The University of York Department of Computer Science

Submitted in part fulfilment for the degree of BSc.

Undecided Title

Callum Hewitt

2016 Oct 12

Supervisor: Lilian Blot

Number of words = 2, as counted by `wc -w`.
This includes the body of the report only.

Abstract

Not completed

Not completed

Acknowledgements

Not completed

Contents

List of Figures

List of Tables

1 Introduction

Autonomous vehicles used to exist solely in science fiction. In 1953, Isaac Asimov wrote Sally [3]; in 1982, Knight Rider introduced KITT [4]; but today, autonomous vehicles can be found on roads around the world. Alphabet's WAYMO is gaining traction, with cars being tested in four different US states [5]. Tesla have deployed their beta Autopilot system into all of their vehicles produced since September 2014. The system has been both hailed for saving lives and blamed for ending them [6] [7].

All of the autonomous vehicle systems currently running on public roads are designed to work alongside human driven vehicles, limiting their benefits. In order to truly embrace autonomous vehicles, we need to design systems which assume every vehicle on the road is automated. The advantages of these systems are numerous, but the most important benefit is safety.

Autonomous vehicles would be able to react to incidents much more quickly than a human driver could. A driver's 'thinking distance' often determines whether someone survives an accident or not. This distance can be greatly increased if the driver of the vehicle is under the influence of alcohol or narcotics. An autonomous vehicle would be able to react to accidents much more quickly than a human, reducing the thinking distance and improving road safety.

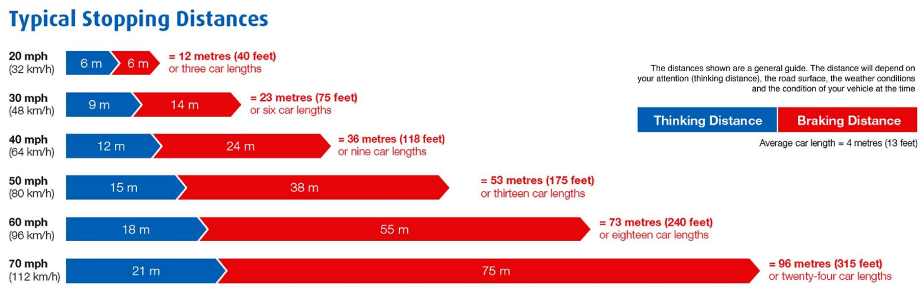

Figure 1.1: Diagram from Rule 126 in the UK Highway Code [1]

Another benefit of autonomous vehicles is efficiency. Research by Mersky suggests that fuel conservation strategies could make autonomous vehicles up to 10% more fuel efficient than current EPA fuel economy test results [8]. Fuel efficient vehicles are becoming increasingly important, with landmark climate change deals such as 'The Paris Agreement' introducing limits on greenhouse gas emissions globally [9]. The introduction of electric vehicles into the car market is also an important factor to consider, as the driving range of such vehicles has still not managed to match that of their gasoline counterparts. Efficient autonomous driving strategies could help bring electric vehicle range up to par.

Congestion contributes to fuel loss in quite a large way. In the US in 2014 an estimated 3.1 billion gallons (11.7 billion litres) of fuel was wasted due to congestion [10]. Automated driving strategies, in situations such as lane changes, could reduce congestion and improve efficiency. Dangerous lane changes don't even have to result in a crash to cause delays. If a car brakes due to a car merging dangerously, it can cause a ripple effect, creating a traffic jam. This ripple effect is known as a 'traffic wave' or 'traffic shock' [11].

As well as more quantifiable benefits, autonomous vehicles could also provide a level of comfort not currently available today. In a world where autonomous vehicles are commonplace, it is not hard to imagine people working, reading or relaxing in their car instead of focusing on driving.

However, today there are still a number of concerns surrounding autonomous vehicles. One of the major concerns is over the reliability of the systems governing the vehicle. These systems need to be responsive and accurate. They cannot afford to fail in such safety critical environments. Today, concerns over Tesla's Autopilot system are impacting the image of the company, and the system isn't even out of beta testing yet [12].

In order to address these concerns safely, we can create simulations which test the reliability of autonomous systems. Researchers at the University of Texas set up the Autonomous Intersection Management (AIM) project, which aims to "create a scalable, safe, and efficient multiagent framework for managing autonomous vehicles at intersections" [13]. The team managed to apply their simulator tested intersection software in a mixed reality test, using a real life autonomous vehicle [14]. This demonstrates how vital simulators are when testing safety critical systems.

In this project we make a number of assumptions. Firstly we assume that the sensors resolving the positions of the vehicle and it's surrounding obstacles are perfectly accurate. We also assume that all vehicles can reliably communicate with each other and with roadside infrastructure. These assumptions ignore existing areas of research which are not considered in this paper. The focus is on how autonomous vehicles can self-organise to minimise delays in traffic, using safe and effective lane merging.

The aims of this project are as follows:

- Attempt to generalise the AIM codebase such that other simulators can be created for non-intersection related situations.
  - Create a decentralised system for managing lane merging.
  - Create a centralised system for managing lane merging.
- Compare the effectiveness of both strategies.

2 Literature Review

2.1 Car Following Models

Most lane changing models stem from 'car-following models', which define actions for a vehicle based on the behaviour of its predecessors (the vehicles in front of it). One early car-following model was defined in 1981 by P.G. Gipps [15]. It was designed to mimic real-world driver behaviour, calculating a safe travel speed for a vehicle based on the speed of its predecessor. A safe travel speed is defined as a speed at which the driver can safely stop if the preceding driver stops.

Gipps' paper defines two equations, which provide constraints on the speed of vehicle n at time $t+\tau$. t is the current time and τ is the apparent reaction time, a constant for all vehicles. The first equation defines the acceleration constraint of the vehicle. It was obtained using measurements from an instrumented car.

$$v_n(t+\tau) \leqslant v_n(t) + 2.5a_n\tau\left(\frac{1-v_n(t)}{V_n}\right)\left(\frac{0.025+v_n(t)}{V_n}\right)^{1/2} \tag{2.1}$$

$v_n(t)$ is the speed of vehicle n at time t. a_n is the maximum acceleration the driver of vehicle n wishes to undertake. V_n is the target speed for vehicle n. The equation shows that the driver accelerates until close to their target speed. They then reduce their acceleration until it reaches zero. At this point the vehicle should be travelling at it's target speed.

The second constraint is the braking profile of the vehicle. This is given as

$$\begin{aligned} &v_n(t+\tau) \leqslant \\ &b_n\tau + \sqrt{\left(b_n^2\tau^2 - b_n\left(2\left[x_{n-1}(t) - s_{n-1} - x_n(t)\right] - v_n(t)\tau - \frac{v_{n-1}(t)^2}{\hat{b}}\right)\right)} \end{aligned} \tag{2.2}$$

b_n is the most severe braking the driver of vehicle n wishes to undertake. It is always a negative value, and should be considered negative acceleration. $\hat{b}$ is the driver of vehicle n's best guess at b_{n-1} where $n-1$ is n's predecessor. $x_n(t)$ is the location of the front of vehicle n at time t. s_n is the effective size of vehicle n. This is equal to the physical length of n, plus a margin n's successor is not willing to enter, even when n is at rest.

Therefore, at time $t+\tau$, assuming the driver travels as fast as is safe, and within the limitations of the vehicle, the speed is given by the minimum of these two equations.

$$v_n(t) = \min\left((2.1), (2.2)\right) \tag{2.3}$$

This model works well at describing the behaviour of traffic. However, translating this work to autonomous vehicles poses a number of problems. Firstly, the work is based on the behaviour of real-world drivers in instrumented vehicles. This introduces human driver variables into the equations. An autonomous vehicle with perfect sensors would have an almost negligible τ, as the vehicles would have a very minimal reaction time. s_n would also need to be adjusted. The margin added can be much less, as autonomous vehicles would be more precise than human drivers, driving closer to their predecessors.

The model also ignores inter-vehicle communication. Autonomous vehicles could communicate their intentions to nearby vehicles, allowing them to act before they do. This could greatly reduce the following distance of successor vehicles, it would also allow vehicles to accelerate and move as a unit, or a 'platoon'. It would also allow autonomous vehicles to gain accurate value for $\hat{b}$.

In 2000 Treiber et al. suggested the 'Intelligent Driver Model' (IDM) [16]. In the IDM, the acceleration of vehicle α, $\dot{v}_\alpha$, is defined using a continuous function of its velocity, v_α; the distance to the rear of its predecessor, s_α; and the velocity difference of α and it's predecessor, also known as the approaching rate Δv_α. The vehicle interactions are solely based on α's relative acceleration to its predecessor. The model only provides position information for a vehicle in relation to its predecessor, and it does not provide its velocity at a given time, as Gipps' model does.

The IDM is broken into two components. The first describes the behaviour of a vehicle on a free road.

$$\dot{v_\alpha} = a^{(\alpha)} \left[1 - \left(\frac{v_\alpha}{v_0^{(\alpha)}} \right)^\delta \right] \tag{2.4}$$

Here $a^{(\alpha)}$ is the maximum acceleration of vehicle α and v_0^α is the desired velocity of α. δ is the acceleration exponent, which is typically 4.

The second component describes the behaviour of a vehicle as it approaches its predecessor.

$$\dot{v_\alpha} = -a^{(\alpha)} \left(\frac{s^*}{s_\alpha} \right)^2 \tag{2.5}$$

As the gap, s_α, between α and it's predecessor, gets closer to the desired minimum gap s^*, α decelerates.

Interpolating the two components gives us the IDM.

$$\dot{v_\alpha} = a^\alpha \left[1 - \left(\frac{v_\alpha}{v_0^\alpha} \right)^\delta - \left(\frac{s^*(v_\alpha, \Delta v_\alpha)}{s_\alpha} \right)^2 \right] \tag{2.6}$$

The desired minimum gap in the IDM varies dynamically with velocity and approaching rate. It is given by the following function.

$$s^*(v, \Delta v) = s_0^{(\alpha)} + s_1^{(\alpha)} \sqrt{\frac{v}{v_0^{(\alpha)}}} + T^\alpha v + \frac{v \Delta v}{2\sqrt{a^{(\alpha)} b^{(\alpha)}}} \tag{2.7}$$

The equation takes the bumper-to-bumper space $s_0^{(\alpha)}$, also known as the minimum jam distance, and adds the comfortable jam distance $s_1^{(\alpha)}$. The bumper-to-bumper space is the minimum gap between α and its predecessor in stationary traffic. The comfortable jam distance is an extra distance added on for comfort, and to allow for a slower driver reaction time. In the paper, this value is set to 0. We can also consider it negligible for autonomous vehicles. T is the safe time headway; it represents the time required for the vehicle to safely come to a stop. Finally $b^{(\alpha)}$ is the desired deceleration for α.

The IDM does not attempt to directly mimic human behaviour in traffic situations. It models a general acceleration and braking profile for a given vehicle. As such, it is well suited for adaptation by autonomous vehicle models, as seen in [17]. However, similarly to Gipps' model, the standard IDM only applies to single lane traffic. It also ignores inter-vehicle communication and platooning opportunities.

In vehicle platoons, such as those analysed by Kamali in 2016 [18], each vehicle autonomously follow it's predecessor, with the lead vehicle controlling the overall pace of the platoon. Platoons make heavy use of vehicle-to-vehicle (V2V) communication to allow vehicles to join and leave, as well as to continuously control vehicle spacing and velocity.

Kamali developed a model for an automated platoon, defining procedures for vehicles joining and leaving.

A joining vehicle can integrate at either the back or the middle of the platoon. The vehicle first sends a join request to the platoon leader. If the vehicle is at the back of the platoon the leader sends an agreement and the vehicle follows its predecessor. If the vehicle requests to join in front of another platoon vehicle, the leader first asks the platoon vehicle to increase space; once the space is large enough for the joining vehicle, the leader sends an agreement. The joining vehicle then manoeuvres into the space and follows the preceding vehicle. Having now joined the platoon, the vehicle sends a confirmation to the leader. The leader then requests that the vehicle that gave way for the joining vehicle decreases their spacing back to normal.

A leaving vehicle sends a request to the leader. When it receives permission to leave the vehicle increases its spacing from its predecessor; once the vehicle is at its maximum distance from its predecessor the vehicle can change lanes. Once out of the convoy the vehicle sends an acknowledgement to the leader.

This model isn't very strict, acting as more of a set of requirements than a true model. The paper sets the requirements using pre-defined gaps, and has no strict calculations guiding following characteristics. It could be implemented using spacing rules from both the IDM and Gipps' model, however, by using V2V communication, the lead vehicle can control the actions of all vehicles in its platoon. Instead of using IDM or Gipps' model, the lead vehicle can control the gaps between vehicles so that they all increase and decrease simultaneously. The gaps could be based on the platoon's velocity, perhaps using (2.7) from the IDM. By centralising control in this way, vehicle platoons can avoid the traffic shock effect [11].

2.2 Centralised and Decentralised

We can divide approaches to autonomous vehicles into centralised and decentralised solutions. Centralised solutions rely on an external agent to manage vehicles. Vehicles use vehicle-to-infrastructure (V2I) communication channels to send information and receive instructions from the external agent. Decentralised solutions use vehicle-to-vehicle (V2V) communication to let other vehicles know their state, their intentions and to arrange any complex actions that might affect surrounding vehicles.

2.2.1 Centralised Systems

The Autonomous Vehicle Intersection management system (AIM) described in [19] is an example of a centralised V2I system. The system works by dividing the intersection into a grid of $n \times n$ reservation tiles. Drivers 'call ahead' to the intersection sending information packets containing

1. The time the vehicle will arrive.
2. The velocity at which the vehicle will arrive
3. The direction the vehicle will be facing when it arrives
4. The vehicle's maximum velocity
5. The vehicle's maximum and minimum acceleration
6. The vehicle's length and width

The intersection infrastructure simulates the journey of the vehicle through the intersection, noting the tiles occupied by the vehicle at each time interval. If any cell is reserved at the same time step the intersection rejects the request. The driver will start decelerating and continue making requests until it obtains a reservation. It will not enter the intersection without a reservation, even if that means coming to a stop at the intersection.

A grid-based reservation system works well in high traffic zones like intersections, because it forces all vehicles to communicate with a single entity. This entity has a global-view of activity at the junction, allowing the system to make vehicle management decisions more easily. A V2V solution would require more complex communication protocols involving large numbers of vehicles. The volume of messages required for each vehicle to obtain a global view of the intersection would be considerably larger, and as such, most vehicles will never get a complete understanding of the status of the intersection.

