Gulliver: A Test-bed for Developing, Demonstrating and Prototyping Vehicular Systems *

[Extended Abstract]

Mitra Pahlavan Marina Papatriantafilou Elad M. Schiller†
Computer Science and Engineering, Chalmers University of Technology, Göteborg, Sweden
pahlavan@student.chalmers.se ptrianta@chalmers.se elad@chalmers.se

ABSTRACT

Vehicular system designers often use simulation tools in order to prove vehicular systems. The computational complexity of detailed simulations limits the scale of such testings. Therefore, it is often the case that the first full-scale demonstrations of new concepts for vehicular systems are done in proving grounds and testing tracks.

We propose Gulliver as a platform for studying vehicular systems on a large scale open source test-bed of low cost miniature vehicles that use wireless communication and are equipped with onboard sensors. Our approach provides a simpler yet detailed investigation of vehicular systems. This paper presents the platform with its design and a set of applications that could be demonstrated by Gulliver. Gulliver allows the design of vehicular systems to focus on the cyber-physical aspects of the studied problems.

We expect that Gulliver will allow affordability and flexibility for a wider range of researchers to directly contribute to the development of future vehicular systems, such as greener transportation initiatives and zero fatality objectives.

Categories and Subject Descriptors

J.7 [**Computers In Other Systems**]: Real time

General Terms

Experimentation

Keywords

Gulliver, Test-beds, Development, Demonstration, Prototyping, Vehicular Systems

*A detailed version of this paper is available as technical report [23]. Partially supported by the ICT Programme of the European Union under contract number ICT-2011-288195 (KARYON).

†Contact person for correspondence.

MobiWac'11, October 31–November 4, 2011, Miami, Florida, USA.

1. INTRODUCTION

Ultimately, vehicular systems are expected to gear vehicles with autopilot capabilities, improve safety, reduce energy consumption, lessen CO_2 omission and simplify the control of traffic congestion. This dramatic change will be the result of advances in driver assistant mechanisms for navigating, congestion control, steering, speed controlling, lane changing, avoiding obstacles to name a few (see Figure 1). Moreover, other technologies, such as driverless cars and vehicle platoons, might also appear on the road; giving way to future vehicle systems that will be controlled by different types of drivers, i.e., driverless, mechanism assisted drivers and nonassisted ones. We propose to study vehicular systems of low cost miniature vehicles that use wireless communication on a large scale open source test-bed. The test-bed may be geared with onboard sensors, such as cameras, laser, radar, speed sensors, etc. Our approach provides a simpler yet detailed investigation of vehicular systems that will be affordable by a wider range of developers than available today.

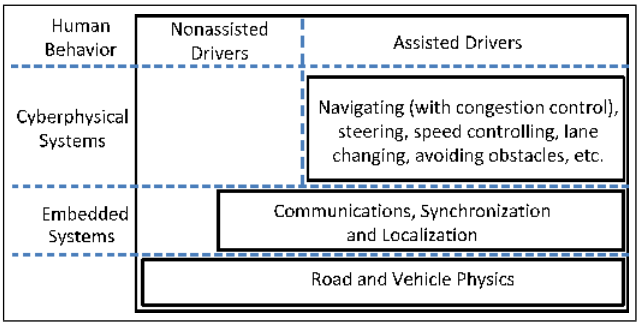

Figure 1: Advanced driver assistance mechanisms and their environment

Vehicular system designers often use simulation tools [5, 14, 17, 20, 30] to prove new concepts. Simulation tools allow extensive testing of software components, say, by using fault injection methods [1, 13]. Simulators can also deal with complex mathematical modeling of physical objects (e.g., vehicles) and their controlling computer systems (see Figure 2). The computational complexity of detailed simulations of future vehicular systems limits the scale of testing and often does not allow extensive system testing for a large number of vehicles. Additional limitations include the absence of hu-

