simulations 1-4.

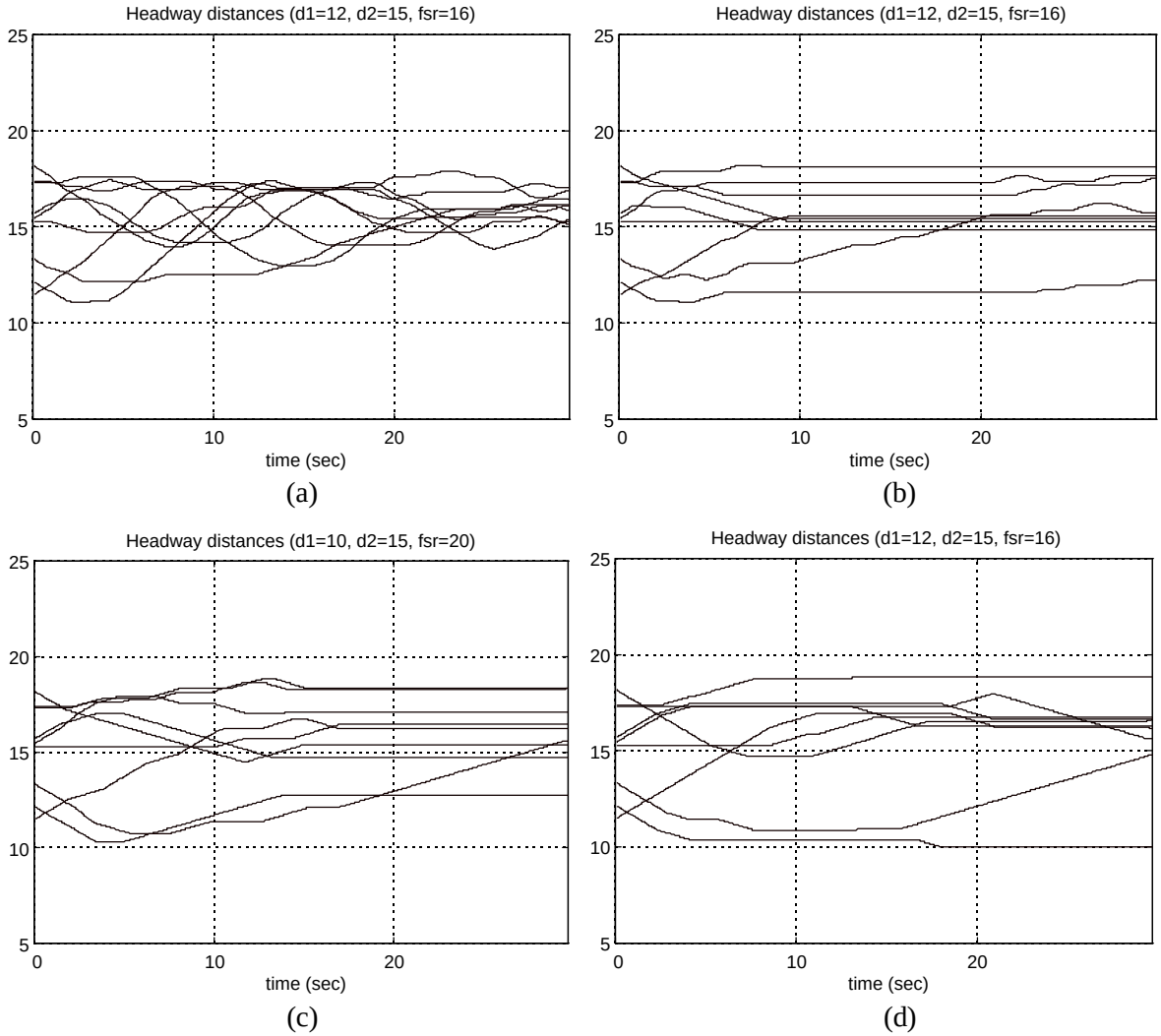
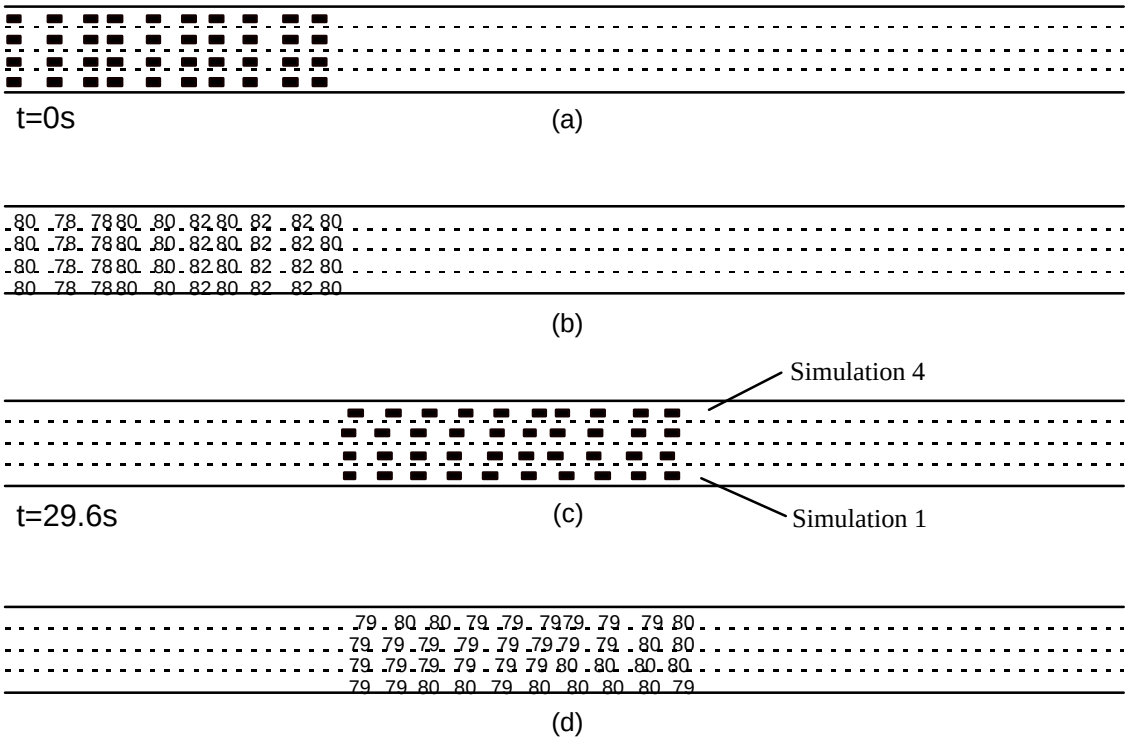


