Towards evaluating the benefits of inter-vehicle coordination

Niall O'Hara, Marco Slot, Julien Monteil, Vinny Cahill, Mélanie Bouroche
Distributed Systems Group, School of Computer Science and Statistics,
Trinity College Dublin, Ireland

Abstract— **While vehicle automation has the potential to significantly improve safety and traffic efficiency, the full potential will only be realised when vehicles start exploiting wireless communication to cooperate with each other and coordinate their interactions in advance. Ensuring that vehicles coordinate safely while improving efficiency is, however, a very challenging problem as it depends on (i) the characteristics of individual vehicles (vehicle physics, sensors), (ii) unreliable wireless communication, and (iii) driving behaviour at a microscopic level, and their compounded effects at scale. The presence of non-communicative, non-automated vehicles must also be considered.**

Designing and evaluating coordination protocols requires a scalable simulation framework that is accurate both microscopically (to assess safety) and macroscopically (to evaluate efficiency). Standard car-following models, where position and velocity are dictated by local input stimuli, produce sometimes unrealistic behaviour when laterally changing position, and lack support for additional inputs. Furthermore, conventional environments used to model traffic flow are either too fine-grained to scale or too coarse to appropriately simulate control logic. This paper introduces RoundaSim consisting of (i) a traffic simulator using a novel approach of mixed discrete-continuous modes of time, and (ii) a framework for implementing car-following models that supports lane-changing and coordination protocols, with additional inputs from advanced sensors and wireless communications. We show how our framework can be used to implement and evaluate a car-following model with lane changes and validate that the traffic flow achieved approximates that of real-world highways. This allows our platform to be used as a baseline for evaluating the safety and efficiency of coordination protocols.

I. Introduction

Whilst drivers sometimes subconsciously coordinate their actions driving behaviour tends to be competitive, often leading to unsafe behaviour and ultimately causing congestion and delays. By communicating with each other over wireless networks, vehicles could share information and actively coordinate their behaviour with the potential to significantly improve safety and efficiency. Real-time coordination [1] and group communication [2] are required to dictate the information a car broadcasts and the actions it should take based on received messages and available sensor data. Designing coordination protocols that ensure safety while improving efficiency is challenging, since:

- vehicles are heterogeneous in terms of physical attributes, but also in terms of capabilities: what sensors and actuators they are equipped with, as well as whether they can communicate,
- sensor range is limited and wireless communication is unreliable (messages can be delayed or lost),
- the behaviour of a vehicle will depend on the driver and their ability to process and act on presented information.

As a result, unexpected traffic patterns may emerge from the logic followed by vehicles when reacting to each other's presence, hence potentially threatening safety, and making it challenging to rigorously evaluate potential efficiency improvements.

Designing and evaluating coordination protocols therefore requires a framework that includes realistic models of the physical properties of vehicles, advanced sensors and wireless communication whilst providing microscopic accuracy to assess safety. It also must support the simulation of a large number of vehicles over wide scenarios to enable the study of efficiency at a macroscopic scale. Furthermore, due to the complexity of the interactions between participants and the resultant emergent behaviour, real-time visualisation of the simulation is particularly helpful. Indeed, observing the emergent behaviour helps understand the effects of the coordination protocol under study, and their evolution over time. Finally, the ability to model vehicles' behaviour, and in particular the behaviour of heterogeneous vehicles, is essential.

In this paper, we present a traffic simulator and framework that provides the ability to accurately model vehicles' dynamics, with sensors for orientation, position, velocity, orientation and indicator detection. In addition LIDAR sensor and wireless communication (802.11p) capabilities are also provided. These can all be used as inputs to a controller, which controls the actuators of the vehicle. The simulation engine, a hybrid between a microscopic traffic simulator and a robotics simulator, where discrete events are interleaved with continuous time progress, supports the evaluation of the safety and efficiency of traffic. We show how our framework can be used to implement a controller for an automated vehicle driving on highways which utilises sensor data as input to support opportunistic lane changes coordinated through indicators.

As automated vehicles will initially mimic human drivers [3], they should show similar overall behaviour. By comparing traffic statistics in a simulated highway scenario to well-established traffic models, based on real-world observations from the Highway Capacity Manual [4], we show that our implementation accurately reproduces the expected relationship between density, speed, and flow rate. This creates a baseline for the evaluation of future coordination protocols.

The next section describes related work and section III provides an in-depth description of RoundaSim and its components. Section IV outlines the implementation of a highway car-following model with lane changes within our framework. In section V we outline our experiments to validate our implementation and present our results. Finally, section VI concludes the paper.