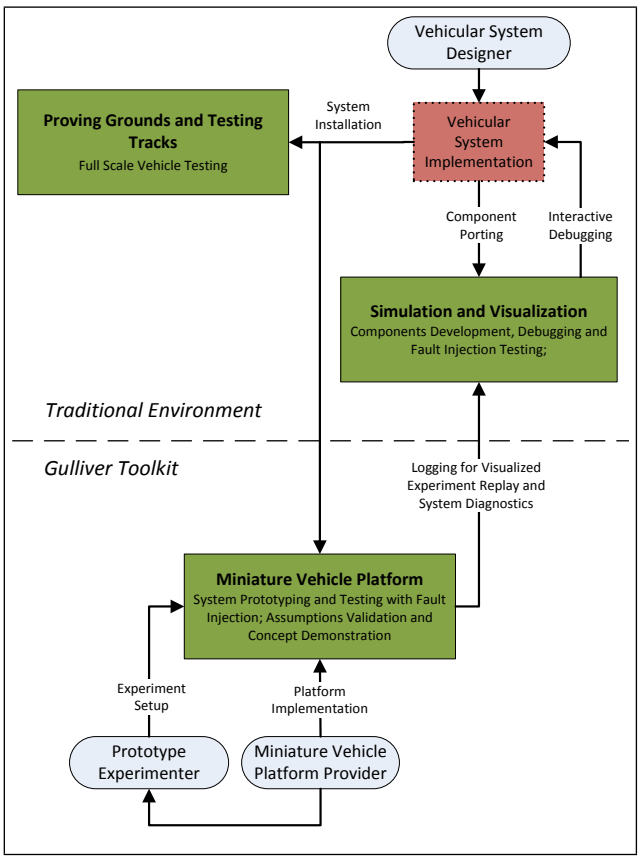

Figure 2: Traditional development environment and the Gulliver toolkit are depicted above, and respectively, below the dashed line. The traditional development environment allows the vehicular system designer to simulate and visualize components of the vehicular system before installing it and testing it in proving grounds and testing tracks. The Gulliver toolkit allows the prototype experimenter to use the test-bed for setting up an experiment in which the vehicular system is tested over a miniature vehicle platform. The experiment is logged for later execution visualization by the simulator.

mans in the loop or the assumption that computer programs can always predict driver reactions.

Due to these limitations, the first demonstrations of new vehicular systems are centered around proving grounds and testing tracks. DemonstRator [24] is an example of a platform that provides a prototype for vehicular systems. It considers full-scale vehicles, which require isolated testing grounds and considerable protections against vehicle crashing. Proving ground facilities are not affordably accessible to a wide range of universities, public research and engineering institutes. By reducing the demonstration costs, we could allow a greater engineering force to participate in the efforts for greener transportation systems with near zero fatalities.

1.1 Proposed toolkit

Recent advances in the field of mobile robots allow the ad hoc deployment of a fleet of miniature vehicle that are controlled remotely by human drivers or computer programs. These affordable miniature vehicles can greatly simplify the development of the cyber-physical [18] layer of new vehicular systems (see Figure 2). Namely, a prototype experimenter can test the vehicle system that is installed on the miniature vehicle platform. These tests can include onboard fault injection. Moreover, the experiment execution can be logged and later replayed and visualized in the simulator.

In order to bring the prototyping of cyber-physical layer into the practical realm, one can take a range of approaches for emulating and substituting relevant system elements. For example, the human driver can be included in the loop of the miniature vehicle control via an onboard or remote computation, hand-held wireless devices, or driver simulation cockpits with multi-angle video streaming in addition to what looks like, sounds like and feels like emulation of the vehicles and their environment.

This paper focuses on the design of the miniature vehicle platform that we name Gulliver. Gulliver's greatest strength lies in its ability to prototype cyber-physical technologies for vehicular systems. We assume that problems related to the interaction among vehicles on the road can be solved before the prototyping phase (see Figure 1). Thus, we can follow the approach in which miniature vehicles can represent full-scale vehicles in the test-bed.

It is up to the prototype experimenter to decide which relevant parts of the embedded system should be included onboard of the vehicle. For example, one of the key difficulties is to understand the impact of vehicle-to-vehicle communication, such as the IEEE `802.11p` standard, which is inherently subject to interferences and disruption. Vehicular systems have safety critical requirements that must mitigate uncertainties, such as the communication related issues. Gulliver can facilitate the study of vehicle-to-vehicle communication at the MAC layer and above, e.g., dynamic bandwidth allocation, pulse synchronization, contention control and packet routing, to name a few. We note that the physical layer can also be studied in Gulliver. However, the platform designer should take into account signal shadowing and fading. E.g., the designer can include signal blocking objects in the test-bed area and onboard the miniature vehicle.

Our approach enables the vehicular systems designer to focus on the cyber-physical aspects of the problem. Moreover, by including the human driver in the loop, many of the designer assumptions about the cyber-physical system and the human driver can be validated. Thus, the Gulliver test-bed is a multifaceted toolkit for testing, prototyping and demonstrating new vehicular systems. The test-bed feedback capabilities and human interaction units are imperative debugging tools for vehicular system developers.

1.2 Related work

Wireless ad hoc networks are often simulated before their installation, e.g., TOSSIM [21], which is TinyOS mote simulator. Wireless Vehicular Networks (VANETs) are often simulated by systems that have a wireless ad hoc network simulator, and a microscopic traffic flow simulator, such as SUMO (Simulation of Urban MObility) [15]. It provides information to the vehicles about how they can traverse along their routes. Similarly, DIVERT (see [8]) is a traffic simula-

