Managing autonomous vehicles at intersections

Rolf Naumann and Rainer Rasche, Mechatronics Laboratory, University of Paderborn
Jürgen Tacken, C-LAB, University of Paderborn and Siemens Nixdorf Informationssysteme AG

In the past, more and broader roads have been built to manage the growing traffic. However, these approaches will be insufficient for future traffic management, because of environmental concerns and a lack of space, especially in urban areas.

Since 1970, centralized traffic-control systems have been installed to meet the demand of increasing mobility. Thus, roads can be used much more flexibly. Nevertheless, resources for centralized systems lack scalability, and the maintenance of such systems involves high cost and much effort.

The significant and rapidly increasing improvements in communication and information technology, as well as the decreasing prices, are motivating a new look at the handling of the growing road-traffic problem. The idea is to put more intelligence into the vehicles to benefit from decentralized solutions. However, the problem’s complexity rises with the number of participating vehicles. One promising solution is for each vehicle to put its own computing power into the system, thus helping to meet the growing demand for computing power.

As an example, we’ve developed an intersection-management strategy for autonomous vehicles. In this strategy, a vehicle crosses an intersection in coordination with other vehicles, collision-free and following a predetermined strategy—for example, to optimize traffic flow or minimize emissions. This relieves the driver of his or her responsibility within a “sphere of influence” around the intersection where the management is at work. The driver also relinquishes some individuality in favor of better traffic flow between all participants. This strategy can operate in today’s traffic to increase safety and avoid congestion. We’ve validated it using formal analysis and simulation.

The intersection-management strategy

Modeling intersection management involves analyzing discrete and continuous vehicle characteristics. Discrete characteristics describe the safety-critical aspects of intersection management, such as collision avoidance and deadlock avoidance (guaranteeing that there is no endless waiting). Continuous characteristics describe such aspects as the traffic-flow rate, vehicle waiting time, fuel consumption, and exhaust emissions. To account for these characteristics, our strategy comprises three parts: the vehicle model, the collision-avoidance algorithm, and the fairness strategy.

The vehicle model. The vehicle model comprises a continuous system and a discrete system (see Figure 1). The continuous system comprises the vehicle dynamics (for example, the accelerating and braking behaviors, drive-train vibrations, and tire slip—all modeled by differential equations), with the corresponding control of lateral and longitudinal dynamics. The discrete system contains the inherent partial intelligence that guarantees communication with other vehicles in the intersection and employs the information gathered to conduct velocity and route planning.

When the vehicle enters a sphere of influence, its discrete system obtains information on the intersection’s geometry and management strategy from a traffic beacon. Additionally, it might detect other vehicles by sensor systems and communication as well as recognize its vehicle’s inertial position in the sphere by traffic markings. The management strategy demands the necessary permissions to enter the intersection and computes a reference velocity for the vehicle from the information available. This becomes the reference value for the continuous system, whose controllers yield velocity and position values. These values have a feedback effect on the strategy of computing the velocity and demanding the permissions, as well as on the whole intersection, including the individual vehicle.

Figure 1 shows the interaction between the continuous system and the discrete system, which computes the reference velocity with respect to the vehicle’s velocity $v$, the distance covered $s$, and whether or not the vehicle possesses the required permissions. The controller calculates the required braking force or motor torque $\tau$ with respect to the reference and real values of $v$ and $s$.

The collision-avoidance algorithm. Figure 2 sketches the intersection and the trajectories corresponding to each direction. The variable $d$ denotes the radius of the sphere of influence where the management
is at work. Localization of all intersecting points of the trajectories (16 in all) yields all potential points of collision. At first, the real lengths and widths are neglected, so these potential collision points can be seen as the respective centers of the critical regions of the radius \(r\), where only one vehicle at a time may stay. The algorithm adapts \(r\) to the geometry of the vehicles. On the basis of a vehicle’s projected course, that vehicle can define the regions it will pass through. Thus, a vehicle going from west to east will pass through the critical regions 1 to 5. In region 4 it might collide with a vehicle going from south to north. Every region has exactly one access permission that a vehicle must obtain before being permitted to enter this region. When leaving the critical region, the vehicle will release that permission.