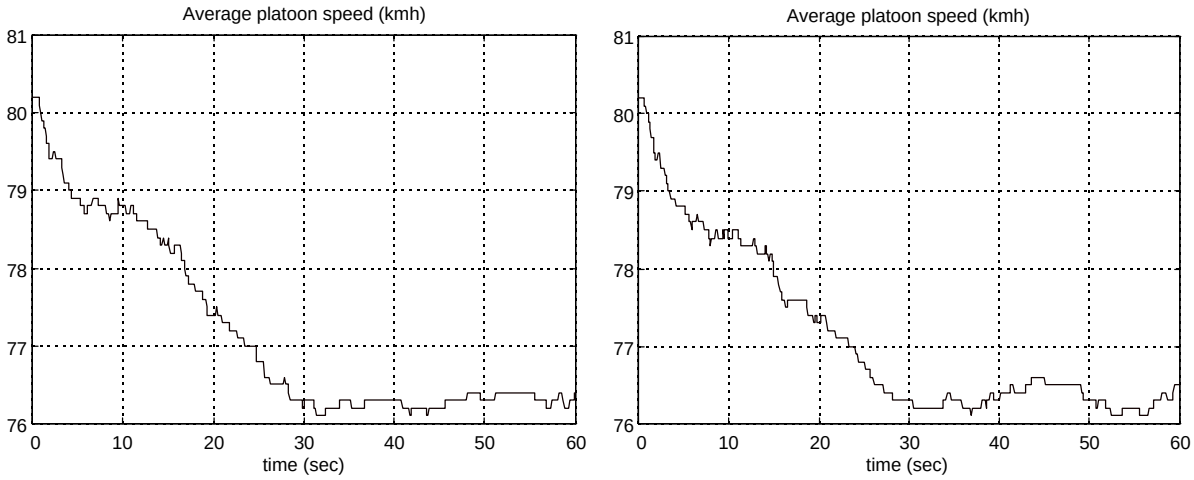
Figure 5.22. The distance between vehicles for simulations 1-4.



ch5m2.mpg
(0.237Mb; every frame represents one fifth of a second)
(e)