II. Related Work

Existing microscopic traffic simulators, such as VISSIM [5] and SUMO [6] discretize time into steps of fixed size, which is appropriate for evaluating traffic flow, but the fidelity of interactions is lost in the coarseness of the time steps. Another drawback of fixed time-step simulation is that many events in the simulation occur at the same instant in time, while this would not be the case in reality. This can lead to peculiar side-effects, especially in scenarios in which the vehicles are not expected to be synchronized in terms of their actions, sensor measurements, and communication. For this reason these simulators cannot be used to evaluate safety. MovSim [7] allows for the investigation of discrete decision making within traffic situations. It implements various car-following models and aims to model basic traffic flow situations and discrete decisions such as lane changing on a highway. It does not however provide any advanced sensor or wireless communication simulation. As such, it is requires extension to make it suitable to investigate cooperative vehicle scenarios [8]. PreScan [9], a physics-based simulator used for the development of Advanced Driver Assistance System (ADAS), provides advanced sensor simulation, realistic vehicle dynamics and visualization tools. It also supports real-time visualisation, but it does not scale well and vehicles cannot be dynamically added at runtime. Furthermore, its wireless network simulation capabilities are quite basic. In order to increase the realism of the scenario under investigation, combining multiple simulation platforms is a route often taken. Mylonas [10] and Bhakthavathsalam [11] evaluated inter-vehicle communication protocols and applications through the integration of microscopic simulation programs (VISSIM, SUMO) and discrete event communication simulation engines (OPNET, NS-2). They demonstrated that the approach was valid when developing such applications however as they focused on inter-vehicle communication, they were not concerned by car-following models.

A core part of traffic simulation is the car-following model used to control decisions of an individual vehicle [12]. The aim of a car-following model is to approximate what a vehicle is likely to do in a given situation, in terms of acceleration, lane changes, or routing. Various car-following models have been developed such as Wiedemann [13], Gipps [14] and the widely used Intelligent Driver Model (IDM) [15], which provides acceleration. It is the default car-following model for SUMO. MOBIL [16] extends acceleration models with lane changes. A car-following model is generally based on simplifying assumptions. For example, most models assume the distance to, and the speed of, a predecessor are known instantly, while human drivers can only roughly estimate these and automated vehicles are limited by the capabilities of their sensors. For the purpose of developing a car-following model for an automated vehicle, safety needs to be preserved in the presence of inaccuracy and uncertainty from measurements and transformations. The heuristics provided by a standard car-following model can still be used to achieve a 'good' baseline behaviour, although the overall behaviour may not perfectly match the expected performance of the vehicle unless inputs from advanced sensors and wireless communications are accurately modelled. The highway controller presented in section IV uses the IDM and an incentive based lane change process similar to that of MOBIL.

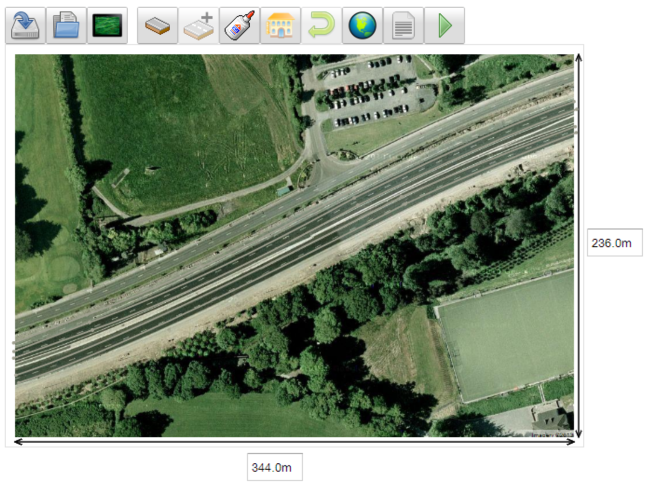

Fig. 1. RoundaSim GUI - Section of the M50 highway in Dublin

III. RoundaSim

This section first outlines RoundaSim's novel simulation approach and describes the concept of tracks, which enable RoundaSim's scalability. We then detail how the capabilities of vehicles are modelled and tied to the simulation environment. Finally we look at how the wireless communications layer is implemented.

A. Simulation Environment

RoundaSim is a traffic simulator that uses mixed discrete-continuous modes of time. Like a discrete-event simulator, it keeps a queue of events and their scheduled execution times. Events could be generated by control logic, such as functions that are called periodically, or external factors such as a message being received. Before executing an event, the simulation moves the vehicles forward according to continuous-time models at nanosecond granularity thus avoiding any loss of fidelity. Uncertainties such as communication latency/failures and sensor inaccuracy/failures can be defined at runtime in order to increase the realism of the scenario under investigation. Heterogeneous vehicle types are defined through sensor availability and parametrised physical capabilities (length, acceleration profile, etc.).