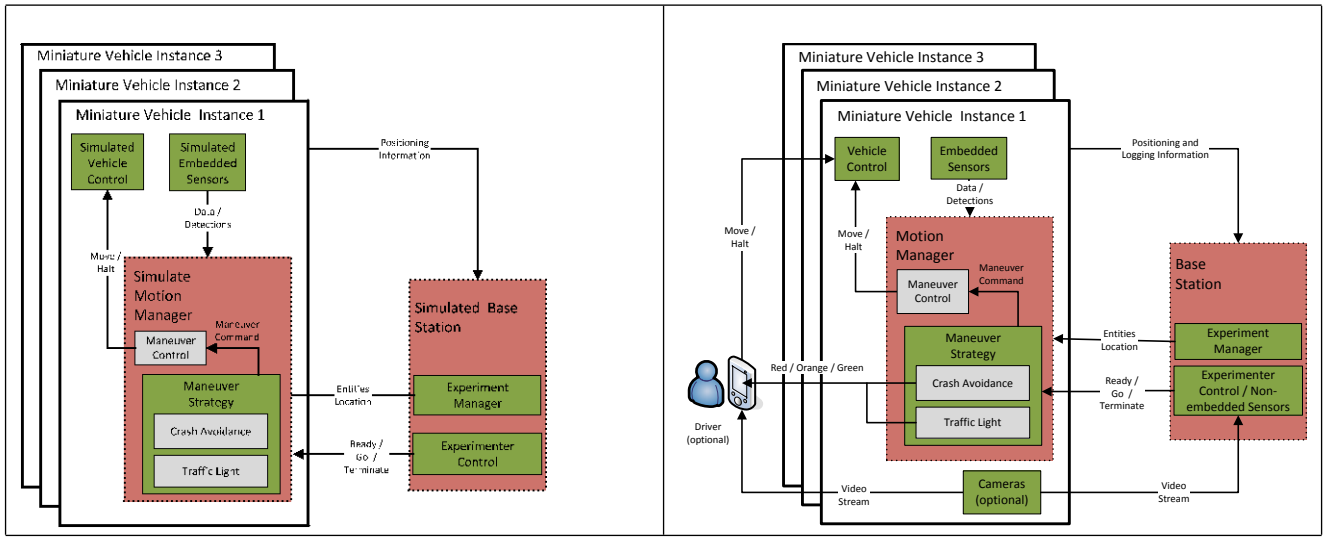

Figure 3: Component diagrams of the Simulator (left) and the Miniature Vehicle Platform (right).

tor that models vehicles' mobility and their communication. For the first time, Gulliver extends the use of simulators and allows testing of new concepts in a test-bed that includes miniature vehicles. The proposed approach allows prototyping of vehicular systems in a more practical realm than computer simulations.

1.3 Our contribution

Gulliver presents a multifaceted toolkit for testing vehicular systems in a practical realm. It lies between computer simulation and full-scale vehicle models, and as such, it simplifies and reduces the costs of vehicular system prototyping and development. Gulliver's greatest strength lies in its ability to prototype cyber-physical technologies for vehicular systems. By that, it allows the system designer to focus on cyber-physical aspects of algorithmic problems in vehicular systems and their networks.

In addition to presenting Gulliver as a concept, this paper outlines the design of its key components. We explain how the miniature vehicles can follow a strategy that allows them to safely traverse the test-bed floor along their Route Plan (see Section 3).

We report on our preliminary implementation efforts. We explain how the prototype experimenter can assure that the miniature vehicles can drive along the lane markings. We also explore additional technological challenges (see Section 4).

Further, vehicular systems will enable vehicular interaction, promote cooperation and will be the first cyber-physical systems to reach the scale of millions of units. Currently, no safety-critical system comes close to this scale. Gulliver design is the first to facilitate the detailed investigation of the vehicle interaction and emerging patterns among hundreds and even thousands of units of a cyber-physical system. These investigations are imperative for the design and development of advanced driver assistance mechanisms, such as virtual traffic lights, vehicle platooning, coordinated contention controls, and coordinated lane changes, to name a few.

2. PRELIMINARIES

We list the assumptions, definitions and notations that are used in this paper.

2.1 Road Map, Route Plan and Lane Marking

We consider drivers of miniature vehicles that plan their way using *road maps*, which the prototype experimenter provides. The driver's *route plan* sets the course of travel and assists with the vehicle navigation, e.g., "on the next intersection, turn left!"

The roads include *segments* that have the index set $S = \{1, 2, \ldots n\}$. Each segment, $s_i \in \{s_k\}_{k \in S}$, has at least one entrance or exit. We define $in(s_i), out(s_i) \subseteq S$ as the sets that include s_i's entrances, and respectively, exits. We define road intersections as segments that have more than one entrance. A *road map* is a directed graph $G(\mathcal{S}, E)$, in which $\mathcal{S} = \{s_k\}_{k \in S}$ is the set of segments (vertices) and $E = \{\langle s_u, s_w \rangle \in \mathcal{S} \times \mathcal{S} : in(s_w) = u \wedge out(s_u) = w\}$ (edges), where s_u is the segment from which the vehicles can enter segment s_w. We note that the prototype experimenter can show the *route plan* to the driver by presenting a directed path that leads from source to destination on the graph $G(\mathcal{S}, E)$. In our pseudo-code, we use the array $RoutePlan_{v_i}[]$ for listing the segments that vehicle v_i should traverse from source, $RoutePlan_{v_i}[s]$, to destination, $RoutePlan_{v_i}[d]$, where $s = 0$ and $d = sizeof(RoutePlan_{v_i}) - 1$.

Given a road map, $G(\mathcal{S}, E)$, we require the prototype experimenter to define how each segment, $s_i \in \mathcal{S}$, is situated on the test-bed. Lane markings are used to align vehicles on the road, i.e., the driver should steer the vehicle between two parallel dashed lines that are marked on the test-bed floor. We also consider virtual lane marking. Namely, the miniature vehicle could aim at driving along a line in the Euclidean plane that is not marked on the test-bed floor.