A centralised system for lane changing was described in a paper by Atagoziyev et al. in 2016 [20]. This system uses roadside infrastructure to help groups of vehicles change lanes before they reach a 'critical-position', such as a motorway exit or intersection. The vehicles send their position and velocity information to the roadside infrastructure; the system then sends a number of orders to the vehicles such that they safely rearrange themselves into the correct lanes. Because the distance travelled by the vehicles involved can be large, particularly at high speeds, the system would struggle to use a reservation tile based system as AIM did, instead Atagoziyev's system manages the gaps between vehicles to safely relocate vehicles into the correct position. A more comprehensive overview is given in 2.3.1.

Atagoziyev's system would only need to be applied during the approach to critical-positions. Vehicles could be managed using platoons or another vehicle following model until that point. A centralised system provides a single communication point which manages all of the vehicles that want to change lanes. This helps to reduce the volume of communications required and creates an entity with a global view of the vehicles' positions and goals. A V2V solution would most likely require more communications and may never obtain a complete picture of the situation, possibly leading to sub-optimal lane changing orders.

Note that Atagoziyev's system could be adapted, such that all of the vehicles communicate with the platoon leader instead of roadside infrastructure. Though this could be called a V2V communication solution, the effective solution is still considered centralised, as all decisions are made by one entity.

2.2.2 Decentralised Systems

The main arguments against centralised systems generally tend to stem from concerns over feasibility and fault tolerance. A centralised V2I solution relies on one system always being available to manage vehicles. The original implementation of the AIM system works well, but if the system were to fail and no longer provide reservations then approaching vehicles will simply halt at the intersection. In a worst case scenario, the system would still give reservations, but fail to compare them to reservations already in place, causing major car crashes in the intersection. Having, a single point of failure (SPOF) like this is a major concern, particularly when lives are on the line.

A paper by Van Middlesworth et al. in 2008 [21] defined a decentralised version of the AIM model using V2V communication protocols.

In Van Middlesworth model each vehicle can broadcast two different types of message. These messages are broadcast repeatedly with a specified period.

1. CLAIM This is a message indicating the vehicle's intention to traverse the intersection. It provides the vehicle's VIN, arrival lane, turning direction, arrival time and exit time. It also provides a message id, which increments when a new message is broadcast. Finally, the CLAIM message contains a boolean indicating whether the vehicle has stopped at the intersection.

2. CANCEL This message releases any currently held reservation, it contains the vehicle's VIN and a message id, which acts the same as the message id in CLAIM.

Two CLAIM messages are in conflict if their paths, as determined by their lane and turn parameters, are incompatible and their time intervals, as determined by their arrival and exit times, overlap. To resolve the conflict Van Middlesworth defines a priority ordering over claim messages using the following rules:

1. If neither vehicle is stopped at the intersection, the claim with the earliest exit time has priority.

2. If both vehicles are stopped, the vehicle whose lane is 'on the right' has priority. This is defined similarly to current US 4-way stop rules.

3. If neither lane can be considered to be on the right the vehicle who is not making a turn has priority.

4. If no other priority order can be established, the vehicle with the lowest VIN has priority.

The protocol starts with approaching vehicles receiving messages from existing pending vehicles. An approaching vehicle may not start broadcasting it's own messages until it is within 'lurk distance' of the intersection.

Once within lurk distance the vehicle tries to make a reservation using a CLAIM message. The vehicle will continue generating CLAIM messages, searching for an available time block, increasing and decreasing its velocity as necessary, until it finds one that allows it to traverse the intersection without any other CLAIM having priority over it.

Once the vehicle has a CLAIM broadcasting it may need to change it if it's looking like the vehicle might be late to the intersection or if a competing CLAIM arrives with a higher priority. A vehicle might also change its CLAIM to take advantage of a newly available time slot. In this situation the vehicle must then send a CANCEL message and a new CLAIM. Once the vehicle reaches the intersection it must traverse according to its current CLAIM, broadcasting its CLAIM throughout the traversal. At this point, the vehicle has the highest priority CLAIM and cannot be interrupted.

The main drive behind the unmanaged AIM intersection was to reduce cost. Adding in new infrastructure to an intersection costs money, and it might not be considered worthwhile for small intersections with only one or two lanes on each side. An unmanaged, decentralised system like that described by Van Middlesworth would drastically reduce the cost to the state in creating automated road networks.

Cost also becomes a major issue for centralised systems when you consider fast moving situations such as lane changing on a motorway. To implement Atagoziyev's model, vehicles must remain in range of the roadside infrastructure. This would mean that the infrastructure will have to continue on for a long distance, which could become very expensive, especially given the number of critical-positions on a motorway. Decentralised solutions reduce these costs massively.

Two examples of decentralised lane changing models are Gipps' 1986 driver decision model [2] and the MOBIL model developed by Kesting et al. in 2007 [17]. These models are decentralised and as such do not have to rely on roadside infrastructure in order to change lanes. This greatly reduces the cost of both implementations and allows the vehicles to be more flexible as to when they change lanes, no longer having to wait until they reach the lead up to a critical-position supported by roadside infrastructure. This flexibility means that vehicles could change lanes to increase their average velocity rather than just changing lanes in order to make a turn or leave the motorway at a critical-position. It also allows vehicles to deal with unexpected situations far from any roadside infrastructure. For example, a broken down car blocking a lane can be evaded. There is more information on Gipps' 1986 model and MOBIL in 2.3.

2.3 Making lane changing decisions

There are a number of reasons that a driver would want to change lanes. The most obvious being that the journey the driver wishes to complete requires the vehicle to move into a different lane. In this case the vehicle *must* change lanes before it reaches a critical position. Beyond this position the driver will need to change their planned route, most likely extending their journey time.

Another reason a driver might change lanes is in order to increase velocity, with the aim of reducing journey time. In general, a driver will aim to change lanes if their average velocity in their current lane is much less than that the velocity it could be achieving in another lane.

2.3.1 Lane Changing to hit a target lane

In 1986 Gipps' modelled driver behaviour in real world circumstances, characterising the decisions a driver has to make in order to determine whether to change lanes [2]. The paper was designed to be used with the Gipps' 1981 car-following model [15], explained in 2.1.

The model itself is constructed as a flow chart, in which the decision nodes are the choices a driver must make.You can see the flowchart in Figure 2.2.

After determining whether a lane change is feasible the model considers whether the driver needs to move into another lane because they are heading towards a critical point.

These decisions are modelled in nodes 3 and 4.

3 *Driver behaviour close to the intended turn*

If the driver is close to their intended turn then they will always attempt to change into their preferred lane. Only if blocked will they consider moving into another lane.

'Close' varies depending on regional differences and the level of traffic, but in the model, close is defined as the driver being within a distance equal to ten seconds of travel from the turn at the driver's desired speed.

4 *Urgency of changing lanes*

The urgency of changing lanes increases as the driver gets closer to their turn. The willingness of the driver to brake harder and accept smaller gaps increases as the driver gets closer to their intended turn.

In the implementation, the braking rate a driver is willing to when first becoming close doubles by the time the intended turn is reached.

D_n is the location of the intended turn

V_n is the desired (or free) speed of the driver

b_n^* is the most severe braking the driver would otherwise be willing to undertake

$$b_n = \left[2 - V_n \frac{(D_n - x_n(t))}{10}\right] b_n^* \tag{2.8}$$

Similarly to Gipps 1981 car-following model, this driver decision model is based on human driver behaviour and as such falls into similar pitfalls. There could be more optimal driver behaviours which would ensure that a driver is in the correct lane well before the critical position. However, because Gipps 1986 model is designed to model human driving behaviour we can expect it to perform less than optimally.

Atagoziyev's model is designed around autonomous vehicles, which allows it to take advantage of vehicle-to-vehicle communication. The model manages a set of vehicles over two lanes, organising their movements into their preferred lanes before they reach a critical position.

To do this the model keeps track of the gaps between vehicles and their relative speeds using roadside infrastructure. Then, a series of equations, each describing a different situation, are used to determine the behaviour of the vehicle.

In the paper SV (subject vehicle) refers to the vehicle that wants to change lanes. CL (current lane) is the vehicle in front of the SV. TL (target lane) is the vehicle the SV wants to be behind in its target lane. LV (lag vehicle) is the vehicle that will be behind SV once it moves to its target lane. Atagoziyev defines seven equations for manipulating vehicles between lanes. They are used in different contexts, each based on the relative positions of the surrounding vehicles. The contexts for each equation are given below, along with the behaviour from the equation during that context.

Case 1 *SV too close to CL in its current lane or TL if it changed lanes.*

SV slows down until the gap is sufficiently large enough.

Case 2 *SV has a large gap between itself and TL and CL, however, LV is too close to SV*

SV can approach CL and TL as long as the gap remains large enough. LV needs to open up a sufficient gap behind SV.

Case 3 *SV has the minimum allowable gap to TL and CL, but LV is too close*

SV follows the closest leader and waits until LV creates the necessary gap.

Case 4 *The gaps between SV and CL/TL/LV are sufficient. But CL/TL are not travelling at the 'nominal speed' established for all vehicles in this exchange*

SV maintains a sufficient gap, waiting for CL/TL to travel at nominal speed again.

Case 5 *SV and CL/TL/LV have sufficient gaps and CL/TL are travelling at nominal speed.*

SV performs the lane change, maintaining nominal speed.

Case 6 *SV obtains the minimum gap to CL/TL and LV maintains a sufficient gap. CL/TL are not travelling at nominal speed*

SV maintains the minimum gap, waiting for CL/TL to travel at nominal speed again.

Case 7 *SV obtains the minimum gap to CL/TL and LV maintains a sufficient gap. CL/TL are travelling at nominal speed*

SV performs the lane change, maintaining nominal speed.

These equations are the building blocks that lead to a lane change. The flowchart in Figure 2.3 shows how they work together to enact a single lane change.

Using this algorithm, we can move multiple vehicles into their correct lanes. LV always provides space to allow each changing vehicle into the correct lane. All of the vehicles are working with each together, such that they can all reach their goal. Comparing this to Gipps' 1986 model, where drivers are acting solely in their own interest, we can see that by having an external agent managing lane changes, the autonomous vehicles can achieve results that might not be possible in situations where drivers are acting selfishly. For example, in a grid locked situation, drivers following Gipps' model might never be able to move into their desired lane; however vehicles following Atagoziyev's model would open up spaces allowing vehicles to move through the traffic.

2.3.2 Lane Changing to improve overall velocity

Gipps' driver decisions model also considered situations where the driver does not have to be in any particular lane. The model considers the effects of transit lanes, heavy vehicles and the effect of the preceding vehicle on the driver's vehicle. These are shown in nodes 5 to 7 and 9 to 11 of Gipps' flowchart in Figure 2.2.

5 *Transit vehicles and lanes* Transit lanes are lanes dedicated solely for public transport and other high occupancy vehicles. These include vehicles such as buses, taxis and carpool cars. These vehicles are known in the model as 'transit vehicles'.

6 *Entry of nontransit vehicles into transit lanes* If there is an obstruction in the present lane, it is often considered to be a valid reason for a non-transit vehicle to enter a transit lane.

7 *Departure of nontransit vehicles from a transit lane* Once the obstruction has been cleared, nontransit vehicle must move back into a valid lane. This forced departure does not affect vehicles that are close to their intended turn.

9 *Relative advantages of present and target lanes* If the driver has not yet been forced to change lanes by any other factors, then they can look at the relative advantages of the present and target lanes, considering obstructions and then determining which lanes obstructions will have the least effect on their safe speed.

10 *The effect of heavy vehicles* If obstructions are level with each other or beyond the range a driver considers, then the driver considers the next heavy vehicle in each lane, as if it were the leading vehicle in an ordinary car following situation. The driver then selects the lane which will give them the higher speed.

11 *The effect of the preceding vehicle* If there are then no heavy vehicles, the driver considers the speed possible in each lane and then changes if they gain a 'sufficient' speed advantage. This is again, subjective, depending on the present lane, target lane and the type of vehicle.

Again, Gipps' 1986 model was built with human drivers in mind, and as such it fails to take advantage of the benefits of autonomous vehicles such as platooning and vehicle-to-vehicle communications.

Work by Kesting et al. in 2007 [17] describes a decentralised model of lane changing that lets vehicles change lanes to increase velocity whilst still ensuring that the overall traffic flow is not disrupted. This helps to avoid traffic shocks and maintains smooth traffic flow. In order to do this, Kesting introduces the MOBIL or 'Minimising Overall Braking Induced by Lane Changes' model. The model uses two criterion that the vehicle must satisfy.

The first criterion deals with safety, ensuring that the deceleration of a successor vehicle $\tilde{a}_n$ in the target lane doesn't exceed a safety limit b_{safe}.

$$\tilde{a}_n \geq -b_{safe} \tag{2.9}$$

This criterion effectively puts a limit on the level of braking a vehicle changing lanes can cause another vehicle to undergo if it pulls out in front of it.

The second criterion is the 'incentive criterion' which is what motivates a driver to change lanes. This criterion introduces a 'politeness factor' p which expresses the extent to which nearby vehicles affect a driver's lane changing decision.

The paper discusses the differences between symmetric ('US') lane changing rules and asymmetric ('European') passing rules, however in this paper we only use US lane changing rules. This gives the incentive criterion:

$$\underbrace{\tilde{a}_c - a_c}_{\text{driver}} + p(\underbrace{\tilde{a}_n - a_n}_{\text{new follower}} + \underbrace{\tilde{a}_o - a_o}_{\text{old follower}}) > \Delta a_{th} \tag{2.10}$$

$\tilde{a}_x - a_x$ is the utility a driver x gets due to the lane change, where $\tilde{a}_x$ is the acceleration of vehicle x after the lane change and a_x was their acceleration before the lane change. c is the vehicle changing lanes, n is the vehicle behind c once it changes lanes, and o is the vehicle following c before the lane change. Δa_{th} is the threshold at which the driver will change lanes. It is designed to model inertia. A driver won't change lanes unless they get above a specific utility gain. The politeness factor p varies from 0 to 1, where $p = 0$ is the most selfish behaviour and $p = 1$ describe drivers who won't change lanes unless collectively all of the drivers gain a utility greater than the threshold. When $p > 1$ drivers won't change lanes at all if it negatively affects the surrounding traffic, drivers will even go so far as to execute lane changes which reduce their own utility. Likewise drivers with $p < 0$ will go out of their way to negatively affect other drivers, even reducing their own utility to do so.