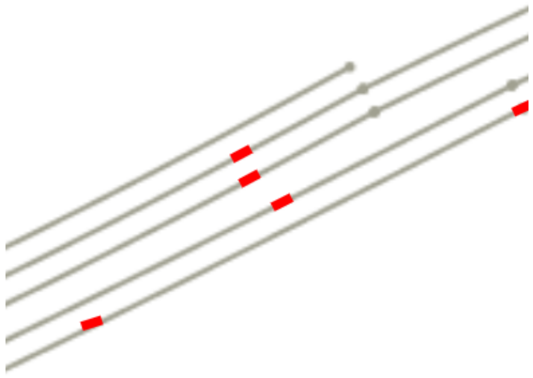
Fig. 2. Tracks on a section of the M50 in Dublin, Ireland

B. Tracks

A key concept in RoundaSim is that of tracks: interconnected, 2-dimensional paths that vehicles can follow. Positions on tracks, called track points, are represented by a track identifier and an offset along the length of the path. Tracks start and end on track points of other tracks, except at the boundaries of a scenario where the tracks are open-ended, it's at these boundaries where vehicles enter and exit the simulation. Tracks can also be connected sideways as lanes, which means that lane changes are possible between the tracks. A lane change requires a new track to be generated at run-time. An example of tracks for a highway scenario is given in Figure 2. Track areas are represented as sets of (track id, start offset, end offset) tuples called track ranges. The benefit of these 1-dimensional representations is that both the simulation logic and vehicle control logic can be implemented efficiently and with relatively simple algorithms. The use of tracks also allows a straight-forward conversion into 2-dimensional geometry in order to simulate sensors and wireless networking, and provide a rich visualization. Using the tool shown in Figure 1, users can define a scenario comprising the primary tracks that vehicles can follow and buildings that can be observed using LIDAR. Figures 2, 5, and 6 show different visualizations produced by the simulator. Section IV-A explains how driving is implemented atop of tracks.

C. Localization

Vehicles have access to basic localization sensors for measuring velocity, orientation, and position. The velocity sensor simply takes the current velocity of the vehicle from the state of the simulation. The orientation and position sensors are implemented by translating the current track position of the vehicle, consisting of a track identifier i and an offset o, into 2 dimensions. The path of the track i is followed from the start for a distance of o to obtain the 2-dimensional position p. The orientation of the vehicle is the orientation of the line segment on which point p lies. Position can also be provided as a track point, which simplifies the control logic.

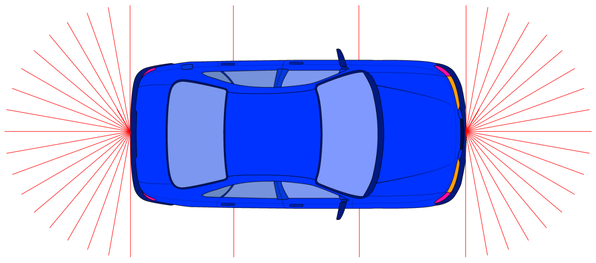
Fig. 3. LIDAR Sensors on the vehicle

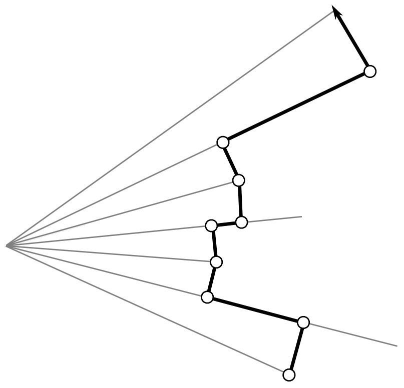
Fig. 4. Converting LIDAR beams to a ring of points

D. LIDAR

LIDAR simulation is implemented using raycasting. By default, vehicles are equipped with two 180° sensors on the front and back of the vehicle with 1° angular resolution, in addition to two single-beam sensors on both sides for sideways observations as shown in Figure 3. This gives vehicles a 360° view of their surroundings, similar to Google's autonomous vehicles [17], but in a 2-dimensional plane. Periodically, beams are drawn from the sensor and intersected with the polygons of other vehicles and buildings. The distance between the position of the sensor and the closest intersection is the value of the distance measurement for that beam. The output of the LIDAR sensor is a set of distance measurements associated with individual beams.