Our strategy uses semaphores (permissions) for collision avoidance. The semaphore handling resembles the token-ring principle in a computer network. That is, only the computer in the network (the vehicle in the sphere of influence) that possesses the send-token (the semaphore) will send out data; all the others will receive data. Then the first computer will pass the token to the next computer, which is then allowed to send, and so on. We’ve slightly modified this principle so that the whole set of semaphores are transferable among vehicles in the sphere of influence.

This feature resembles a basket full of entry tickets. Vehicles being handed this basket will take from it the permissions they need—if those permissions have not been assigned to others—and put the permissions back as soon as they have passed the critical region. The first vehicle approaching the intersection generates the set of permissions, according to the relevant collision-avoidance algorithms.

Fairness by priority. The critical-region model we’ve just described should guarantee collision-free traffic, but it yields no information on permission management. So, we’ve incorporated the strategy of handing out permissions according to the highest priority. To include individual features, such as idling time, velocity, or accumulation of vehicles in certain directions (during rush hour, for example), we have implemented a heuristic parameterized computation of priority \(P_i\) for each vehicle.

We use weighting factors to enforce a particular vehicle behavior:

\[
P_i = k_d \cdot d + k_v \cdot v + k_a \cdot a + k_t \cdot t + \sum_{j=1}^{n} P_j
\]

\(k_d\) denotes the distance covered in the sphere of influence, \(k_v\) is the velocity, and \(k_t\) is the idling time. So, no vehicle will wait endlessly.

By adding up the priorities of vehicles following in one direction (weighted by means of \(k_d\)) we can adapt the traffic flow to each situation online without altering the algorithm. If a traffic jam occurs in one direction—for example, because of a large factory closing after work—the vehicles from this direction will have the highest priority because of the summation. In this way, the strategy also accounts for ambulances. They give themselves top priority; therefore, they can cross the intersection without further delay.

If not all permissions necessary for the crossing are in the basket of entry tickets, a vehicle’s own priority will be marked on the permissions available if its priority is higher than those previously set by other vehicles. In this way, the vehicle employs its priority to secure an option for the respective entry permissions.

Validating the strategy

Because our strategy incorporates discrete and continuous characteristics, we had to take two different approaches to validate it. To guarantee collision avoidance, discrete characteristics require proof by formal analysis. Therefore, we used state-space analysis of Petri net models of those characteristics. Because continuous characteristics involve complex algorithms, a formal proof is not feasible. So, we used simulation to validate those characteristics.

Validating the discrete characteristics. Petri nets are bipartite directed graphs consisting of places and transitions. To define a system’s behavior, places may contain tokens. The edges between places and transitions (partly) define the possible flow of tokens caused by the firing of transitions.

We use a higher form of Petri nets—Predicate/Transition–Nets (PrT–Nets)3—
Figure 3. Predicate/Transition-Nets: (a) a basic Prf-Net showing the firing of a transition, which adds two integers; (b) a Prf-Net model for part of an intersection.

for system modeling. In contrast to the nonindividual tokens in Petri nets, in Prf-Nets, the token may be constant tuples, the edges may be annotated by sums of constant or variable tuples, and transitions may carry first-order formulas over a set of constants and variables. When a transition fires, it removes tokens from its input places and produces some new tokens on its output places, according to the flow specified by the edges and the annotations. Figure 3a shows the firing of a transition in a simple Prf-Net, which adds two integers.

Some of the classical Petri net analysis methods, such as state-space analysis, partial state-space analysis, and the computation of linear invariants, have been adapted to colored Petri nets. Colored Petri nets are very similar to Prf-Nets. They differ in the possible annotations of the net elements. Prf-Nets are a bit more powerful, but therefore the analysis of Prf-Nets is a little more difficult. We are trying to apply these methods to Prf-Nets. Currently we can do a state-space analysis for nonhierarchical Prf-Nets models. For the modeling and analysis of Prf-Nets, we use the SfA (System Engineering and Animation) environment, developed at C-LAB (http://www. c-lab.de/sea).

Passenger safety. Figure 3b shows a Prf-Net that models part of the collision-avoidance algorithm. The figure depicts only a quarter of the whole intersection; the rest is modeled similarly. Places P3 to P7 model the critical regions (see Figure 2). Each of these places has a corresponding semaphore place, S3 to S7, that initially contains a simple token (a token with no special value).