The idea of a MOBIL model means that drivers will only change when it increases the sum of all of the accelerations increases. This would be at $p = 1$ and $\Delta a_{th} = 0$. In this case the equation becomes

$$\tilde{a}_c + \tilde{a}_n + \tilde{a}_o > a_c + a_n + a_o \tag{2.11}$$

Kesting found that the most important parameter affecting the rate of lane changing was p. With a p value of 1 the maximum lane changing rate was almost halved. Kesting also discovered that 'altruistic' lane changing behaviour increased the mean speed of both lanes involved in the simulation, improving overall traffic performance.

Comparing this to Gipps' 1986 approach we can see that MOBIL is far more considerate of other drivers and as such the overall speed of the vehicles on the road could be higher. MOBIL is also designed for autonomous vehicles, which allows it to enforce 'altruistic' behaviour from vehicles on the road. The model could also be extended to include V2V communications. Transmitting accurate braking profiles to other vehicles would make it easier to determine the effect a lane change would have on the overall system.

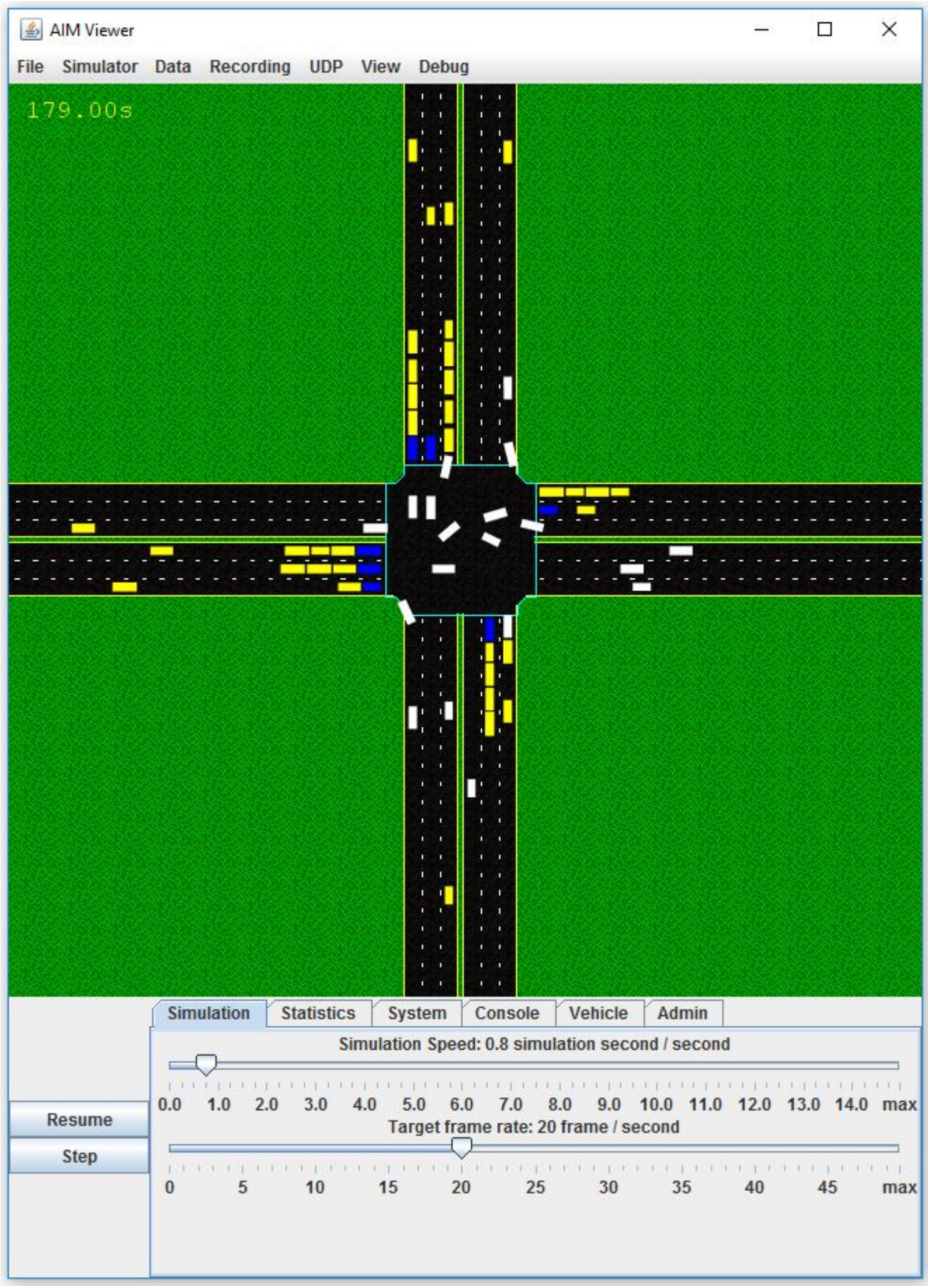

Figure 2.1: Screenshot of the AIM Simulator

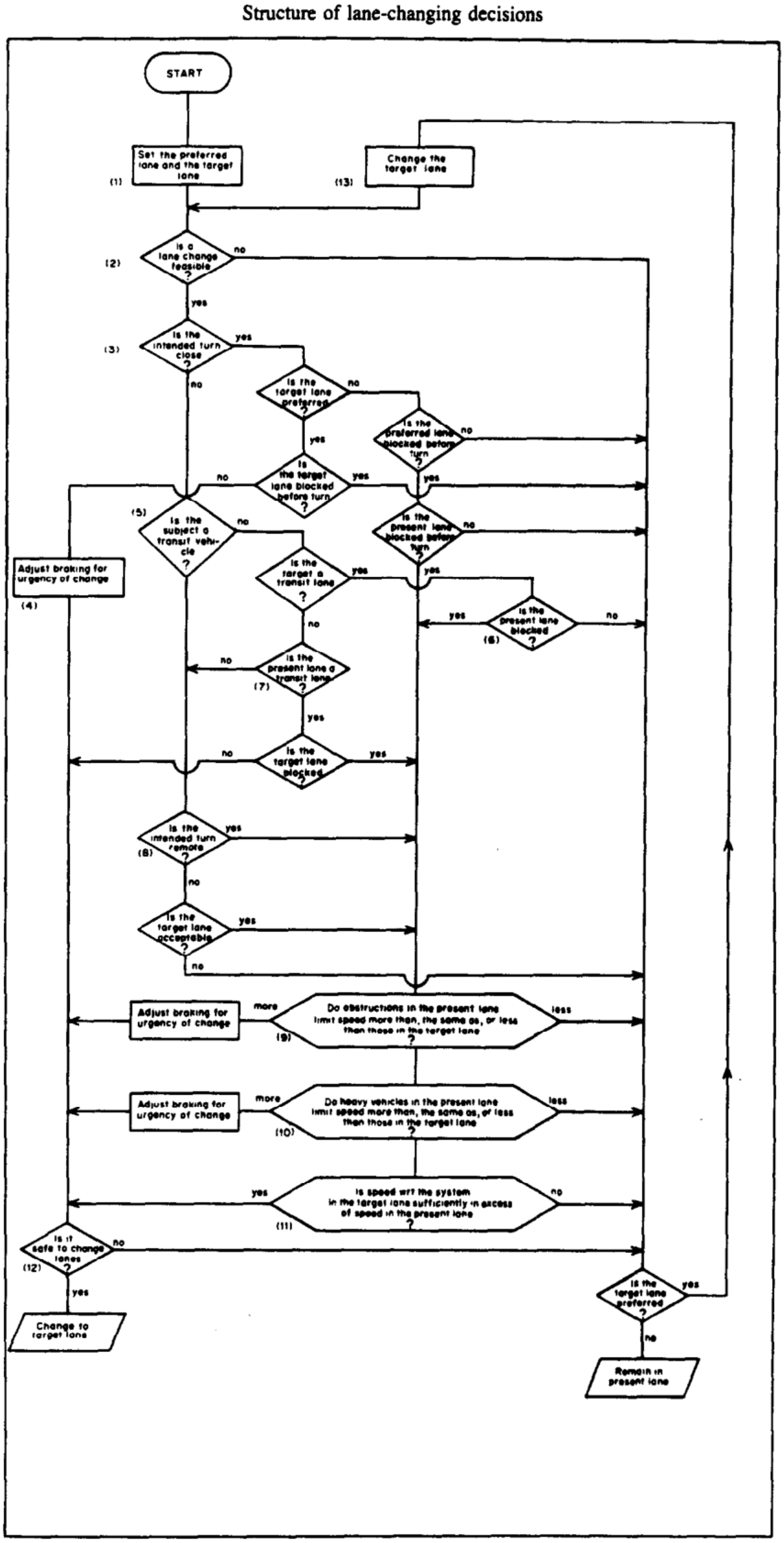

Figure 2.2: The flowchart for lane changing decisions from [2]

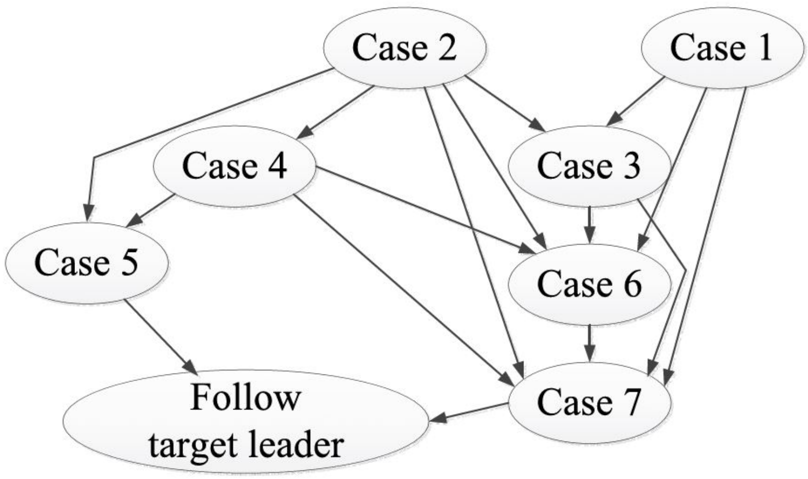

Figure 2.3: The flowchart for Atagoziyev's Lane Changing model equations

3 Problem Analysis

3.1 Lane Merging Problems

Vehicles may have to merge into another lane for a number of reasons. In this paper we focus on merges made at 'critical positions' such as junctions. This analysis could later be applied to merges made at non-critical points, though centralised approaches may struggle here.

3.1.1 Single-to-Single Merge

A single-to-single merge (S2S merge) describes a situation where a vehicle moves from a single lane road into another single lane road, as seen in Figure 3.1. In this situation we label the lane that vehicles are moving from the 'current lane' (CL), and we label the lane that vehicles move to the 'target lane' (TL). We describe the vehicles that start on the CL as 'merging vehicles' (MV) and the vehicles that start on the TL as 'target vehicles' (TV). We have our critical position where the CL and TL connect.

The main issue with an S2S merge stems from the limited options available to vehicles arriving at the critical position. Target vehicles do not have the opportunity to move laterally out of the way of merging vehicles, and vehicles on both lanes could struggle to reduce their velocity without affecting their successors.

3.1.2 Single-to-Single Merge with slip-road

Many S2S merges are performed with an attached slip-road, as seen in Figure 3.2.

The slip-road gives merging vehicles time to travel parallel to the target lane before merging. This makes the merge easier for both MVs and TVs as MVs don't slow down in front of TVs in order to make the turn into the TL. The effectiveness of slip-roads should change with length: the longer the slip-road, the more time MVs have to merge. This should improve the effectiveness of the merge position. We can vary the length of the slip-road to see how the performance of the merge changes.

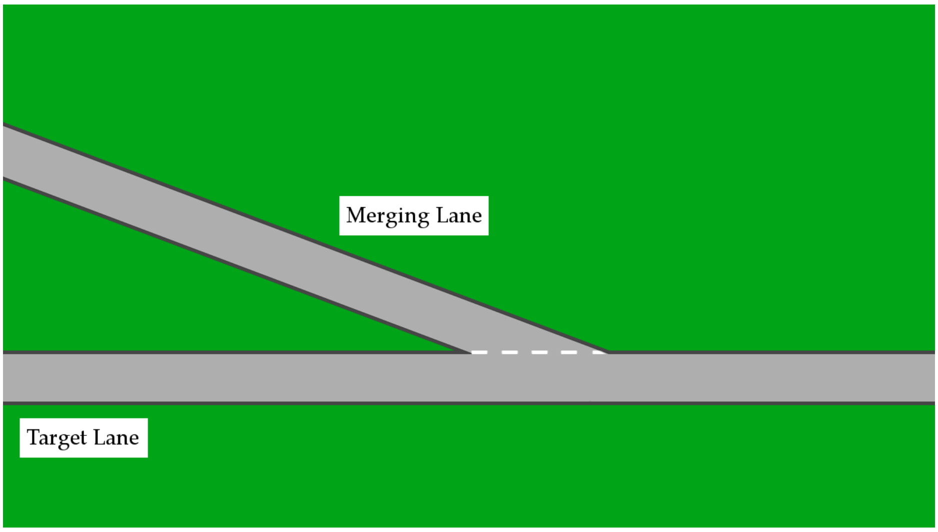

Figure 3.1: A road with a single-to-single lane merge (S2S)

3.1.3 Single-to-Double Merge

A single-to-double merge (S2D merge) describes a situation where a vehicle moves from a single lane road into a double lane road, as seen in Figure 3.3. In this situation we have two target lanes. The upper lane which directly links to the merging lane is called 'target lane 1' (TL1) and the lower lane is called 'target lane 2' (TL2). We still have only one critical position where the merging lane meets TL1.

An S2D merge provides more options for vehicles on the targets lanes at the critical position. Target vehicles now have the opportunity to move laterally to avoid merging vehicles. Two lanes also allows for more vehicles on the target lane which should give vehicles greater freedom to adjust their velocity without affecting their successors.