We assume that each segment, s_i, has an index set, $L_{s_i} = \{1, 2, \ldots k\}$ that represents lanes. For each lane, $\ell \in L_{s_i}$, we define $in(\ell)$, $out(\ell) \subseteq \mathcal{S}$ as ℓ's entry, and respectively, exit segments in s_i, i.e., $in(\ell) \in in(s_i)$ and $out(\ell) \in out(s_i)$. For the case of $in(s_i) = \emptyset$ or $out(s_i) = \emptyset$, we define $in(\ell) = s_i$, and respectively, $out(\ell) = s_i$. Let $s_f \in in(s_i)$ and $s_t \in$

$out(s_i)$. We define $FT_{s_i}(s_f, s_t) = \{\ell \in L_{s_i} : in(\ell) = s_f \wedge out(\ell) = s_t\}$ as the index set of all lanes in segment s_i that go from s_f to s_t. Moreover, we define $OrderedLanes_{s_i} = \{\ell_{left}, \ldots, \ell_{right} \in L_{s_i}\}$ is an index list of lanes in segment s_i that is ordered from left most ℓ_{left} to the right most ℓ_{right}.

The vehicle may access information that is local to its current segment, $CurrentSegment_{v_i}$, and its lanes, $\ell \in L_{s_i}$. Namely, $in(s_i)$, $out(s_i)$, $in(\ell)$ and $out(\ell)$. When the vehicle current lane is *CurrentLane*, it can also query information about other vehicles in its proximity and nearby lanes. We define the set V as the set of system objects (vehicles, pedestrians, etc) and $V(v_i) \subseteq V$ as the set of object that vehicle $v_i \in V$ queries about. This information includes $v_j \in V(v_i)$ current location, $location_{v_j}$. Moreover, vehicle v_i can know if object (vehicle) $v_j \in V(v_i)$ is on its current trajectory. We require that $V(v_i)$ includes all the objects that are on the line of sight with v_i, i.e., any object, v_j, for which there is a straight line of bounded length to v_i that does not cross any object between them.

2.2 Maneuver control

We assume that the miniature vehicles are able to perform basic maneuvers, such as keeping their lanes ($KeepLane$) and changing their lanes ($RightChangeLane$ and $LeftChangeLane$), see [6]. The pseudo-code uses the primitive $ManeuverControl(ManeuverCommand)$ for issuing maneuver commands. For example, the vehicle will change its lane to the right one when using the command *RightChangeLane*, and will instruct the maneuver control to keep the current lane when using the $KeepLane$. The maneuver command *Stop* halts the vehicle.

3. DESIGN OUTLINE

The test-bed includes two subsystems: a Simulator and a Miniature Vehicle Platform (see Figure 2). Both these subsystems share several components. We first describe the two subsystems before looking into the main component, Motion Manager.

3.1 Simulator

The Simulator subsystem allows the vehicular system designer to develop and test new components referred as the Base Station, and the Miniature Vehicle (see Figure 3 left).

The Base Station includes the Experimenter Control that can send to the Motion Manager the key platform command, Ready, Go and Terminate, for initializing, starting, and respectively, terminating the experiments. The Base Station also includes the Experiment Manager, which sends to the Motion Manager all the information that is required for moving the miniature vehicles in the platform according to the experiment plan, i.e., Road Map, Route Plan and Virtual Lane Marking. The Base Station monitors and controls the experiment execution by periodically receiving the vehicle's positioning information.

The Miniature Vehicle controls the vehicle motion after receiving data from the onboard sensors and commands from the Base Station. The Motion Manager is the unit that controls the vehicle by issuing the commands Move and Halt for steering, and respectively, stopping the vehicle. These two commands are generated by the $ManeuverControl()$ primitive and allow each vehicle to take a sequence of maneuvers from source to destination along the traveling route.

3.2 Miniature Vehicle Platform

This subsystem allows prototype demonstration and testing of vehicular systems together with its components. During such experiments, the system logs its states for later diagnostics and playback in the simulator (see Figure 2). The logging information can be either stored by the vehicle processing units or transmitted on the fly. Future extensions of our design can also consider onboard fault injections.

In addition, one can validate the designer assumptions regarding the behavior of the human driver. Our implementation considers hand-held wireless devices (see Figure 3 right). Future extensions can use driver cockpits with multi-angle video streaming in addition to what looks like, sounds like and feels like emulation of the vehicles and their environment.

3.3 Motion Manager

This key component is mounted on the miniature vehicle and is in charge of receiving sensory information, which includes the vehicle location, and deciding which maneuver the vehicle should take. The maneuvers are controlled by the $ManeuverControl(ManeuverCommand)$ primitive (see Section 2). It is up to the Maneuver Strategy to decide on which $Command$ each miniature vehicle should take. The Maneuver Strategy must make sure that the miniature vehicles do not crash when traveling to their destinations. In order to do that, we use two mechanisms for crash avoidance and traffic light signaling. Next, we present these mechanisms before presenting the Maneuver Strategy itself.