A useful transformation is to convert the distance measurements into a track area, such that the measured distances (empty space) can be associated with the driving paths of vehicles. The transformation can be performed in 3 steps. First, the measurements are converted into a 2-dimensional ring (polygon) around the vehicle. Second, the ring is intersected with the tracks to obtain a set of track boundaries. Finally, the track boundaries are converted into an empty track area.

The first step of the transformation constructs beams from the sensor positions. For every pair of adjacent beams, the projection of the tip of the shortest beam onto the longest beam is added to a ring as shown in Figure 4. This determines the area in between the beams that is actually empty as a polygon. The polygon is translated to the 2-dimensional

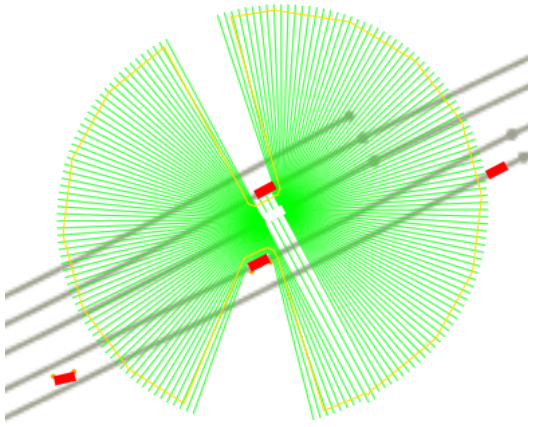
Fig. 5. LIDAR beams measured by a vehicle are converted to a polygon

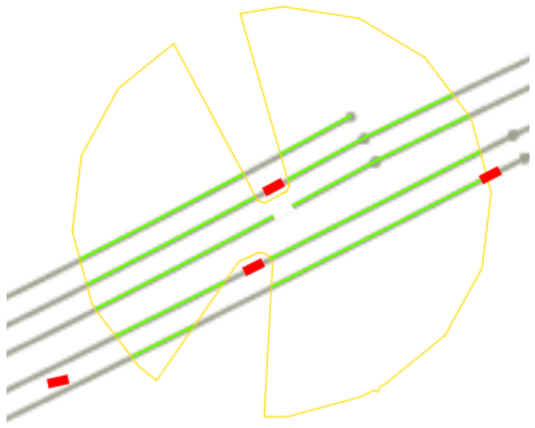
Fig. 6. Polygon converted to 1-dimensional track ranges

position and orientation of the vehicle (Figure 5). To factor in potential inaccuracy in the position sensor, *polygon offsetting* can be used to reduce the size of the polygon by the worst-case inaccuracy value.

The second step of the transformation converts the polygon into a set of track boundaries, which are (track id, offset, forward/backward) tuples. This is achieved by intersecting the polygon with the paths of the tracks. To make this operation efficient, a spatial data structure is used to index the individual line segments of the tracks at the start of a simulation. When an intersection is found, the direction of the boundary is determined to point inwards into the polygon. The offset of the track boundary is found by taking the distance between the start of the line segment and the intersection plus the lengths of the all the preceding line segments. When all intersections are converted to boundaries the second step is completed.

The third and final step of the transformation converts the set of boundaries into a set of track ranges. From an arbitrary, unmarked boundary, a breadth-first search is started to find the other boundaries by following the tracks in the direction of the boundary. The track ranges that are traversed in the search are added to the track area. When the search encounters another boundary, it is marked. Since an area may consist of multiple partitions, the search process is repeated until all boundaries are marked. The resulting set of track ranges is the final empty track area. An example is shown in Figure 6.

The empty track area can be used in additional transformations, which are effectively part of the control logic. By tracing the route of the vehicle from its current position to the edge of the track area, a track-aware forward-looking distance is obtained that can be used in acceleration decisions. An interesting side effect of using the empty track area for distance measurement is that the curvature of the road and the line-of-sight of the vehicle will be taken into account.

E. WLAN

The wireless network in RoundaSim is simulated using SWANS [18] (Scalable Wireless Ad-hoc Network Simulator), a Java library for simulating an ad-hoc 802.11 network. It is normally used in combination with JiST, a virtual machine simulator. The JiST execution model used by SWANS is not entirely compatible with a discrete event simulator that uses explicit scheduling, but the modifications to make SWANS compatible with RoundaSim are minor. The parameters used to configure SWANS are taken from the 802.11p standard for wireless access in vehicular environments [19]. Rayleigh fading and two-ray propagation [20] are used to model the physical channel. Control software running in the simulator uses SWANS through generic *unicast*, *broadcast*, and *receive* interfaces.