3.1.4 Single-to-Double Merge with slip-road

S2D merges can also take advantage of a slip-road, as seen in Figure 3.4

3.1.5 Double-to-Double Merge

A double-to-double merge (D2D merge) describes a situation where a vehicle moves from a double lane road into another double lane road,

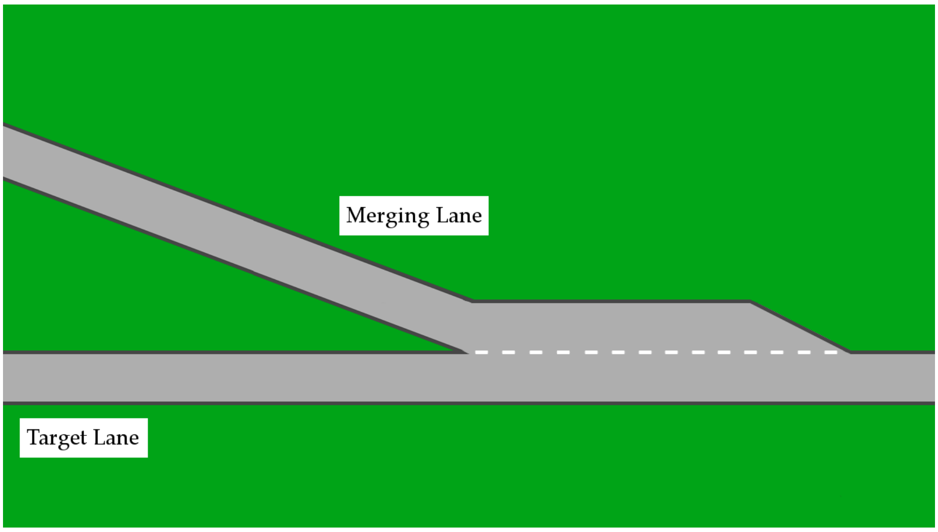

Figure 3.2: A road with a single-to-single lane merge and slip-lane (S2S)

as seen in Figure 3.5. We now have two merging lanes. The upper lane, 'merging lane 1' (ML1) merges into TL1 and the lower lane, 'merging lane 2' (ML2) merges into TL2.

With a D2D merge we now have to consider the effect of merging vehicles from ML2 driving across TL1. In addition, target vehicles can no longer laterally move out of the way of merging vehicles as they did before. Combining these factors with the wider range of options available to MVs, we can see that a D2D merge is far more complex than an S2D merge.

3.1.6 Double-to-Double Merge with slip-road

D2D merges can also take advantage of a slip-road, as seen in Figure 3.6

D2D merges with a slip-road work differently to other slip-road schemes. In this instance vehicles on ML1 and ML2 will both merge into TL1. However, ML2 merges into TL1 as vehicles on an S2D merge would. ML1 vehicles merge into TL1 as vehicles on an S2D merge would when there is a slip-road in play.

3.1.7 Lane Obstruction Merge

A lane obstruction merge is where a vehicle needs to change lanes to avoid an obstacle in their way, as seen in Figure 3.7. It is essentially an S2S merge although the vehicle will tend to move laterally to avoid the

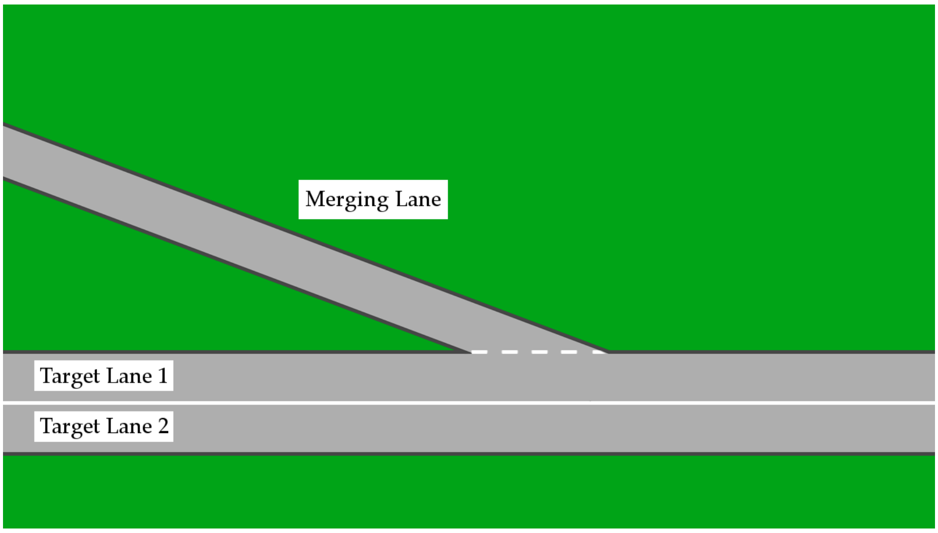

Figure 3.3: A road with a single-to-double lane merge (S2D)

obstacle. In this situation the critical position is the obstruction on the CL.

The obstacle could be a broken down vehicle or some debris on the road. Because of the unexpected nature of the obstacle it may sometimes be difficult to have a centralised approach to the problem. Although, if the obstacle was a broken down vehicle, the vehicle might be able to act as the centralised system managing approaching vehicles.

3.2 Measuring Success

In order to evaluate the effectiveness of solutions to the problems we need to define measurements of success.

Solutions to the merging problems above have to satisfy the following conditions:

1. *No collisions* This means avoiding collisions at the critical position between merging vehicles and target vehicles, as well as avoiding collisions between vehicles on the same lane.

2. *Minimise delays to both lanes* Vehicles should not suffer large delays to travel time due to the merge. *This means measuring both average*

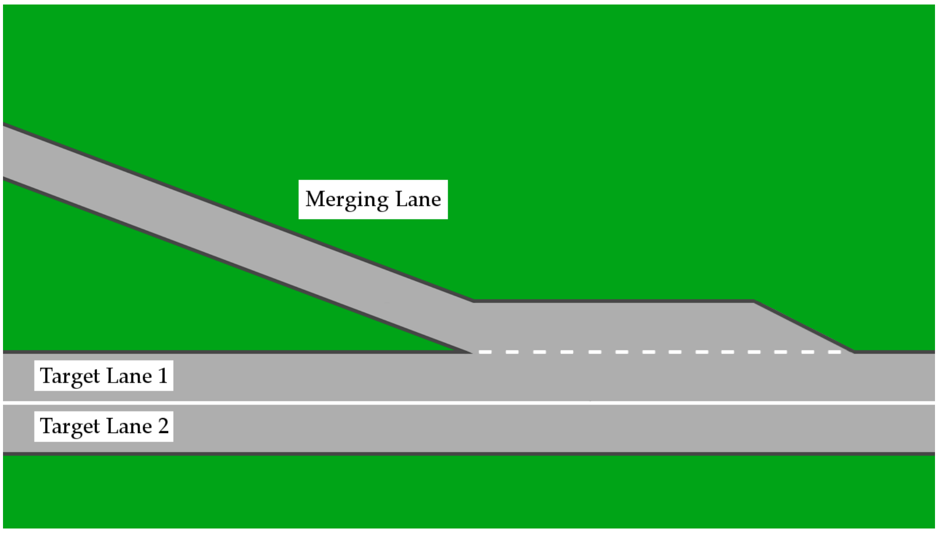

Figure 3.4: A road with a single-to-double lane merge and slip-lane (S2D)

delay and maximum delay. We do not want vehicles on one lane starving (not moving) for the benefit of vehicles on the other lane.

3. *Maximise throughput* By minimising delays and velocity loss we aim to maximise the throughput of the critical position.

4. *Minimise changes in velocity* *Though not necessary, we should aim to minimises changes in vehicle velocity, for both passenger comfort and vehicle efficiency.*

We need to measure how well solutions meet these conditions.

3.2.1 Collisions

Preventing collisions is a basic safety requirement for any autonomous vehicle system. We can measure this by comparing the positions of vehicles in the system, and ensuring that there is no overlap.

We should also consider measuring near misses. We can define a minimum spacing between vehicles, perhaps equal to the minimum braking distance of the vehicle plus an additional comfort distance. This would mimic the IDM model [16].

Any collisions that do happen should be reported immediately. The system should automatically be considered a failure.

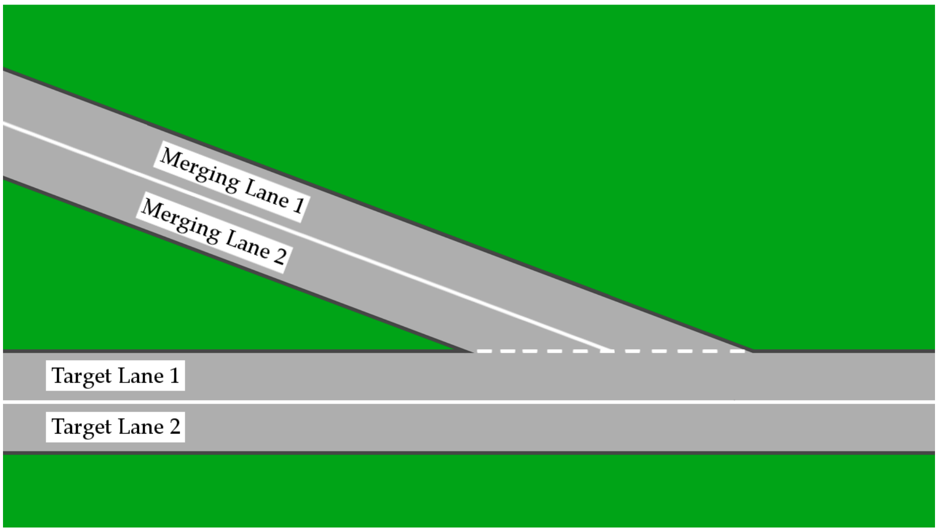

Figure 3.5: A road with a double-to-double lane merge (D2D)

3.2.2 Delay

Delay measures the effect that the critical position had on the overall journey of the vehicle. It is the primary metric considered in Dresner et al.'s 2004 paper [19] on AIM. We will measure delay in a similar manner, calculating both average delay and maximum delay.

Dresner et al. provide the following equation for measuring average delay.

$$\frac{1}{|C|} \sum_{v_i \in C} \big(t(i) - t_0(i)\big) \tag{3.1}$$

C is the set of vehicles that pass through a critical position within a set time frame. Assuming no other vehicles on the road, a vehicle v_i would complete it's trip in time $t_0(i)$, otherwise v_i would complete it's trip in time $t(i)$. We can represent this trip for vehicles in the simulator as the time difference between the vehicle spawning in and the vehicle being removed from the simulator.

Dresner et al. also provide the following equation for measuring maximum worst case delay:

$$\max_{v_i \in C} \big(t(i) - t_0(i)\big) \tag{3.2}$$

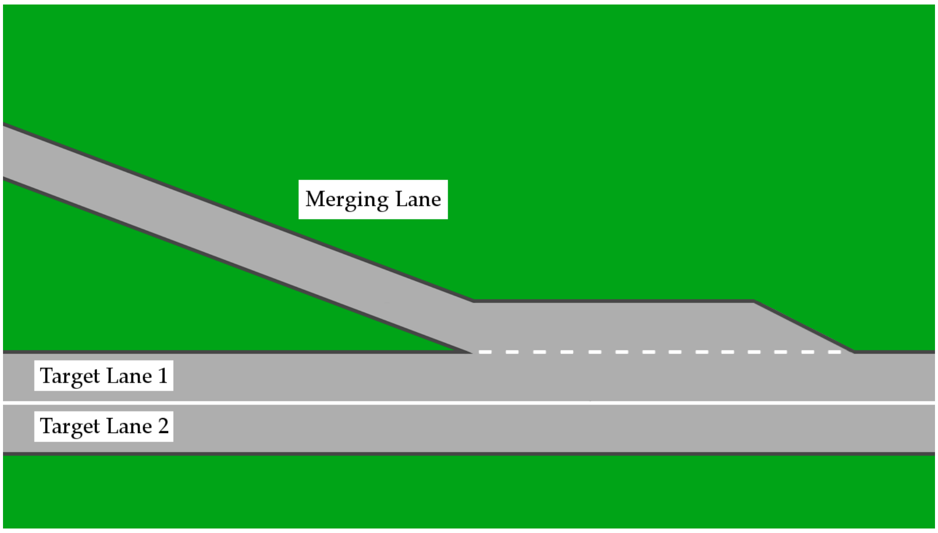

Figure 3.6: A road with a single-to-double lane merge and slip-lane (S2D)

Measuring maximum delay (and minimising it) is important, as we do not want to have a solution where some vehicles have extremely large delay times and others have very low delay times. This should help avoid a 'starvation' situation where some vehicles never get to complete their trips.

3.2.3 Throughput

By minimising delay we should also maximise throughput; the two are closely related. However we should also collect direct metrics.

$$\text{Vehicle throughput} = \frac{|C|}{t} \tag{3.3}$$

Here t is the time it took for all of the vehicles in C to pass through the critical position. If we want to measure the rate at which merging vehicles and target vehicles pass through the intersection separately we can change the definition of C to reflect that.

3.2.4 Velocity Changes

We want to reduce velocity changes as much as possible, aiming especially to eliminate rapid changes. Ideally autonomous vehicles should have very smooth

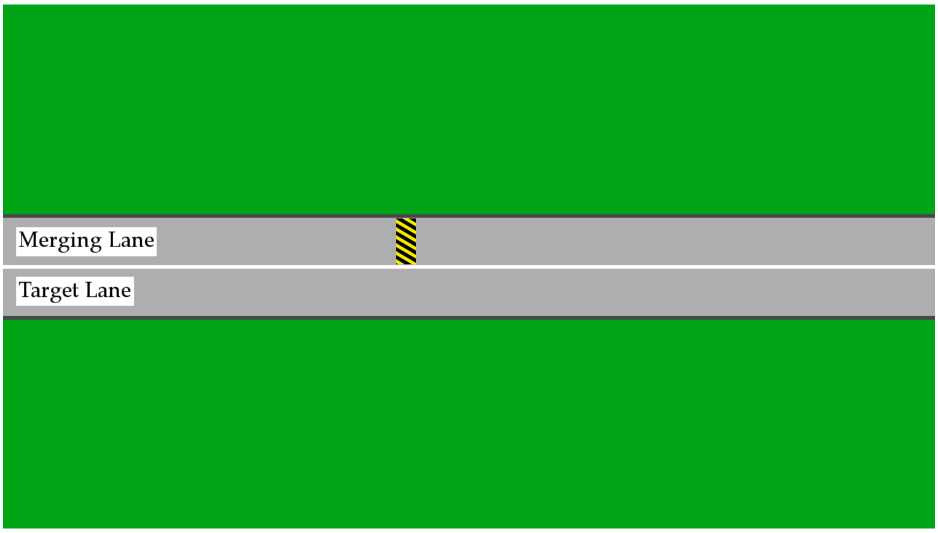

Figure 3.7: A road with a lane obstruction

acceleration and braking profiles. This both increases passenger comfort and improves fuel efficiency.