Crash avoidance mechanisms for miniature vehicles. Gulliver considers miniature vehicles that are remotely controlled either by computers or human drivers. We assume the existence of crash avoidance mechanism that can assist in keeping people and equipment safe.

We consider a crash avoidance mechanism for vehicle v_i that warns the vehicle when it gets too close to objects that reside on its trajectory. Algorithm 1 refers to the function $CrashAvoidance()$. The function returns *Green* as long as v_i keeps a safe distance from any vehicle on v_i's trajectory. When the safe distance is violated, the function returns *Red*. See [23] for more details about $CrashAvoidance()$.

Traffic light mechanisms for miniature vehicles.

Traffic lights assure that at any time, no two vehicles from conflicting directions may enter the intersection, see [2]. In the proposed platform, traffic lights are important for assuring that the miniature vehicles do not crash when entering intersections. We assume the existence of traffic light mechanisms that periodically broadcast their signal state to all arriving vehicles. The arriving vehicle, v_i, caches the state encoded by the traffic light beacons. Vehicle v_i can query the traffic light state by executing the function $TrafficLight(CurrentSegment_{v_i})$.

3.3.1 Maneuver Strategy

This strategy allows the miniature vehicle to safely traverse the test-bed floor along the Route Plan. We present the maneuver strategy in Algorithm 1. The algorithm consists of a single action, which we assume to be fired periodically.

Before and after taking this update action, the algorithm tests whether or not vehicle v_i satisfies safety conditions, such as, crash avoidance requirements, the traffic light state, and the instructions of the prototype experimenter.

The action starts by testing that the vehicle has not reached its destination and that it follows its route plan correctly. Then, the algorithm deterministically selects a lane that leads vehicle v_i to its destination. Different cases of lane selections are considered: (1) No lane in the current segment can lead the vehicle to the next correct segment, (2) The current lane is the correct one, and (3) Vehicle v_i is required to perform a lane change maneuver in order to reach its destination. The latter case requires the algorithm to test safety conditions that is concerned with the distance between v_i and any other object, v_j, on its current lane or the lane to which it is moving into.

4. IMPLEMENTATION CHALLENGES

Recent advances in the field of mobile robots allow the ad hoc construction of affordable test-beds at your own parking lot. In fact, converting RF miniature vehicles to be a WiFi-Bot [25] controlled is a popular student project. We explain how the miniature vehicles can follow the lane markings, and we explore key technological challenges.

4.1 Following the lane marking

The miniature vehicles follow road markings in order to keep their lanes. All vehicle maneuvers, such as intersection crossing and lane changing, should follow the shortest possible path between the vehicle current and target positions, while keeping a safe distance from all other vehicles. Given the fact that we wish to use inexpensive miniature vehicles that move quickly in the test-bed, we have to endure a degree of unpredictability in the vehicle motion. Namely, different miniature vehicles respond differently to the driver's commands and the same miniature vehicle may behave differently under very similar conditions.

One challenge that the prototype experimenter faces is how miniature vehicles can drive along the lane markings. For example, the steering of the miniature vehicles might not allow driving along very sharp curves even at their slowest speed.

The implementation details that this paper considers are mainly focused on road designs that allow the maneuver strategies to follow the (virtual) road marking. The detailed report on our preliminary implementation efforts appears in [23]. We explain how the vehicles can follow the lane marking with the aid of mechanisms for lane detection and tracking [6, 22, 29].

4.2 Technological challenges

It is imperative to assure that the miniature vehicles do not crash due to the possible failure of system components, such as (clock) synchronization, communication and localization. Unlike the synchronization, communication and localization requirements in the case of fully deployed vehicular systems, some of the needed autonomic characteristics can be simplified in the case of Gulliver. Therefore, we explain the implementation of these primitives using a coordinating base-station (and consider that as the default implementation) before considering the possible extensions that have autonomous characteristics.

4.2.1 Clock synchronization

Synchronization mechanisms can simplify the design of the communication primitives and applications for vehicular systems. We base our implementation on the self-stabilizing and autonomous design of Herman and Zhang [11]. Their design is based on the converge-to-the-max technique in which all mobile motes adjust their internal clocks to the maximal time value that they have recently seen.

4.2.2 Communications

Vehicle to vehicle (V2V) communications are carried out via message passing (radio transmissions). Using these messages, the vehicles coordinate their "world" perception and decide on their joint actions. Existing ad hoc communication mechanisms, such as CSMA/CA, can implement V2V communication and facilitate clock synchronization mechanisms [16, 26]. Greater efficiency can be achieved when using a coordinating base station [10]. Recent developments consider media access control with higher predictability [19, 28]. This is another way to go toward an autonomous implementation.

4.2.3 Miniature vehicle localization

Positioning systems, such as GPS [12], have a wide range of applications in vehicular systems. Gulliver uses these positioning systems to inform the vehicles about the location of platform entities, like for example vehicles, road segments, intersections, etc.

We consider a design of the default implementation in which each vehicle sends beacons to peripheral receivers (anchors) that have known locations. The anchors then report the received beacons to a coordinating base station in which a localization algorithm [such as 4] is used for extracting location information before sending this information to the moving miniature vehicles. The above centralistic design allows us to use powerful processing at the coordinating base-station. As for future extensions, we plan to consider a more decentralized design that can lead towards scalable and autonomous implementation.