IV. Controllers & Car-Following Models

Within our framework, a controller is the implementation of a car-following model, and possibly a coordination protocol, as a Java class of which an instance is created whenever a new vehicle enters the scenario. The physical capabilities of the vehicle such as WLAN and sensors are exposed to the controller through an abstract set of interfaces. Conceivably, the interfaces could be implemented on a real car that the controller could steer.

The inputs of a controller are the results from sensor data and the properties of the vehicle, the output is actuation. Vehicle control is implemented by a set of event handlers. The handlers act in response to incoming sensor data, received messages, or periodic events. The controller is constructed with a set of capabilities, which are interfaces to simulated sensors, WLAN, actuators, properties of the vehicle, road map, and a few more administrative interfaces such as logging and the random number generator. This section shows how a controller implementing a car-following and lane changing model for driving on highways can be defined using the input and abstractions provided by RoundaSim.

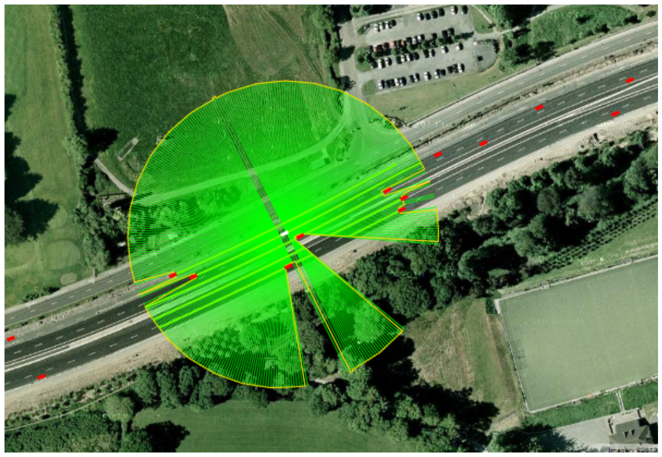

Fig. 7. Automated vehicles view of a highway

Actuation for this highway model consists of setting the acceleration, indicator, and lane changes to follow.

A. Driving

Driving in the simulator is implemented by increasing the offset of the position of a vehicle on a track according to its velocity and acceleration until the it reaches its next track. A vehicle has a queue of track identifiers representing a sequence of tracks to follow. When a vehicle reaches the track point on which the track at the head of the queue starts, its position changes to the start of the new track and this track is removed from the queue, an example of this would be taking an off ramp on a highway. Otherwise, the vehicle changes to the track point to which the end of the current track connects, i.e. remain on the highway. A controller can steer the vehicle across a scenario by adding tracks to the queue. It can also generate a new track at runtime before doing so to facilitate lane changes. The use of tracks for driving avoids the need to implement complex steering logic in the controller, allowing developers to focus on cooperative and behavioural aspects, which can be very complex in itself. Moreover, this implementation is efficient enough to run simulations in real time on commodity hardware even when taking millions of arbitrarily small time steps per second.

B. Indicator Control

To support lane changes, vehicles have indicators and indicator detectors. A controller can set the indicators of the vehicles to 3 states: off, left on, right on. Other, nearby vehicles can detect the presence of a vehicle indicating towards the current lane of the vehicle and the distance to the vehicle. The detector is implemented in the simulator by projecting the 1-dimensional positions of vehicles that are indicating to the lane that lies in the direction of the indicator. If the projected position lies in an area of 60m ahead of the vehicle, an indicator is detected. The distance to the closest indicator in the area, if any, is returned to the controller. We expect that the indicator detector can be implemented using camera sensors, but a real-world implementation is beyond the scope of this paper. Automated vehicles can also use radio signals to communicate indicator states, but indicator detection is still necessary to interact with human-driven vehicles and automated vehicles that do not support the same communication protocol.

C. Acceleration Control

To set acceleration, the controller uses the IDM, which provides a 1-dimensional acceleration value. The IDM function has several parameters [21]: desired speed s, desired time headway T, vehicle length l (set to 4.12m - the average length of a passenger car), minimum spacing s (set to 2m), maximum acceleration a (set to 3.4m/s^2), and comfortable braking deceleration b (set to 3m/s^2). Additionally, the IDM function takes the speed of the vehicle, the distance to the predecessor, and the speed of the predecessor as parameters. The speed of the vehicle is assumed to be directly provided by a sensor. The minimum of the length of the empty route ahead and the distance to the nearest indicator (if any), is used as the forward-looking distance.