TODO: Ask Lilian for ideas on best measurements.

Current thoughts on measurement:

Maximum decleration: How to measure? Sample each second to see changes? More frequently than that? How large a sample is too big or too small?

Similar questions for maximum acceleration.

Should we also measure net velocity change over the whole critical position?

4 Design

4.1 S2S

The S2S merge is the simplest of those described in Chapter 3. Any merging system must perform well in the S2S merge before being developed further to tackle more complex merge problems.

4.1.1 User controls

Users of the simulator should be able to adjust simulation features in order to test how different factors affect the performance of the system.

- *Traffic Level* Dictates the rate at which vehicles spawn. The higher the traffic level, the greater the spawn rate of each lane.
- *Target lane speed limit* The maximum speed that vehicles on the target lane can travel.
- *Merge lane speed limit* The maximum speed that vehicles on the merge lane can travel.
- *Target lane lead in distance* The distance between the target vehicle spawn point and the point at which the target lane and merge lane end.
- *Target lane lead out distance* The distance between where a target vehicle leaves the map and the end of the area where the target lane and merge lane meet.
- *Merge lane lead in distance* The distance between where a merge vehicle spawns and where the target lane meets the merge lane.
- *Merging Angle* The interior angle θ at the point where the merging lane meets the target lane.

Figure 4.1 illustrates some of the user controllable parameters.

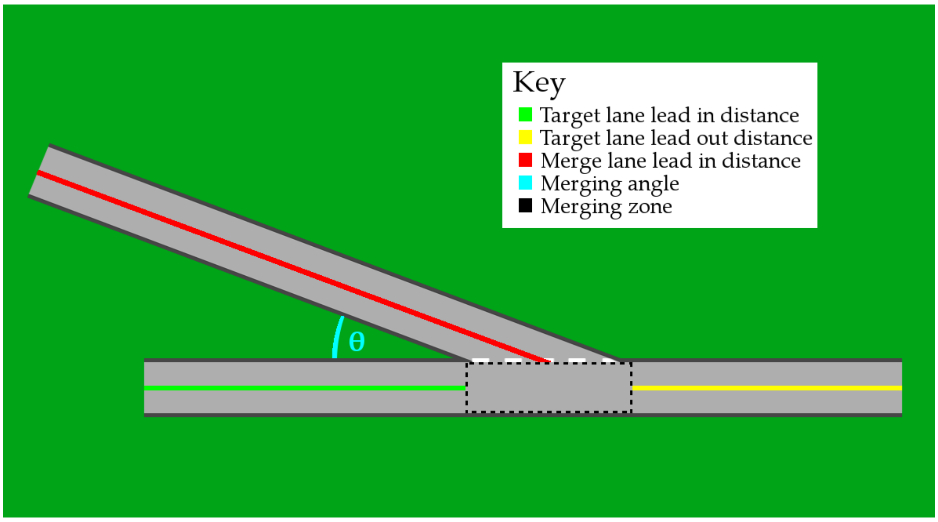

Figure 4.1: An S2S diagram marked with some of the user controllable parameters.

4.1.2 Map

In order to build the map using the user's parameters we will need to calculate the relative positions of the lane entrances and exits as well as the locations of the data collection lines and spawn points.

To start with, we need to calculate the dimensions of the merging zone. The height of the merging zone will be the same as lane width of the target lane. The length can be calculated using the right-angled triangle in Figure 4.2. Using this triangle and some trigonometry we can calculate the length of the merge zone (h in Fig. 4.2) using equation 4.1.

$$h = o/\sin(\theta) \tag{4.1}$$

We also need to know whether the horizontal width of the merge lane on the map, or it's 'base width' is longer than the target lane's lead in distance, plus the merge zone length. This will determine the width of the overall map, as if the merge lane's base length is longer then the target lane will not start with co-ordinate $x = 0$ as it would if the target lane determined the width of the map.

Firstly we need to calculate the X and Y adjustments at the merge lane entrance. Because the vehicles drive in the centre of the lane and the merge lead in distance is defined by the middle line of the lane we still

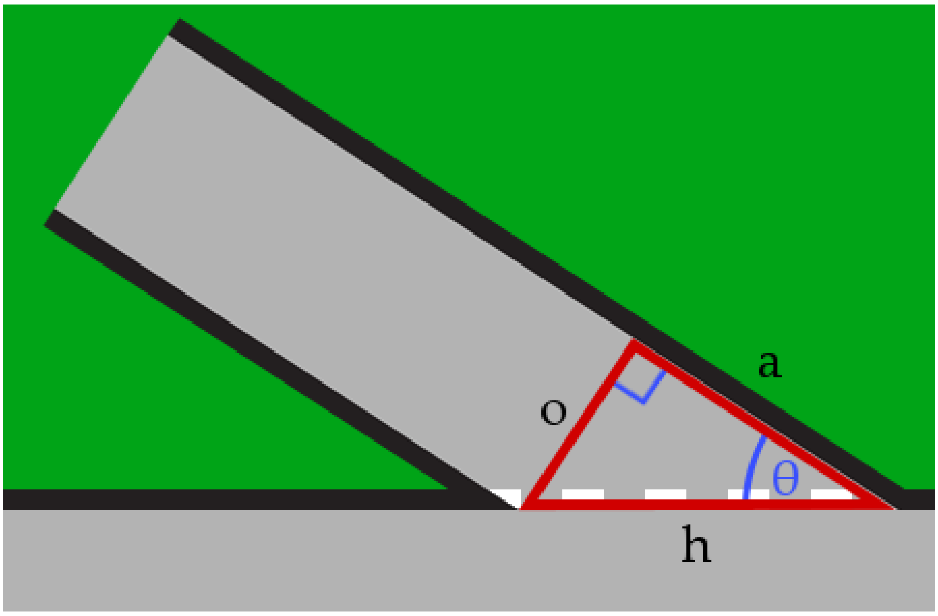

Figure 4.2: A right-angled triangle used to calculate the size of the merging zone.

need to calculate how far the lane extends in the x and y directions due to it's width. To do this we can use the right-angled triangles shown in Figure 4.3

These triangles have the same dimensions and have an interior angle of 90 - θ due to the 'alternate angle' or 'z-angle' rule. Each triangle has a hypotenuse with a length equal to half the width of the lane.

The x-adjustment for the merge entrance is the length of the adjacent side of one of the lower triangle and the y-adjustment for the merge entrance is the length of the opposite side of the upper triangle (though both triangles do have the same dimensions). We can use equation 4.2 to calculate the X-adjustment and equation 4.3 to calculate the Y-adjustment.

$$a = h\cos(90 - \theta) \tag{4.2}$$

$$o = h\sin(90 - \theta) \tag{4.3}$$

To calculate the 'base width' of the merge lane we will also need to calculate the adjacent side of the triangle in Figure 4.4. In this triangle the

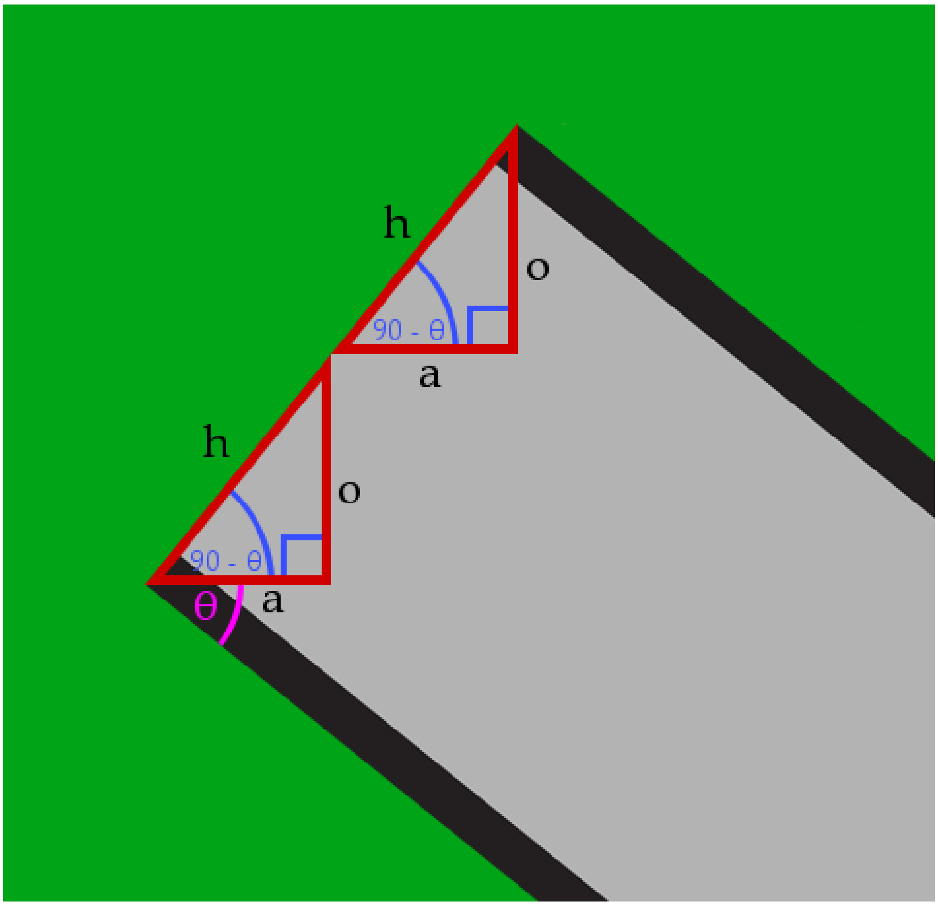

Figure 4.3: Two right-angled triangles used to calculate the x and y adjustments for the merge entrance.

hypotenuse has a length equal to the merge lead in distance. Therefore, we can use equation **??** to calculate the length of the adjacent side. After obtaining the length of this side we simply add the merge entrance X-adjustment and half the length of the merge zone to find the merge base width.

$$a = h\cos(\theta) \tag{4.4}$$

We also need to find the point at which the merging lane's centre line crosses the target lane's centre line in the merge zone. We know the Y-coordinate for this point as it will be the same as the Y-coordinate of

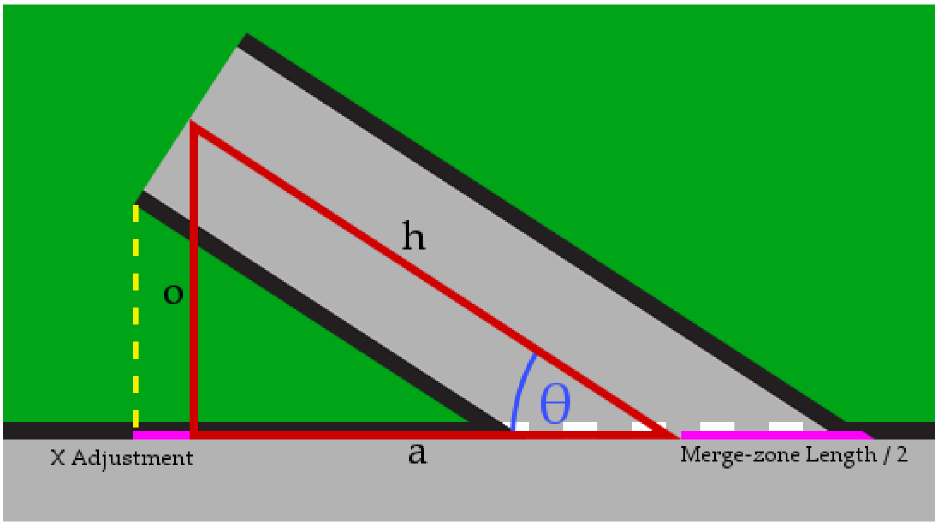

Figure 4.4: A right-angled triangle used to help calculate the base width of the merge lane, along with the X-adjustment and merge-zone length.

the target lane centre line. We also know the X-coordinate of the point at which the merge lane's centre line meets the target lane. We can use these two co-ordinates to create the triangle shown in Figure 4.5. We can then use equation 4.5 to find the X-adjustment from the merge zone centre to the point where the two centre lines cross.

$$a = o / \tan(\theta) \tag{4.5}$$

4.1.3 Intersection Management System

4.1.4 Decentralised Communication System

4.1.5 Measuring Success

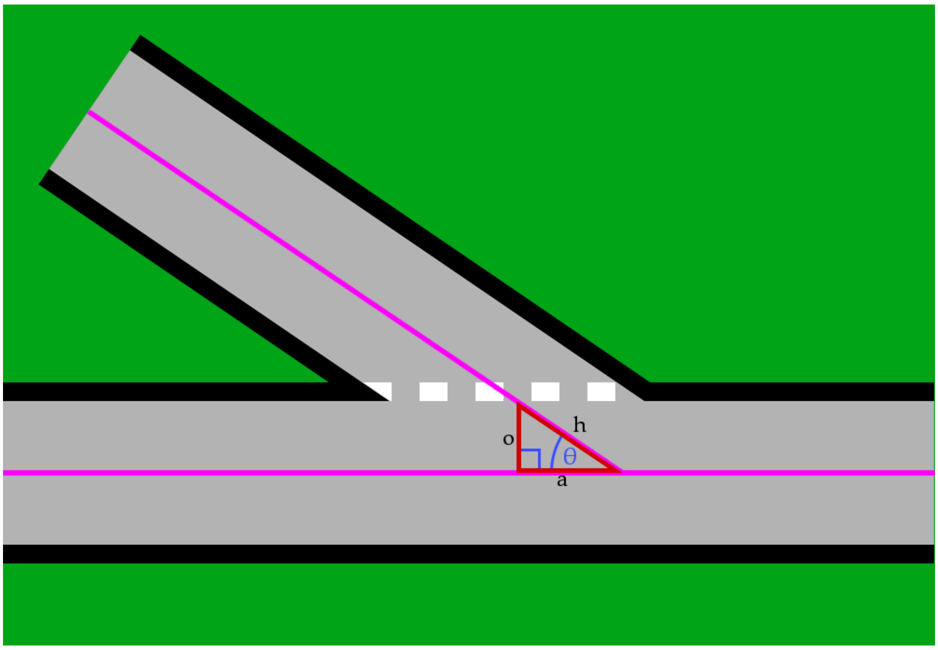

Figure 4.5: A right-angled triangle used to calculate where the two centre lines meet. The centre lines are indicated in pink.