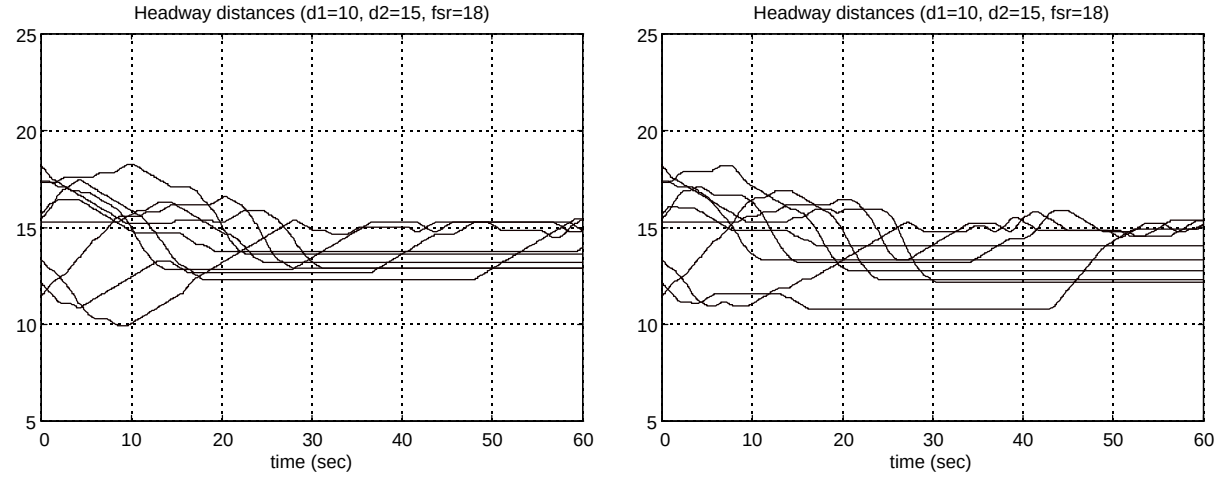
Figure 5.23. Initial and final positions (a,c) and speeds (b,d) of automated vehicles in four ten-vehicle platoons. The *mpeg* movie of the simulation is accessible via icon (e).

Figures 5.24-5.26 show the results of a second batch of simulations (numbers 5 and 6). In both simulations, the initial conditions are the same; the desired speed for all vehicles is still 80kmh except the leading vehicle which ‘desires’ to slow down to 77mph. In this scenario, the effect of the learning parameters is obvious: a fast learning vehicle will react faster to the speed changes (and therefore headway changes). Also, a very short sensor region may create a problem because it directly affects the reaction time. However, the effect of the length of the memory buffer is not that obvious. In simulations 5 and 6 (Figures 5.24 and 5.25), the only difference between two platoons is the size of the memory vector. There are no apparent differences in the behavior of the platoon. Again, the separations are distributed in a larger region for a smaller memory vector, but the difference is minor.



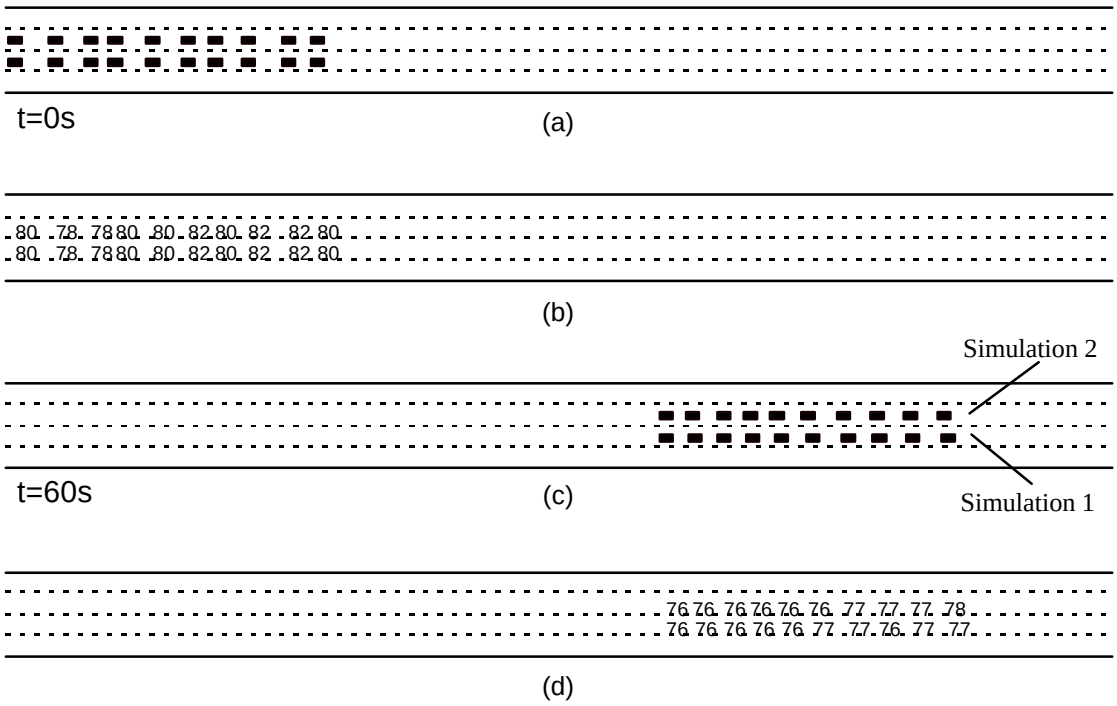
(a) (b)

Figure 5.24. Simulations (a) 5 and (b) 6: average speed for a platoon of ten automated vehicles.