5. DISCUSSION

After summarizing the paper, we discuss some of the possible applications that can be studied by Gulliver, before drawing our conclusions.

5.1 Summary

We presented a design that allows one to conduct a variety of experiments in areas such as vehicle safety (e.g., crash prevention), energy (e.g., multi-lane vehicle platoon), to name a few. This inexpensive test-bed facilitates deployment of testing procedures. For example, protections against vehicle crashing of a miniature vehicle (500 g, 50 cm and 200 to $2k$ Euro) are simpler than a full-scale vehicle (1 ton, 5 m and $100k$ Euro).

Gulliver lies between computer simulation and full-scale vehicle models, and as such, it simplifies and reduces the costs of vehicular system prototyping and development. Note that the cost of each miniature vehicular unit is at least one or two order of magnitude less than a full-scale vehicular prototyping unit. By simplifying prototype development and demonstration processes, Gulliver opens up new research opportunities and promotes cross-fertilization between academic and industrial research.

5.2 Possible applications

One of the important issues that future vehicular systems will deal with is accident prevention. In [9], the authors

Algorithm 1: Maneuver Strategy for Vehicle v_i

Input $V(v_i)$: Set of vehicles that are on v_i's line of sight;
Input RoutePlan[]: Array of segments that defines v_i's route plan;
Input CurrentSegment: ID of v_i's current segment;
Input CurrentLane: ID of v_i's current lane;
Input $location_{v_i}$: Current geographical location of vehicle v_i;
Side effects $ManeuverControl(ManeuverCommand)$: Maneuver control primitive. The commands $KeepLane$, $Stop$, $RightChangeLane$, and $LeftChangeLane$ are used for keeping the current lane, stopping, changing to lane to the right, or respectively, to the left;
External TrafficLight(): Returns a set of $\langle Segment, Signal\rangle$, where $Segment \in in(CurrentSegment)$ is an incoming segment, and $Signal \in \{\mathsf{Red}, \mathsf{Orange}, \mathsf{Green}\}$ is a traffic light signal;
External CrashAvoidance(): Red and Green are the two Crash Avoidance signals;
External ExperimenterControl(): Ready, Go and Terminate are the three External Control commands;
Local Cursor: Iterator for v_i's $RoutePlan$. Initialized to the first element, 0;
Local TargetLane: When v_i is changing lane, this is the ID of the lane which v_i will use;
Local distance(locA, locB): Euclidean distance between locations $locA$ and $locB$;
Constant SafeDistance: Minimum required distance between any two vehicles that are moving in different lanes;
Alias PreviousSegment = in(CurrentLane), NextSegment = RoutePlan[Cursor+1]: IDs of previous, and respectively, next segments that for v_i to traverse;

Action : Maneuver Strategy;

/* Before moving, v_i checks if the experiment should be carried on, there is no crashing danger, and, in case v_i is approaching an intersection, the traffic light signals green */
Precondition : $((ExperimenterControl() = \mathsf{Go}) \wedge (CrashAvoidance() = \mathsf{Green})) \wedge ((in(CurrentSegment) = \emptyset) \vee (\langle PreviousSegment, \mathsf{Green}\rangle \in TrafficLight(CurrentSegment)))$;

/* Verify that the crashing avoidance policy was kept */
Postcondition : $CrashAvoidance() = \mathsf{Green}$;

begin
  if $Cursor < sizeof(RoutePlan) - 1$ **then**
    if $CurrentSegment = RoutePlan[Cursor]$ **then**
      $Candidates \leftarrow \arg\min_{|CurrentLane-\ell|}(\{\ell \in FT_{CurrentSegment}(PreviousSegment, NextSegment)\} \cup \{\infty\})$;
      /* Select the closest lanes that leads to the next segment, or ∞ if there are no such lanes */
      case $Candidates = \{\infty\}$: $ManeuverControl(Stop)$; /* Error: Vehicle v_i is moving on the wrong segment, because there is no lane that leads to the next segment */
      case $Candidates = \{CurrentLane\}$: $ManeuverControl(KeepLane)$; /* Current lane leads to the next segment */
      otherwise if $\exists CandidateLane \in Candidates : CurrentLane < TargetLane \wedge LaneChangeCrashAvoidance(CurrentLane, TargetLane) = Green$ **then** ; /* Move to a lane on the right or the left, after testing the safety conditions */
        $ManeuverControl(RightChangeLane)$;
      else
        if $LaneChangeCrashAvoidance(CurrentLane, TargetLane) = Green$ **then**
          $ManeuverControl(LeftChangeLane)$
    else if $CurrentSegment = NextSegment$ **then**
      $Cursor \leftarrow Cursor + 1$; /* Vehicle v_i traversed one segment and reached next segment of the route plan */
    else $ManeuverControl(Stop)$; /* Error: Vehicle v_i is moving on the wrong segment */
  $ManeuverControl(Stop)$; /* Vehicle reached the route's end */