5 Implementation

5.1 Generalising the Codebase

The simulators created for this project were created partly using the AIM simulator codebase [22]. We wanted to expand the simulator so that it could run AV simulations in situations other than 4-way intersections. To do this we needed to generalise the AIM codebase so that new developers could add simulators with very little effort, however, we also needed to ensure that these changes would not affect the functionality of the AIM simulator. These changes to the codebase were done in conjunction with Rebecca Milligan, who is also working on AV simulations in her car park management project.

To generalise the codebase we refactored key classes into separate general and AIM specific classes. The general classes can be expanded to create other simulator specific classes, whilst the AIM specific classes maintain the functionality of the original simulator. This helps to reduce code duplication when developing new simulators.

All class diagrams were created using IntelliJ IDEA 15.0.3 internal diagram tool. Figure 5.1 provides a key for understanding these diagrams.

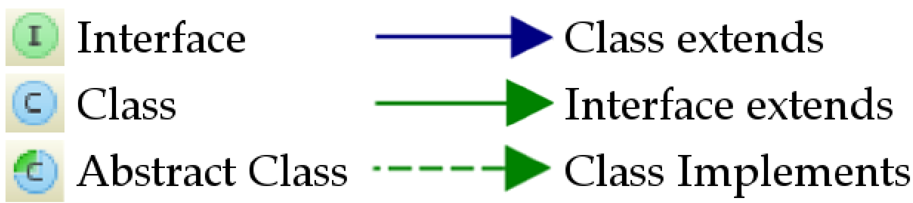

Figure 5.1: Key for the class diagrams in this report.

5.1.1 aim4.driver

aim4.driver controls how a vehicle behaves on the map. In the original simulator the drivers were built to deal with 4-way intersections, with general functionality tied into the same class. You can see how this was done in Figures 5.2 and 5.3.

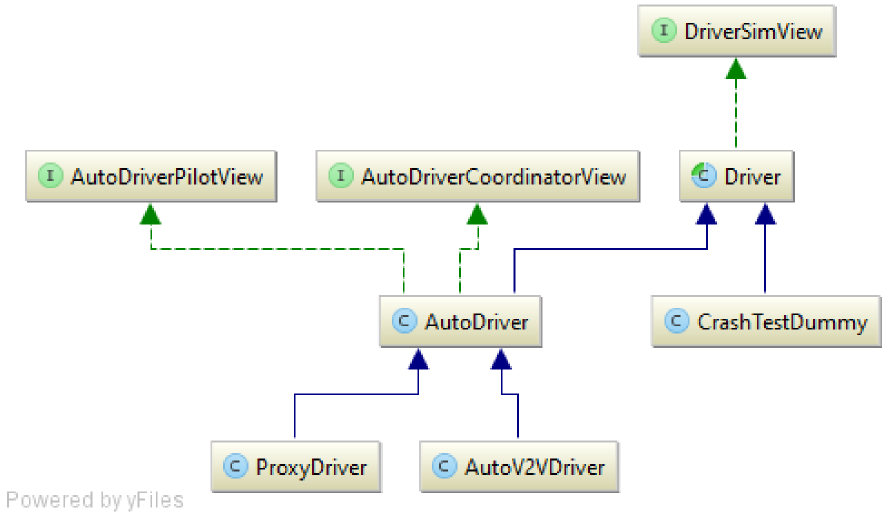

Figure 5.2: The original class structure for *Driver*.

The first major change was renaming *DriverSimView*, *AutoDriverPilotView*, and *AutoDriverCoordinatorView* to end in *Model* instead of *View*. These interfaces are used to limit the methods that other classes can access in Driver and AutoDriver, thus changing their 'view' of that class. We felt that *View* could cause confusion with the GUI elements of the simulator; we instead chose to refer to these interfaces as *Models*, because the accessors are effectively given a model of Driver and AutoDriver (beyond which they care very little) that they can use to access methods.

The next change was separating out all of the AIM specific code into its own classes and interfaces. You can see how this was done in Figure 5.3 with *AIMDriverSimModel* and *AIMDriver*. The merge specific code found in *MergeDriverSimModel*, *MergeDriver* and *MergeAutoDriver* is structured in a very similar manner to its AIM counterpart, taking advantage of the generalised code.

As a consequence of breaking out the code like this, a number of additional changes had to be made. Driver was changed into an interface and a new class *BasicDriver*. *Driver* is simply used as an interface for accessing Drivers in non-simulation contexts (such as *BasicVehicle*). *BasicDriver*

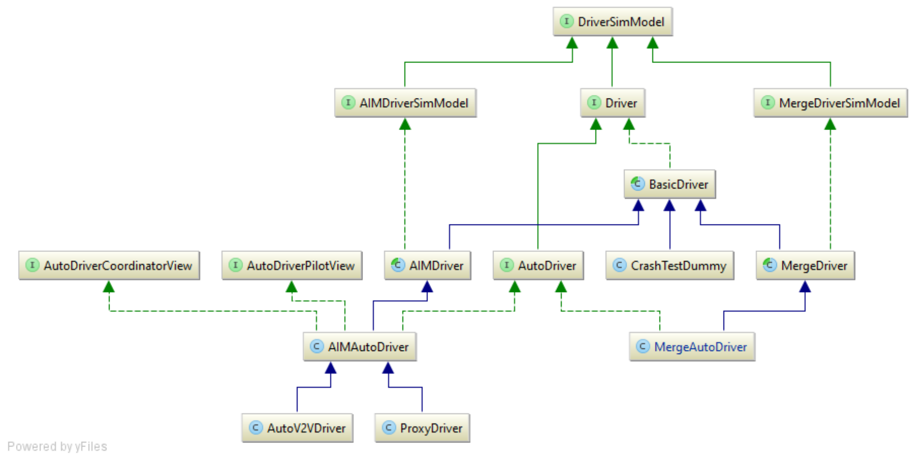

Figure 5.3: The new class structure for *Driver*.

contains the generalised functionality all *Driver* objects should need, with AIM specific activities moved to *AIMDriver*. Extending from *Driver* is the *AutoDriver* interface, which adds no new methods but is instead used to categorise autonomous drivers. *AIMAutoDriver* contains almost exactly the same code as the original *AutoDriver* class.

5.1.2 aim4.gui

aim4.gui controls the GUI for the simulator. We had to adjust this to allow for non-AIM simulations to be run. We chose to use tabs to allow users to switch between simulators (these are greyed out when a simulation is running). To make adding new tabs and simulation screens easier we had to refactor *Viewer* into smaller, separate components. You can see the structural changes in Figures 5.4 and 5.5.

In the original simulator *Viewer* displays the simulator set-up controls, *SimSetupPanel*, and the simulation viewer *Canvas* inside *mainPanel*. *mainPanel* is a *JPanel* with a *CardLayout* allowing the panel to switch between displaying the set-up controls and the viewer. In the new simulator we replaced *mainPanel* with *tabbedPane*, a *JTabbedPane* object that allows users to switch between the different simulators using tabs. Each tab displays a *SimViewer*, which behaves in a similar way to *mainPanel* allowing users to switch between the set-up screen and the simulation screen using *CardLayout*. Each simulator will have their own SimViewer type, as shown in

Figure 5.6.

We didn't want to force new simulators to use a full representation of vehicles on screen, as *Canvas* does. To avoid this we created a new interface *SimScreen* which *SimViewer* uses to describe it's viewer card. Any class implementing *SimScreen* can be used as the viewer for a simulation. Figure 5.7 shows how *MergeStatScreen* and *Canvas* using *SimScreen*.

We also generalised the *SimSetupPanel* class to allow *SimViewer* to display non-AIM set-up controls. Figure 5.8 shows the new class structure for *SimSetupPanel*.

We also made a small adjustment to the behaviour of the reset option in the menu. Now the simulator must be paused in order for the reset button to be active. We did this because resetting the simulator without pausing was creating *NullPointerException*s.

5.1.3 aim4.map

aim4.map is used to describe the environment vehicles are required to navigate. They also spawn vehicles that then drive through the map. Figures 5.9 and 5.10 show the original and new class structure for *aim4.map*.

The changes made to *aim4.map* were relatively straight-forward. The AIM specific features in *BasicMap* were extracted out in *BasicIntersectionMap* and *GridMap* was renamed to *GridIntersectionMap* and now inherits from the new interface. This allows for new map types, such as *MergeMap* to implement a map type without AIM features.

SpawnPoint was also broken out into general and AIM specific features. This had to be done because *SpawnPoint* used to create *SpawnSpec* objects with *destination* fields. *destination* is an AIM specific field relating to the intersection exit a vehicle plans to reach. By extracting this out new map types can spawn vehicles with *SpawnSpec* instances specific to their map type.

5.1.4 aim4.sim

aim4.sim contains the code responsible for constructing and running simulations. The original code was very focussed on AIM simulations and so we had to break the interfaces to allow for different types of simulators.

Simulator is an interface that new simulators need to implement. We decided to extract out some of the AIM specific features into *AIMSimulator*. We also added an override to *getMap()*, forcing AIM simulators to use *BasicIntersectionMap* maps. The class structure changes can be seen in Figures 5.11 and 5.12.

SimSetup was also modified to separate AIM specific set-up options and simulator creation code from other simulators. Figures 5.13 and 5.14 show how these classes were altered.

5.1.5 aim4.vehicle

aim4.vehicle controls the different vehicles used during simulations. Vehicles are used by both *Driver* and *Simulator* instances. To allow them to do that the original simulator code used *View* interfaces similar to those in 5.1.1. Figure 5.15 shows how these interfaces link together. Extracting AIM behaviour was quite difficult because of how interconnected these interfaces were. The solution we came up with was to create AIM specific interfaces and link them together in a similar manner, inheriting from the generic ones if possible. Figure 5.16 shows how the new structure links together.

The first change made to *aim4.vehicle* was to rename all of the files ending in *View* to end in *Model* instead. This matches the changes made to *aim4.driver*.

AIMVehicleSimModel and *AIMAutoVehicleDriverModel* are at the top of the AIM interface tree. They both extend their generic counterparts. *AIMAutoVehicleSimModel* extends these two interfaces along with *AutoVehicleSimModel*. This matches up to the original inheritance structure. Any future vehicles will need to create their own version of these interfaces, as seen in *MergeVehicleSimModel*, *MergeAutoVehicleDriverModel* and *MergeAutoVehicleSimModel*.

In terms of classes we made *BasicAutoVehicle* abstract and extracted out AIM specific behaviour to *AIMBasicAutoVehicle*. *BasicAutoVehicle* had to be abstract because we wanted to force *getDriver()* to be overridden in subclasses to retrieve the simulator specific *AutoDriver* for that vehicle (for example *AIMAutoDriver* in AIM simulators).

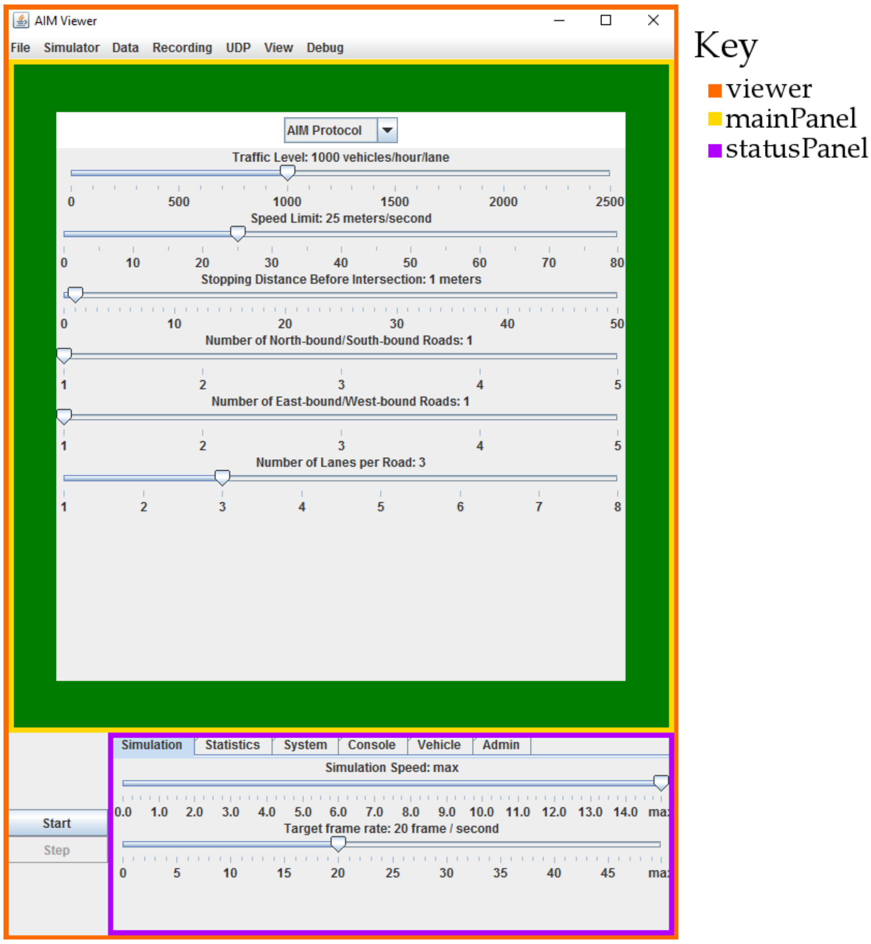

Figure 5.4: Panel layout in the original simulator.