(a) (b)

Figure 5.25. Simulations (a) 5 and (b) 6: distances between vehicles for a platoon of ten automated vehicles.



ch5m3.mpg
(0.345Mb; every frame represents one fifth of a second)
(e)

Figure 5.26. Initial and final positions (a,c) and speeds (b,d) of automated vehicles in four ten-vehicle platoons. The *mpeg* movie of the simulation is accessible via icon (e).

When the speed change for leading vehicle is much larger, the effect of the memory vector in platoon behavior can be seen much clearly. Two *mpeg* movies in Figure 5.27 show two unstable platoons. When the lead vehicle slows down to 75mph, the following vehicles are not able to keep their safe distances. Using the same parameters as before, a smaller length of the memory vector, resulting in faster firing of actions, gives a better result. With a memory vector of length 25, the third (or sometimes fourth) vehicle crashes into the vehicle in front, while with a smaller length of 9, the platoon stability is kept longer, and the problem occurs at the end of the platoon (seventh, eighth, or ninth vehicles). Since the speed (and headway) deviations increase as we move back in the platoon, an increase in the capability of the automated vehicles avoids an early problem.

The simulations given here are run using the general linear reward-penalty learning scheme. Examples of the nonlinear learning scheme and its comparison to linear schemes are given in Appendix C.

ch5m4.mpg
(0.065Mb; every frame represents one fifth of a second)
(a)

ch5m5.mpg
(0.100Mb; every frame represents one fifth of a second)
(b)

Figure 5.27. Simulations 7 and 8: *mpeg* movies of two simulations of 10-vehicle platoons with lead vehicle decelerating to 75mph: Length of the memory vector is (a) 25 and (b) 9, all other parameters are the same as in simulation 6.

5.4.3 Encountered Problems

5.4.3.1 Environment Model

In our application, the input to the automata is binary, and thus, the environment is said to be of P-model. Although the end result of the decision modules and functions is binary, we can visualize the method we use as a multi-valued logic. Additional conditions and the logical OR gate can be replaced by a simple logic function which takes multiple values into account. In this sense, the teacher outputs are somehow similar to a Q-model environment where multiple values in the range [0, 1] are possible. However, changing the learning schemes for a Q-model environment is tedious. In fact, there are no known applications of the Q-model multi-teacher environments.

The lack of presence of the S-model in this work is due to similar reasons. As stated in Chapter 6, the nonlinear reinforcement schemes are designed with S-model environments in mind. Our initial attempts to define teacher modules for the S-model have failed because of the difficulty in combining these outputs in a way that makes sense to the automaton. Again, the lack of previous work on the continuous environment feedback and the choice of weighting factors for the model leads to the difficulty. Nonlinear reinforcement models with continuous feedback responses are discussed in the literature, however there are no applications of this model, even for a single teacher case. The problem of choosing the appropriate weighting parameters is an open research area, and needs to be investigated.

5.4.3.2 Information Content and Other Issues

A great percentage of the previous research on intelligent control has emphasized the control methods and technological necessities for AVC, rather than making intelligent decisions, as we discussed previously in Chapter 1. In other words, the research mainly covers the regulation layer in [Varaiya93]. All AHS projects currently carried out in the world have not yet considered lane change maneuvers for implementation. Although the vehicles are individually capable of such actions, the coordination of such maneuvers will obviously be difficult to implement. In the case of a hierarchical control structure, there is not much need for autonomous decision making since the control commands will be the direct result of information fed by the hierarchy. The advantage of this approach is mainly in the content of the information. A hierarchical AHS system will have complete knowledge about each vehicle, and will be able to relay this ‘global’ information to

every agent on the highway. The autonomous vehicle approach, on the other hand, suffers from the lack of global information. While designing the intelligent controller, we assumed local sensing and minimal or no communications.

Solving the path planning problem locally may be relatively inexpensive and simple due to individual design goals and lack of communications, but the price to be paid may be the optimality of such a distributed system. Consider the situation given in Section 5.3.1 (Figure 5.17). For an autonomous vehicle to detect that the current solution is only 'local,' and does not satisfy the global goals (in the sense that the vehicle is capable of avoiding collision, but is not able to reach its desired speed), the information about the relative position (and maybe speed) of other vehicles needs to be known. This information can be provided by link layer in a hierarchical system [Varaiya93], or may be extracted using visual clues in the autonomous vehicle approach [Özgüner96, Pomerlau96, Weber96]. Again, the information content will be lower in the latter case. The need for *more global* information, rather than using only local sensors, is crucial and apparent, considering difficulties encountered during this research.