Since the measured distance is limited by the line-of-sight and the range of distance sensors, the value is often lower than the real distance between vehicles. The effect is that the observed free-flow speed, dependant on the current traffic density, is typically lower than the desired speed of the controller. The desired speed and desired headway need to be corrected to factor in this effect, which is shown in the evaluation.

D. Lane Change Control

We have modelled lane changes as follows. When a suitable opportunity arises, vehicles may change lanes. A vehicle may always decide to try to change to the preferred side of driving (slow lane), and may decide to change to the other side if the speed of the current predecessor is more than 5% below the desired speed of the vehicle. Whether a vehicle can make the lane change depends primarily on, the speed of, and distance to, the potential future predecessor and successor on the target lane. The controller requires the vehicle to have at least 1.5s of time headway between a future successor or predecessor to safely initiate a lane change. Additionally, the future successor needs to be slower than the vehicle and the future predecessor needs to be faster. When these conditions are met, the vehicle enters the lane change procedure.

This can easily be implemented using the track abstraction. The lane change procedure goes through two phases: indicating and lane changing. At the start of the indicating phase, the vehicle generates a new track for the path of the lane change. The track is a Bézier curve with 2 control points on the current lane, and two control points further down the target lane. The total length of the lane change depends on the velocity of the vehicle and is given by the function $0.0118v^2 + 0.0862v + 20.943$ for speed v, which is based on observations by Naranjo et al. [22]. The start of the lane change track lies a distance of $0.85v+5$ ahead of the vehicle when it decides to change lanes. These values were found by studying the visualization until a smooth traffic flow arose.

After generating a lane change track, the vehicle adds it to the head of the queue of tracks to follow and starts its indicators in the appropriate direction.

The lane change starts when the vehicle enters the lane change track. While on the track, the vehicle keeps indicating. As a rule, the indicator detector in RoundaSim projects the position of a vehicle on a lane change track into its target lane. This means that vehicles on the target lane will detect the presence of the vehicle both before and during the lane change and treat it as a predecessor. The combination of vehicles reacting to indicators and the distance that vehicles require to vehicles on the target lane before initiating a lane change is critical to the safety of the lane change. While a formal proof is beyond the scope of this paper, the evaluation encompasses millions of simulated driving hours without any crashes occurring.

V. Evaluation

To evaluate the validity of traffic simulations in RoundaSim, we simulate a highway scenario and compare traffic properties to models for multi-lane highways from the Highway Capacity Manual based on results from a National Cooperative Highway Research study. While our simulator and framework cater for traffic composed of vehicles of several degrees of automation, this evaluation focuses on homogeneous traffic, without the use of wireless communication, for validation purposes.

A. Experiment

The experiment simulates a 2km two-lane highway (one-way). Vehicles are spawned at the start of either lane at a configurable rate with time between vehicles following a Poisson distribution. However, a vehicle is omitted when another vehicle is too close to the starting point for a new vehicle to enter without having to adjust its desired speed. The actual flow rate could therefore be lower than the configured spawn rate. Once the vehicle is created, a highway controller is started that drives the vehicle to the other side of the highway, changing lanes if necessary.

A large number of runs are performed with different spawn rates and different desired speeds, in blocks of 1 hour. Each run is repeated 5 times for the same set of the parameters, but a different random number generator seed. However, for the metrics of interest, we find that the differences between runs are negligible.

B. Results

During simulation runs, the average speed, density, flow rate, and headway of vehicles are measured after 5 minutes, when the highway has filled with cars. Results are given for a particular *free-flow speed* (FFS). This is the average speed of vehicles in low density, unencumbered traffic. The free-flow speed can be controlled with the desired speed and headway parameters of the IDM. IDM is normally used to make an approximation of human behaviour under idealized conditions. It does not take into account limitations of automated vehicles, such as a limited distance sensor