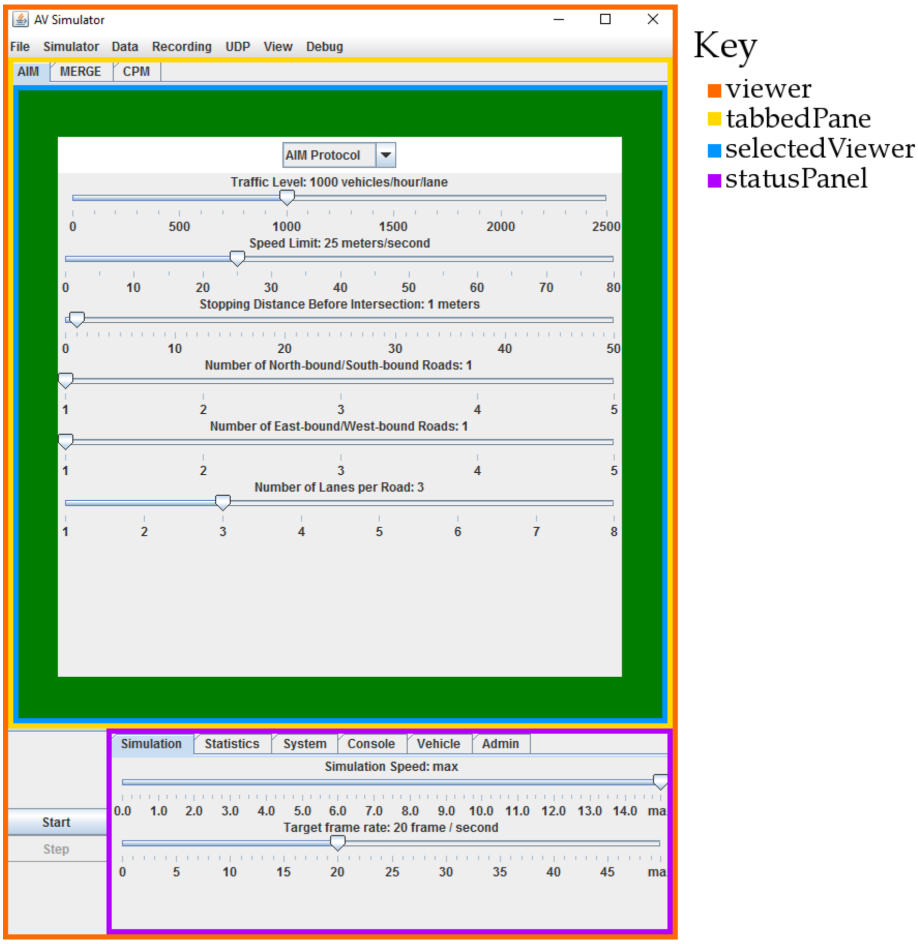

Figure 5.5: Panel layout in the new simulator.

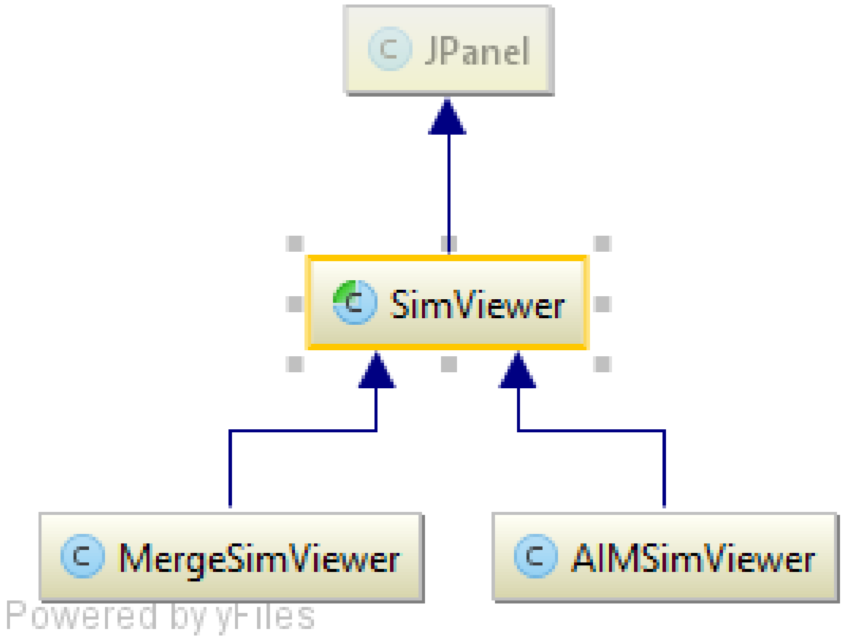

Figure 5.6: The class diagram for *SimViewer*.

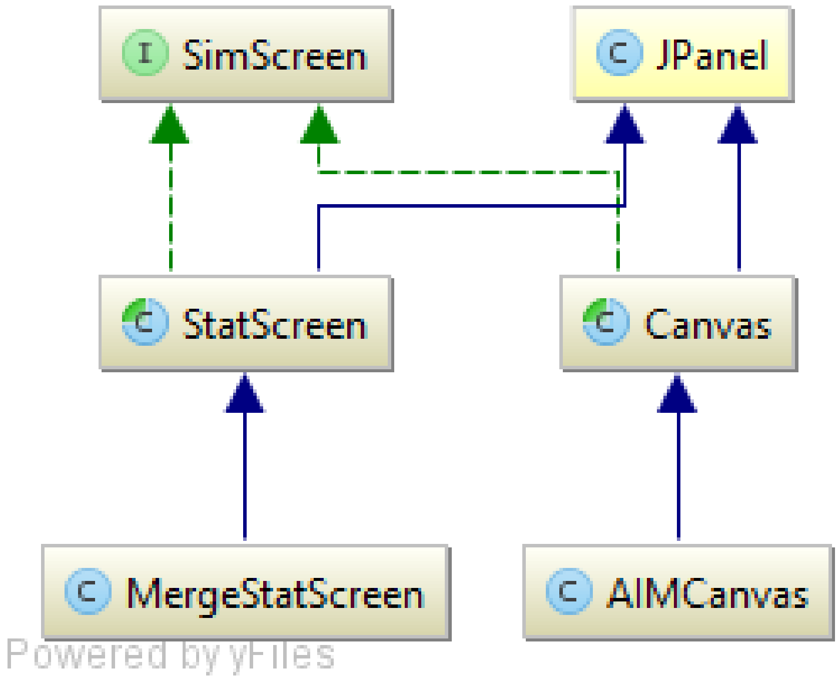

Figure 5.7: The class diagram for *SimScreen*.

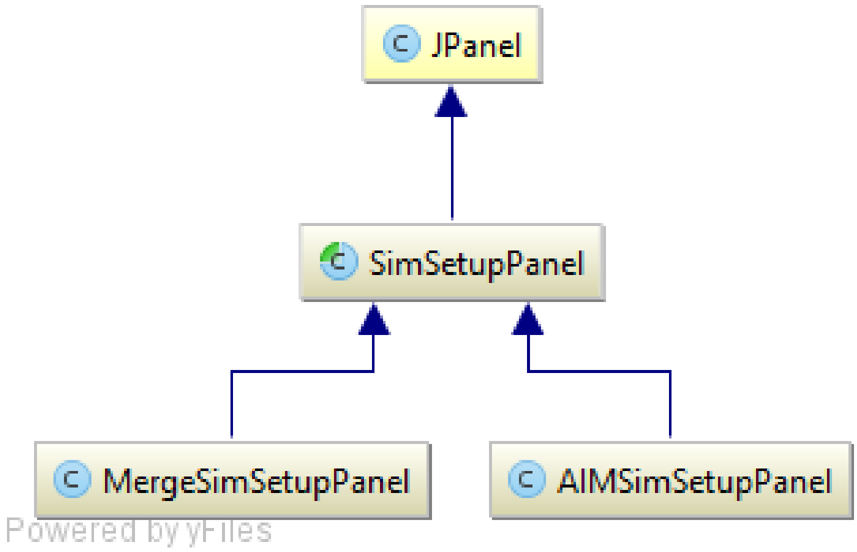

Figure 5.8: The class diagram for *SimSetupPanel*.

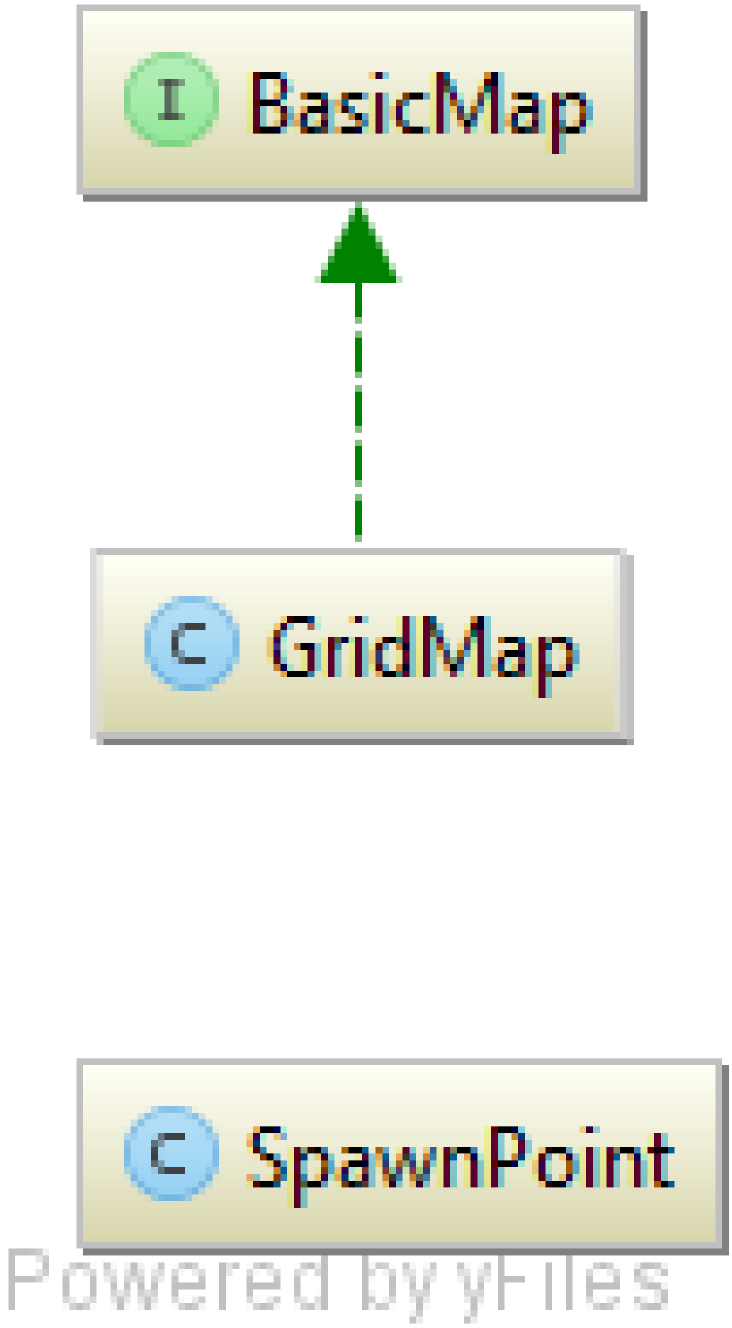

Figure 5.9: The original class structure for *BasicMap* and *SpawnPoint*.

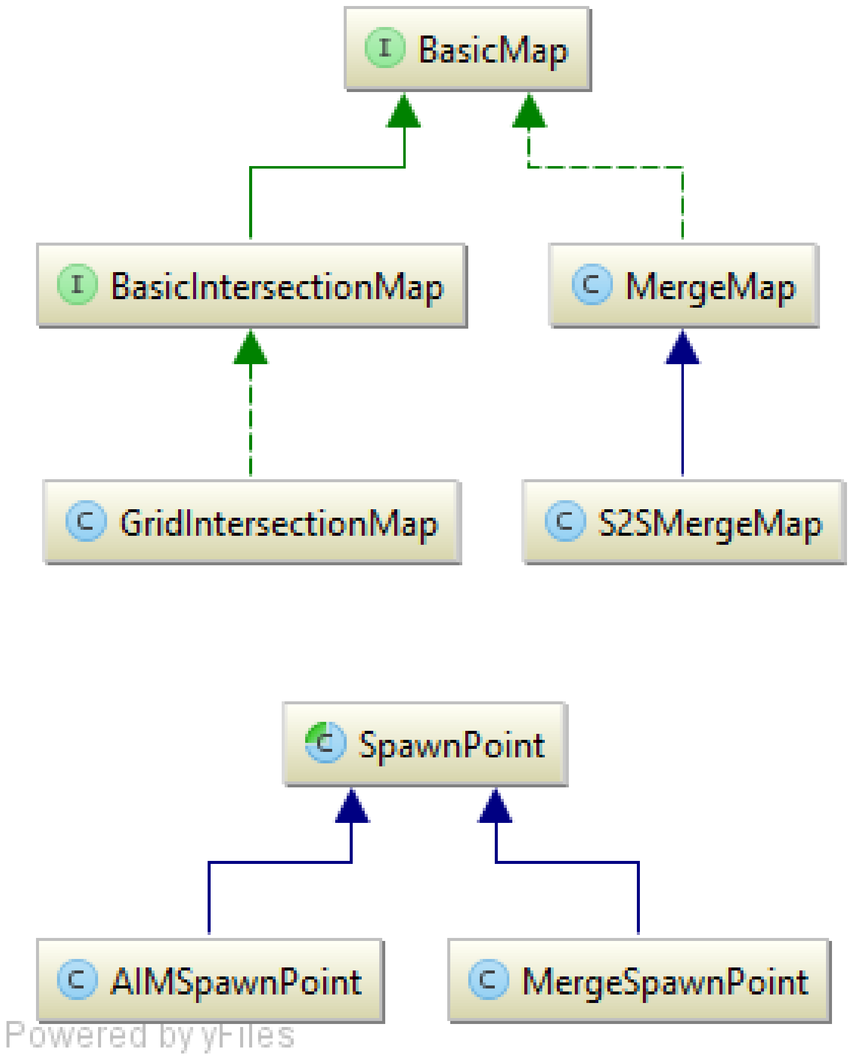

Figure 5.10: The new class structure for *BasicMap* and *SpawnPoint*.