A vehicle that enters this part of the intersection is modeled by a token on the place In0. Such a token contains the desired direction (n, s, w, or e). Every time a vehicle enters the intersection, one of the three transitions (T1 to T3) that model the three possible directions must collect all the tokens from the semaphore places corresponding to all critical regions touched during the crossing. In this way, the semaphore places should guarantee a safe crossing of the intersection. The transitions modeling the leaving of critical regions replace the collected tokens on the semaphore places after the vehicle leaves every region.

To prove that the algorithm for crossing the intersection is safe means to prove that there is never more than one vehicle in every critical region. This is equivalent to the statement that there is never more than one token (representing a vehicle) on each of the places P1 to P16 (the critical regions) in the Prf-Net. The easiest way to prove this is to build an occurrence graph, or state space, of the net that includes all possible states. To build the graph, tokens for the vehicles crossing the intersection must be added randomly to the incoming places (for example, In0 in Figure 3b). So, we added transitions as input for each of those places that randomly produce such tokens, and we built a complete graph containing 11,136 nodes. This graph contains all the possible simulation runs and, therefore, all the possible combinations of arriving vehicles from all four directions. The state-space analysis tells us that the minimum number of tokens for places P1 to P16 is 0 and the maximum number is 1. This verifies the safety of the algorithm.

No endless waiting time at the intersection. State-space analysis can also find possible deadlocks. Every state in the occurrence graph with only incoming arcs is a deadlock. In the occurrence graph of our net, no such state exists, so the net is deadlock-free. We can even prove that all transitions are alive and that the occurrence graph is strongly connected. So, every state is reachable from every other state. This means that every arriving vehicle can cross the intersection safely.

Validating the continuous characteristics. Simulation of different strategies and the continuous characteristics must be hybrid;
that is, the discrete and continuous systems must be combined and simulated together.\(^4\)

The simulation must also take into account the changing number of vehicles and thus the dynamically varying coupling structure of the intersection system. To simulate hybrid systems with dynamically changing structures, we have implemented an object-oriented simulation environment based on the CAMeL tools (Computer-Aided Mechatronic Laboratory) of the Mechatronics Laboratory of Paderborn.\(^6\)

The simulation of three vehicles illustrates how the priority strategy operates.

First, Vehicle 1 arrives from the east and continues westward (see Figure 4a); then, Vehicles 2 and 3 arrive from the north, make up a convoy, and turn west.

Figure 4b shows the distance covered over time. The graph's gradient indicates the actual velocity. Because the convoy's velocity is higher than that of Vehicle 1, it reaches the intersection first and can cross it as a convoy—that is, seen as a logically individual vehicle with a reduced speed while turning. In the meantime, Vehicle 1 arrives at the intersection and is forced to stop because it does not have the corresponding permissions for crossing. When the last vehicle of the convoy has left the intersection, thus releasing the permissions, Vehicle 1 will accelerate and cross the intersection.

Figure 4c displays the course of the priorities over time. Vehicle 1 reaches the sphere of influence first and has the highest priority at first and therefore the permissions. Vehicle 2 enters the sphere of influence after 1.1 seconds. It has at first a lower priority than Vehicle 1, but a higher priority increase because of higher velocity. When Vehicle 3 enters the sphere of influence, the computed priority will be transferred to Vehicle 2 on account of the strategy.

Thus, Vehicle 2 obtains the permissions and keeps them until the convoy has passed through the intersection. The course of the priorities of Vehicle 1 clearly shows the increase that is due to the idling time.

One of the main benefits of our strategy is its decentralization. For example, it transfers most of the necessary infrastructure (computing power, sensor system, and algorithms) to the vehicles. Each vehicle thus becomes an autonomous mechatronic system that, when combined with other vehicles, constitutes a networked mechatronic system.\(^5,6\) Therefore, this approach does not require much additional infrastructure at the intersection. In addition, because we've divided the algorithm into safety-critical parts and fairness parts, we can use different methods for validation and can modify the fairness properties while maintaining verification of collision avoidance.