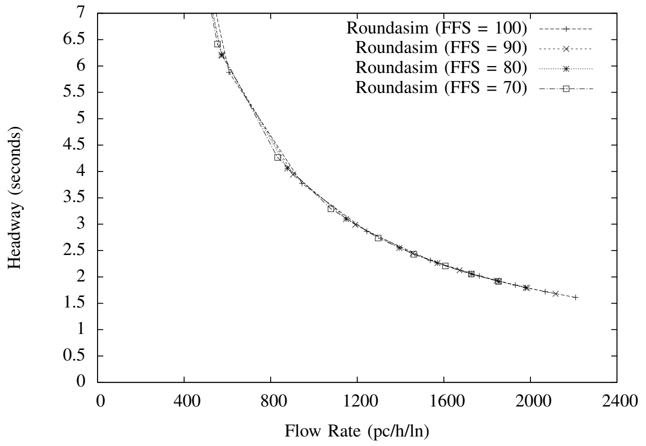

Fig. 8. Average headway vs. Flow rate

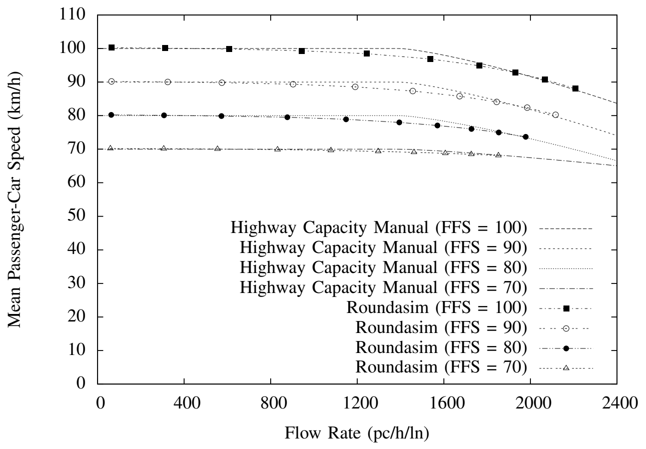

Fig. 9. Average speed vs. Flow rate

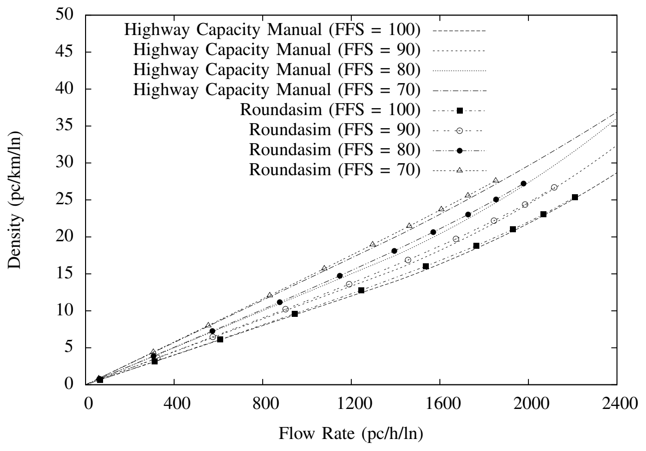

Fig. 10. Average density vs. Flow rate

IDM Parameters		
FFS	**Desired Speed** s	**Desired Time Headway** T
70km/h	70.7km/h	0.44s
80km/h	81.5km/h	0.72s
90km/h	92.2km/h	0.75s
100km/h	103.0km/h	0.82s

TABLE I
IDM PARAMETERS TO MATCH FFS VALUES FROM HIGHWAY CAPACITY MANUAL

range (80m). We used the speed and headway parameters to correct the behaviour of IDM in order to mimic the behaviour of real vehicles. Table I shows the IDM parameters used to obtain a particular free-flow speed. The overall behaviour of vehicles, which depends on factors such as line-of-sight, is more conservative than what the IDM would predict. The desired headways used are therefore set to be relatively low to encourage more aggressive (closer) driving. However, the actual headways depend only on the flow rate, as shown in Figure 8.

Figure 9 shows the average speed of vehicles according to the Highway Capacity Manual and RoundaSim for different free flow speeds compared to the flow rate expressed as passenger cars per hour per lane. For most flow rates, the results produced by RoundaSim are very similar to the capacity manual, except around 1400pc/h/ln. The capacity manual assumes this to be the cut-off point from which speed starts to deteriorate, but we find the decline to start at lower densities. This may be due to simplifying assumptions in the capacity model. Nevertheless, the average speeds in RoundaSim are within 5% of the reference value in all cases. Figure 10 shows the average density for different flow rates with a similar pattern. For both the flow and the density, RoundaSim is able to reproduce the traffic properties found in the Highway Capacity Manual very accurately.

VI. CONCLUSION

We have presented RoundaSim, a novel simulation engine and framework supporting advanced sensors, communication capabilities, and actuators in order to evaluate control strategies for heterogeneous vehicle types and their interaction with each other. Using a car-following model and controller for highway scenarios, which supports lane changes and sensor input, we demonstrated the ability of RoundaSim to accurately reproduce the expected speeds and densities of traffic on highways. We consider RoundaSim as filling the gap between traffic and vehicle simulators for evaluating control logic and coordination protocols.

VII. ACKNOWLEDGEMENT

This work was supported, in part, by Science Foundation Ireland grant 10/CE/11855 to Lero - the Irish Software Engineering Research Centre (www.lero.ie).