Function $LaneChangeCrashAvoidance(CurrentLane, TargetLane)$
begin
  if $\{v_j \in V(v_i) : CurrentLane_{v_j} \in \{CurrentLane_{v_i}, TargetLane\} \wedge distance(location_{v_j}, location_{v_i}) < SafeDistance\} = \emptyset$ **then return** $Green$; /* Change lane only when there is no vehicle on v_i's trajectory */
  return Red

describe a self-organizing virtual traffic light (VTL). VTLs allow the ad hoc deployment of traffic lights in every road intersection. The authors of [9] use leader election mechanisms to allow a single vehicle to serve as the VTL server. It is up to this server to broadcast the VTL's status to arriving vehicles. These messages are then displayed to the drivers. The leader election criterion includes proximity considerations and requires agreement. Their concept is demonstrated via DIVERT [8] with sampled traffic information.

A more robust approach for VTL construction is presented in [3, 7, 27] using Virtual Node Layer (VNLayer). VNLayer is a programming abstraction in order to have virtual nodes emulated by physical nodes. They use the VNLayer for emulating the virtual traffic light. The implementation of traffic light in [3, 27] is deployed via a small set of HP iPAQ handheld computers that are mounted on slow moving robots.

Gulliver can examine advanced accident prevention services for future vehicular systems that are based on VNLayer, such as VTL and monitoring lane changes. Gulliver can demonstrate such concepts via extensive testing in a platform that has many miniature vehicles and has a logging-replaying mechanism, rather by simulation only or a small set of traffic scenarios.

5.3 Conclusions

The use of automotive technology can increase traffic throughput by improving vehicle density on roads. Safety requirements can be met by monitoring drivers' behaviors without necessarily building new road infrastructures. Thus far, many of the future vehicular systems are not allowed to operate on public roads due to the collision risks. Moreover, the lack of knowledge about the possible emerging patterns in large scale deployment requires additional testing. Gulliver provides an open source platform for demonstrating cyber-physical aspects of vehicular systems, making sure that their safety requirements are met and that their emerging patterns assure high traffic throughput.

Nowadays, there are 750 million motor vehicles in the world and the numbers are doubling every 30 years. Vehicular systems will enable vehicular interaction, cooperation and will be the first cyber-physical systems to reach the scale of millions of units. Currently, no safety-critical system comes close to this scale. Gulliver design is the first to facilitate the detailed investigation of the vehicle interaction and emerging patterns among hundreds and even thousands of units of a cyber-physical system.

Acknowledgment

The authors would like to thank Alireza Davoudian, Mohamed Mustafa, Philippas Tsigas, Johan Karlsson, Amir Tohidi, Ali Salehson, Jonas Sjöberg, Erik Ström, Fatemeh Ayat, Henk Wymeersch, Mahdieh Sadat Mir Hashemi, and Anna Nilsson-Ehle for many useful discussions.

References

[1] J. Aidemark, J. Vinter, P. Folkesson, and J. Karlsson. Goofi: Generic object-oriented fault injection tool. In *DSN*, page 668. IEEE Computer Society, 2003.

[2] P. Ammann. A safety kernel for traffic light control. Technical Report ISSE-TR-94-06, Department of Information and Software Engineering, George Mason University, 4400 University Drive MS 4A5, Fairfax, VA 22030-4444 USA, 1994.

[3] M. Brown, S. Gilbert, N. Lynch, C. Newport, T. Nolte, and M. Spindel. The virtual node layer: a programming abstraction for wireless sensor networks. *SIGBED Rev.*, 4:7–12, July 2007.

[4] M. A. Caceres, F. Penna, H. Wymeersch, and R. Garello. Hybrid gnss-terrestrial cooperative positioning via distributed belief propagation. In *Proceedings of the Global Communications Conference (GLOBECOM'10)*, pages 1–5. IEEE, 2010.

[5] K. Captain, A. Boghani, and D. Wormley. Analytical tire models for dynamic vehicle simulation. *Vehicle System Dynamics*, 8(1):1–32, 1979.

[6] W. Chee and M. Tomizuka. Lane change maneuver of automobiles for the intelligent vehicle and highway system (ivhs). In *American Control Conference*, volume 3, pages 3586 – 3587, June 1994.

[7] S. Dolev, S. Gilbert, L. Lahiani, N. A. Lynch, and T. Nolte. Timed virtual stationary automata for mobile networks. In J. H. Anderson, G. Prencipe, and R. Wattenhofer, editors, *Principles of Distributed Systems, 9th International Conference, OPODIS'05*, volume 3974 of *LNCS*, pages 130–145. Springer, 2005.

[8] R. Fernandes, P. M. d'Orey, and M. Ferreira. Divert for realistic simulation of heterogeneous vehicular networks. In *IEEE 7th International Conference on Mobile Adhoc and Sensor Systems, MASS'10*, pages 721–726. IEEE, 2010.

[9] M. Ferreira, R. Fernandes, H. Conceição, W. Viriyasitavat, and O. K. Tonguz. Self-organized traffic control. In *Proceedings of the seventh ACM international workshop on VehiculAr InterNETworking*, VANET '10, pages 85–90, New York, NY, USA, 2010. ACM.