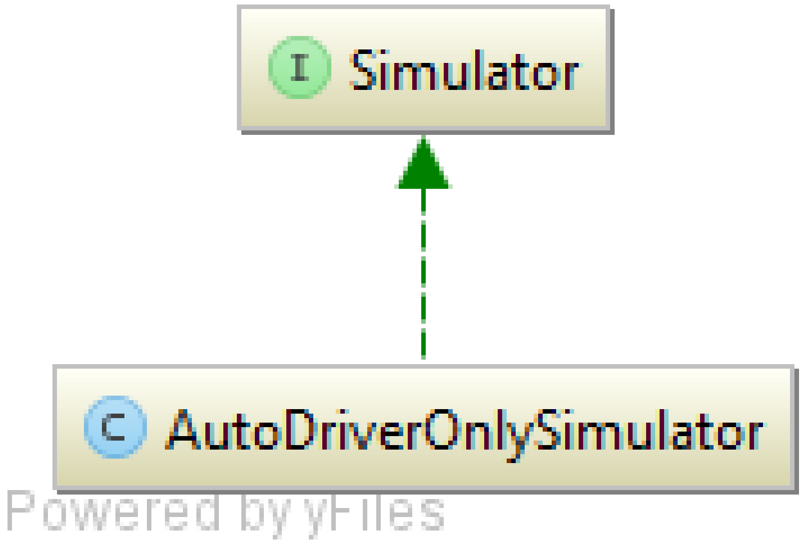

Figure 5.11: The original class structure for *Simulator*.

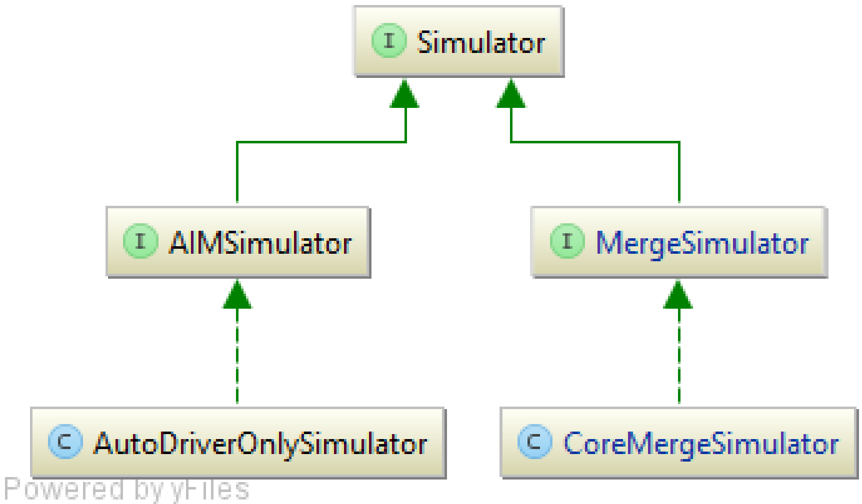

Figure 5.12: The new class structure for *Simulator*.

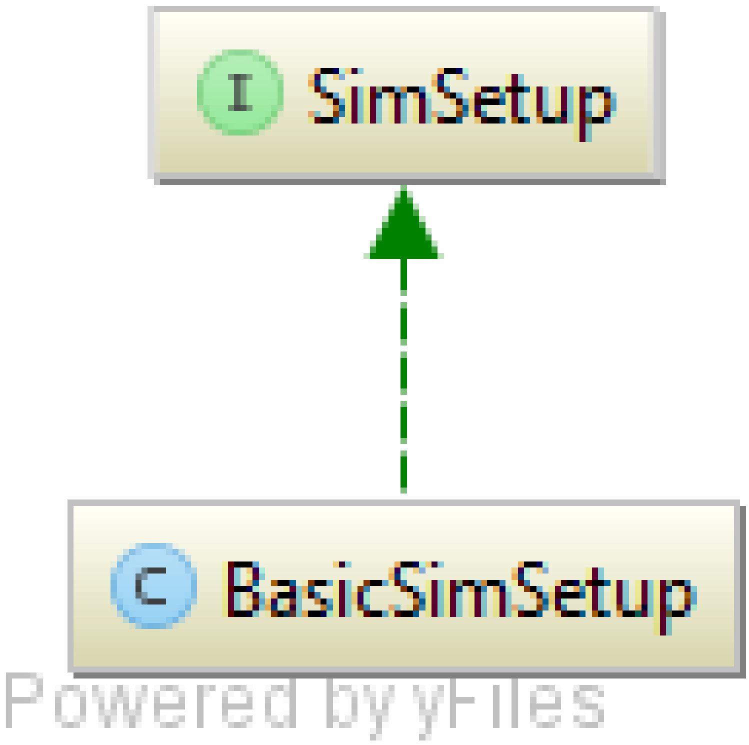

Figure 5.13: The original class structure for *SimSetup*.

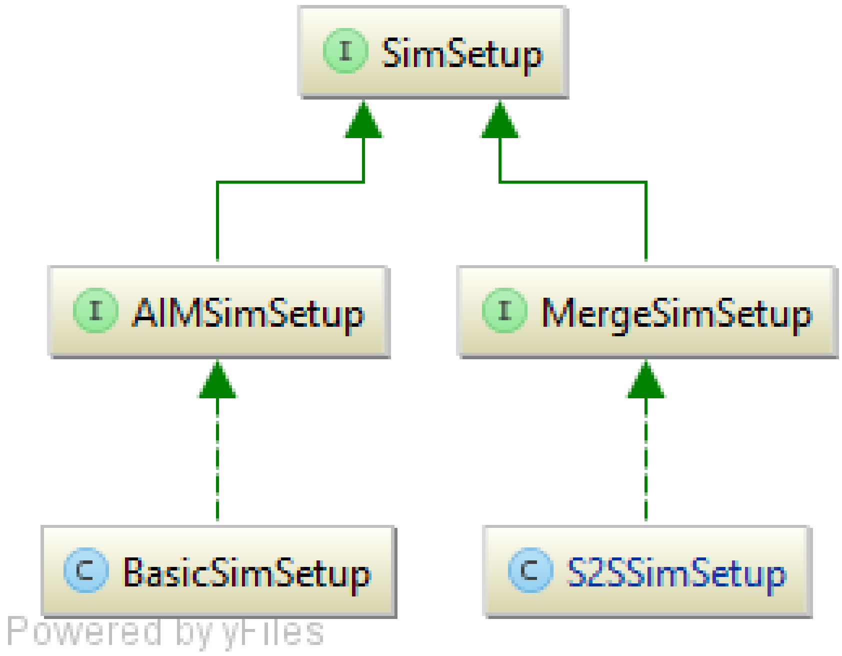

Figure 5.14: The new class structure for *SimSetup*.

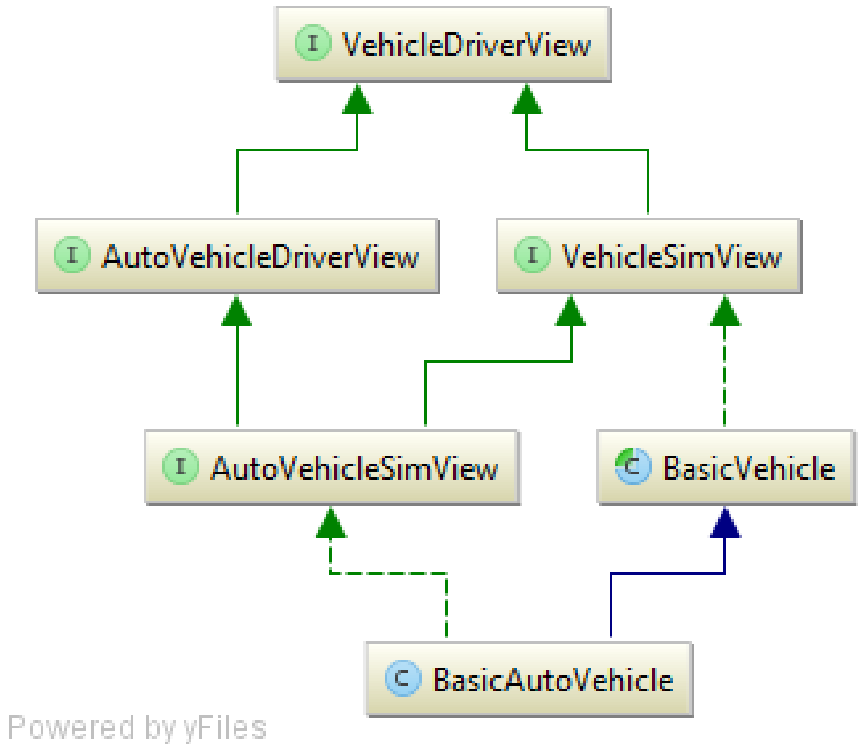

Figure 5.15: The original class structure for *aim4.vehicle*.

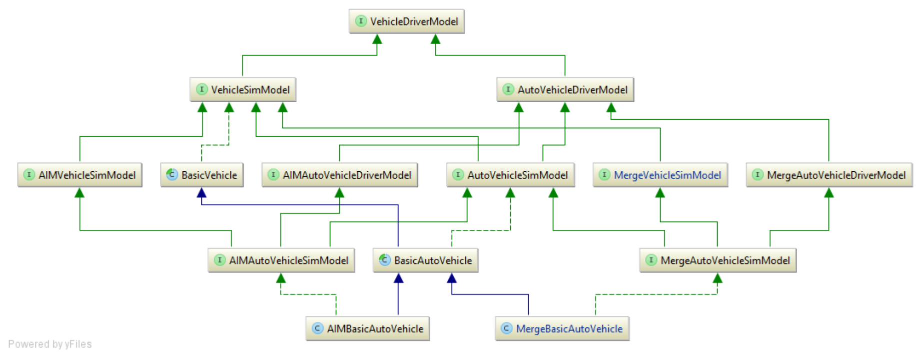

Figure 5.16: The new class structure for *aim4.vehicle*.

6 Results

Lilian Notes:

1. It will be interesting to have a graph with |C| on the x-axis and throughput on y-axis. Similarly, a 3D graph where |Ctl| on the x-axis, |Cml| on the y-axis, and throughput on z-axis.

Callum Notes:

1. As a "things I'd do differently" or "changes to make" it would be cool to move to a Spring implementation. It might make it easier to add different simulator types.

Bibliography

[1] (2016, October) General rules, techniques and advice for all drivers and riders (103 to 158). Website. Her Majesty's Government. Rule 126. [Online]. Available: https://www.gov.uk/guidance/the-highway-code/general-rules-techniques-and-advice-for-all-drivers-and-riders-103-to-158

[2] P. Gipps, "A model for the structure of lane-changing decisions," *Transportation Research Part B: Methodological*, vol. 20, no. 5, pp. 403–414, October 1986.

[3] I. Asimov, "Sally," *Fantastic*, vol. 2, no. 3, May 1953.

[4] G. A. Larson, "Knight rider," Television, Sep. 1982.

[5] (2016, Dec.) Waymo. Alphabet. [Online]. Available: https://waymo.com/ontheroad/

[6] (2016, August) Tesla car drives owner to hospital after he suffers pulmonary embolism. Website. BBC. [Online]. Available: http://www.bbc.co.uk/newsbeat/article/37009696/tesla-car-drives-owner-to-hospital-after-he-suffers-pulmonary-embolism

[7] D. Lee. (2016, June) Us opens investigation into tesla after fatal crash. Website. BBC. [Online]. Available: http://www.bbc.co.uk/news/technology-36680043

[8] A. C. Mersky and C. Samaras, "Fuel economy testing of autonomous vehicles," *Transportation Research Part C: Emerging Technologies*, vol. 65, pp. 31–48, apr 2016. [Online]. Available: http://dx.doi.org/10.1016/j.trc.2016.01.001

[9] T. U. Nations, "The paris agreement," Oct. 2016. [Online]. Available: http://unfccc.int/files/essential_background/convention/application/pdf/english_paris_agreement.pdf

[10] T. L. J. B. David Schrank, Bill Eisele. (2015, August) 2015 urban mobility scorecard. Website. Texas A&M Transportation Institute & INRIX. [Online]. Available: http://d2dtl5nnlpfror.cloudfront.net/tti.tamu.edu/documents/mobility-scorecard-2015-wappx.pdf

[11] C. F. Daganzo, "The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory," *Transportation Research Part B: Methodological*, vol. 28, no. 4, pp. 269–287, aug 1994. [Online]. Available: http://dx.doi.org/10.1016/0191-2615(94)90002-7

[12] (2016, September) Tesla criticised over autopilot safety. Website. BBC. [Online]. Available: http://www.bbc.co.uk/news/technology-37372933

[13] (2016, October) Aim homepage. Website. The University of Texas. [Online]. Available: http://www.cs.utexas.edu/~aim/

[14] M. Quinlan, T.-C. Au, J. Zhu, N. Stiurca, and P. Stone, "Bringing simulation to life: A mixed reality autonomous intersection," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Institute of Electrical and Electronics Engineers (IEEE), oct 2010. [Online]. Available: http://dx.doi.org/10.1109/iros.2010.5651993

[15] P. Gipps, "A behavioural car-following model for computer simulation," *Transportation Research Part B: Methodological*, vol. 15, no. 2, pp. 105–111, apr 1981. [Online]. Available: http://dx.doi.org/10.1016/0191-2615(81)90037-0

[16] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical Review E*, vol. 62, no. 2, pp. 1805–1824, aug 2000. [Online]. Available: http://dx.doi.org/10.1103/physreve.62.1805

[17] D. H. Arne Kesting, Martin Treiber, "General lane-changing model mobil for car-following models," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1999, pp. 86–94, 2007.

[18] M. Kamali, L. A. Dennis, O. McAree, M. Fisher, and S. M. Veres, "Formal verification of autonomous vehicle platooning," *CoRR*, vol. abs/1602.01718, 2016. [Online]. Available: http://arxiv.org/abs/1602.01718

[19] P. S. Kurt Dresner, “Multiagent traffic management: A reservation-based intersection control mechanism,” in *Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 04)*, vol. 2, Jul. 2004, pp. 530–537.

[20] M. Atagoziyev, K. W. Schmidt, and E. G. Schmidt, “Lane change scheduling for autonomous vehicles,” *IFAC-PapersOnLine*, vol. 49, no. 3, pp. 61–66, 2016. [Online]. Available: http://dx.doi.org/10.1016/j.ifacol.2016.07.011

[21] M. VanMiddlesworth, K. Dresner, and P. Stone, “Replacing the stop sign: Unmanaged intersection control for autonomous vehicles,” in *AAMAS Workshop on Agents in Traffic and Transportation*, Estoril, Portugal, May 2008, pp. 94–101. [Online]. Available: "http://www.cs.utexas.edu/users/ai-lab/?ATT08-vanmiddlesworth"

[22] P. Stone, G. Sharon, M. Albert, and J. Hanna. (2017) Aim project homepage. University of Texas. [Online]. Available: http://www.cs.utexas.edu/~aim/