REFERENCES

[1] M. Bouroche, B. Hughes, and V. Cahill, “Real-time coordination of autonomous vehicles,” in *Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2006, pp. 1232–1239.
[2] M. Slot and V. Cahill, “A reliable membership service for vehicular safety applications,” in *Intelligent Vehicles Symposium*. IEEE, 2011.
[3] J. Markoff, “Google cars drive themselves, in traffic,” *The New York Times*, vol. 10, p. A1, 2010.
[4] T. R. Board, *Highway capacity manual*, ser. Special report. National Research Council (U.S.)., 2010.
[5] M. Fellendorf and P. Vortisch, “Validation of the microscopic traffic flow model vissim in different real-world situations,” PTV AG, Tech. Rep., 2001.
[6] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, “Sumo - simulation of urban mobility: An overview,” in *International Conference on Advances in System Simulation (SIMUL)*, Barcelona, Spain, 2011.
[7] A. Kesting, “Movsim,” http://www.movsim.org/, June 2011.
[8] M. Gueriau, R. Billot, J. Monteil, S. Hassas, and N.-E. El Faouzi, “Agent-based cooperative traffic modeling and simulation,” in *Proceedings of the 94th Transportation Research Board Annual Meeting*, Washington, DC, USA, 2015.
[9] F. Hendriks, M. Tideman, R. Pelders, R. Bours, and X. Liu, “Development tools for active safety systems: Prescan and vehil,” in *International Conference on Vehicular Electronics and Safety (ICVES)*. IEEE, 2010, pp. 54–58.
[10] Y. Mylonas, M. Lestas, A. Pitsillides, and P. Ioannou, “Speed adaptive probabilistic flooding for vehicular ad-hoc networks,” in *Personal Indoor and Mobile Radio Communications (PIMRC), 2011 IEEE 22nd International Symposium on*, sept. 2011, pp. 719 –723.
[11] R. Bhakthavathsalam, S. Nayak, and M. Srikumar, “Expediency of penetration ratio and evaluation of mean throughput for safety and commercial applications in vanets,” in *Ultra Modern Telecommunications Workshops, 2009. ICUMT ’09. International Conference on*, oct. 2009, pp. 1 –5.
[12] S. Panwai and H. Dia, “Comparative evaluation of microscopic car-following behavior,” *Intelligent Transportation Systems, IEEE Transactions on*, vol. 6, no. 3, pp. 314–325, 2005.
[13] R. Wiedemann, “Modelling of RTI-elements on multi-lane roads,” *Advanced Telematics in Road Transport*, vol. DG XIII, 1991.
[14] R. E. Wilson, “An analysis of gipps’s car-following model of highway traffic,” *IMA Journal of Applied Mathematics*, vol. 66, no. 5, pp. 509–537, October 2001.
[15] M. Treiber, A. Hennecke, and D. Helbing, “Congested traffic states in empirical observations and microscopic simulations,” *Phys. Rev. E*, vol. 62, no. cond-mat/0002177, pp. 1805–1824, 2000.
[16] A. Kesting, M. Treiber, and D. Helbing, “General lane-changing model mobil for car-following models,” *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1999, no. -1, pp. 86–94, Jan. 2007.
[17] E. Guizzo, “How googles self-driving car works,” *IEEE Spectrum Online, October*, vol. 18, 2011.
[18] R. Barr, Z. J. Haas, and R. van Renesse, *Scalable Wireless Ad hoc Network Simulation*. CRC Press, Aug. 2005, pp. 297–311.
[19] IEEE, “IEEE standard for information technology–telecommunications and information exchange between systems–local and metropolitan area networks–specific requirements part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications amendment 6: Wireless access in vehicular environments,” *IEEE Std 802.11p-2010*, pp. 1 –51, 2010.
[20] J. Proakis and M. Salehi, *Digital communications*, ser. McGraw-Hilll higher education. McGraw-Hill Higher Education, 2008.
[21] V. Punzo, M. Montanino, and B. Ciuffo, “Do we really need to calibrate all the parameters? variance-based sensitivity analysis to simplify microscopic traffic flow models,” *Intelligent Transportation Systems, IEEE Transactions on*, vol. 16, no. 1, pp. 184–193, Feb 2015.
[22] J. Naranjo, C. Gonzalez, R. Garcia, and T. de Pedro, “Lane-change fuzzy control in autonomous vehicles for the overtaking maneuver,” *Intelligent Transportation Systems, IEEE Transactions on*, vol. 9, no. 3, pp. 438–450, 2008.