[10] J. J. Garcia-Luna-Aceves and C. L. Fullmer. Floor acquisition multiple access (fama) in single-channel wireless networks. *Mobile Networks and Applications (MONET)*, 4:157–174, October 1999.

[11] T. Herman and C. Zhang. Best paper: Stabilizing clock synchronization for wireless sensor networks. In A. K. Datta and M. Gradinariu, editors, *Proceedings of Stabilization, Safety, and Security of Distributed Systems, 8th International Symposium (SSS'06)*, volume 4280 of *LNCS*, pages 335–349. Springer, 2006.

[12] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins. *Global Positioning System: Theory and Practice*. Springer, Wien (Austria), 1993.

[13] M.-C. Hsueh, T. K. Tsai, and R. K. Iyer. Fault injection techniques and tools. *IEEE Computer*, 30(4):75–82, 1997.

[14] J. Kim, V. Sridhara, and S. Bohacek. Realistic mobility simulation of urban mesh networks. *Ad Hoc Networks*, 7(2):411–430, 2009.

[15] D. Krajzewicz, G. Hertkorn, C. Rössel, and P. Wagner. Sumo (simulation of urban mobility). In *Proc. of the 4th Middle East Symposium on Simulation and Modelling*, pages 183–187, 2002.

[16] F. Kuhn, C. Lenzen, T. Locher, and R. Oshman. Optimal gradient clock synchronization in dynamic networks. In A. W. Richa and R. Guerraoui, editors, *Proceedings of the 29th Annual ACM Symposium on Principles of Distributed Computing, PODC'10*, pages 430–439. ACM, 2010.

[17] C. Lee, J. Kim, J. Hallquist, Y. Zhang, and A. Farahani. Validation of a FEA tire model for vehicle dynamic analysis and full vehicle real time proving ground simulations. Technical report, Society of Automotive Engineers, 400 Commonwealth Dr, Warrendale, PA, 15096, USA„ 1997.

[18] E. A. Lee. Cyber physical systems: Design challenges. In *11th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'08)*, pages 363–369. IEEE Computer Society, 2008.

[19] P. Leone, M. Papatriantafilou, E. M. Schiller, and G. Zhu. Chameleon-mac: Adaptive and self-* algorithms for media access control in mobile ad hoc networks. In S. Dolev, J. A. Cobb, M. J. Fischer, and M. Yung, editors, *Proceedings of Stabilization, Safety, and Security of Distributed Systems - 12th International Symposium, SSS'10*, volume 6366 of *Lecture Notes in Computer Science*, pages 468–488. Springer, 2010.

[20] M. D. Letherwood and D. D. Gunter. Ground vehicle modeling and simulation of military vehicles using high performance computing. *Parallel Computing*, 27(1-2):109–140, 2001.

[21] P. Levis, N. Lee, M. Welsh, and D. E. Culler. Tossim: accurate and scalable simulation of entire tinyos applications. In I. F. Akyildiz, D. Estrin, D. E. Culler, and M. B. Srivastava, editors, *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, SenSys'03*, pages 126–137. ACM, 2003.

[22] J. C. McCall and M. M. Trivedi. An integrated, robust approach to lane marking detection and lane tracking. In *Intelligent Vehicles Symposium, 2004 IEEE*, pages 533 – 537, june 2004.

[23] M. Pahlavan, M. Papatriantafilou, and E. M. Schiller. Gulliver: A testbed for developing, demonstrating and prototyping vehicular systems. Technical Report 2011:14, ISSN 1652-926X, Computer Science and Engineering, Chalmers University of Technology, May 2011.

[24] C. Pinart, P. Sanz, I. Lequerica, D. García, I. Barona, and D. Sánchez-Aparisi. DRIVE: a reconfigurable testbed for advanced vehicular services and communications. In *Proceedings of the 4th International Conference on Testbeds and research infrastructures for the development of networks & communities*, pages 1–8. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[25] Robosoft. URL. http://www.robosoft.fr/eng/.

[26] P. Sommer and R. Wattenhofer. Gradient clock synchronization in wireless sensor networks. In *Proceedings of the 8th International Conference on Information Processing in Sensor Networks (IPSN'09)*, pages 37–48. ACM, 2009.

[27] M. Spindel. Simulation and evaluation of the reactive virtual node layer. Master's thesis, Computer Science and Artificial Intelligence Laboratory, 2008.

[28] S. Viqar and J. L. Welch. Deterministic collision free communication despite continuous motion. In S. Dolev, editor, *Algorithmic Aspects of Wireless Sensor Networks, 5th International Workshop (ALGOSENSORS'09). Revised Selected Papers.*, volume 5804 of *Lecture Notes in Computer Science* pages 218–229. Springer, 2009.

[29] Y. Wang, E. K. Teoh, and D. Shen. Lane detection and tracking using b-snake. *Image Vision Comput.* 22(4):269–280, 2004.

[30] Y. Zhang, A. Tang, T. Palmer, and C. Hazard. Virtual Proving Ground-an integrated technology for full vehicle analysis and simulation. *International journal of vehicle design*, 21(4):450–470, 1999.