Further work will include the verification of the continuous characteristics. One important question is how to model fuel consumption or exhaust emissions in the Pr/T-Nets, so that we can exploit the huge mathematical basis of Petri net theory to validate different strategies. We also plan to refine the computation of priorities to take into account optimization criteria such as maximum traffic flow and minimal fuel consumption. Another goal is to formulate...
Call for Articles

Intelligent Integration and Use of Large Text, Image, Video, and Audio Corpora

Guest Editor: Alexander Hauptmann, Carnegie Mellon University

This special issue of IEE Intelligent Systems will describe current research and potential future contributions of intelligent, integrated systems that use text, image, video, and audio (TIVA) sources. The issue is not intended to deal with applications, network delivery, or interface designs for TIVA material; instead, it will emphasize the software processing and analysis of TIVA sources in preparation for later use.

This special issue will deal with two aspects of the exploitation of TIVA sources: (1) What intelligent systems can be built to use or reuse those sources? (2) How can these multimedia sources be used to create smarter programs?

We particularly encourage submissions that demonstrate exchange and cross fertilization across the fields of vision, speech processing, natural language processing, machine learning, and information retrieval, to allow readers to collect ideas and approaches for making TIVA sources accessible and useful in other programs. Although there has been active research in all these areas in specific subfields, this special issue will explore avenues of integration between these topics across some or all of the TIVA source media.

Submit articles (four hard copies in plain VCH format) by Aug. 31, 1998, to:

Alexander Hauptmann, Dept. of Computer Science,
Carnegie Mellon Univ., Pittsburgh, PA 15213-3890
Phone: (412) 268-1448; FAX (412) 268-5576
Email: alex@cs.cmu.edu

All manuscripts will be reviewed by a select panel of referees. Address inquiries about the special issue to Alexander Hauptmann, at alex@cs.cmu.edu.

For further information about IEE Intelligent Systems, visit http://www.computer.com/intelligent.

For full publication guidelines are available at https://computer.org/intelligent/guide.html.

this task as a scheduling problem known from operating-system theory. A typical scheduling problem is the fair allocation of limited resources to processes. This is very similar to our problem if you think about critical regions as resources and vehicles as processes. (Priority-scheduling algorithms seem to be of particular interest for our problem.) We'll focus mostly on the methods for online optimization of the algorithm, which calculates the priorities for the vehicles and thus determines the intersection's throughput. So, the system will be able to adapt automatically to a wide range of traffic situations.

Acknowledgments

This work is supported by the German Research Association (DFG) within the Sonderforschungsbereich (Center of Special Research) 376 (SFB 376), "Massive Parallelism: Algorithms, Design Methods, Applications", at the University of Paderborn.

References


Rolf Naumann is a research scientist at the Mechatronic Laboratory of Paderborn, at the University of Paderborn. His research interests include the design of CAD systems for complex mechatronic systems, especially the design of hybrid systems; parallel computing; and design methods for intelligent controllers. He received his diploma in mechanical engineering at the University of Paderborn and is working toward his PhD. He is member of the VDI (Verein Deutscher Ingenieure—the Association of German Engineers). Contact him at the University of Paderborn, Pohlweg 55, 33098 Paderborn, Germany; naum@mlap.uni-paderborn.de; http://mlap.uni-paderborn.de/

Rainer Rasche is a research scientist at the Mechatronic Laboratory of Paderborn, at the University of Paderborn. His research interests include the modeling and designing of mechatronic systems, parallel computing, and computer networks. He received his diploma in mechanical engineering from the University of Paderborn and is working toward his PhD. Contact him at the University of Paderborn, Pohlweg 55, 33098 Paderborn, Germany; rasche@mlap.uni-paderborn.de; http://mlap.uni-paderborn.de/

Jürgen Tacken is a research scientist at C-LAB, a joint venture between the University of Paderborn and Siemens-Nixdorf Informationsysteme AG. His research interests include the specification and analysis of system designs using predicate/transition-nets, embedded-system design, parallel computing, and the simulation of hybrid systems. He received his diploma in computer science from the University of Paderborn and is working toward his PhD. Contact him at C-LAB, Fürstenallee 11, 33094 Paderborn, Germany; theo@clab.de; http://www.c-lab.de/~theo/