A Parallel Autonomy Research Platform

Felix Naser[1], David Dorhout[1], Stephen Proulx[1], Scott Drew Pendleton[2], Hans Andersen[2], Wilko Schwarting[1], Liam Paull[1], Javier Alonso-Mora[1], Marcelo H. Ang Jr.[2], Sertac Karaman[1], Russ Tedrake[1], John Leonard[1], Daniela Rus[1]

***Abstract*— We present the development of a full-scale "parallel autonomy" research platform including software and hardware. In the parallel autonomy paradigm, the control of the vehicle is shared; the human is still in control of the vehicle, but the autonomy system is always running in the background to prevent accidents. Our holistic approach includes: (1) a drive-by-wire conversion method only based on reverse engineering, (2) mounting of relatively inexpensive sensors onto the vehicle, (3) implementation of a localization and mapping system, (4) obstacle detection and (5) a shared controller as well as (6) integration with an advanced autonomy simulation system (Drake) for rapid development and testing. The system can operate in three modes: (a) manual driving, (b) full autonomy, where the system is in complete control of the vehicle and (c) parallel autonomy, where the shared controller is implemented. We present results from extensive testing of a full-scale vehicle on closed tracks that demonstrate these capabilities.**

I. INTRODUCTION

Autonomous driving systems offer higher safety, fuel economy, mobility and comfort. The US National Highway Traffic Safety Administration (NHTSA) has classified vehicles into one of five levels (0-4) of automation [1]. The autonomous driving problem can be considered solved once we have developed a vehicle at level 4, since that would require no input or oversight from the user. However, opinions diverge on the best path to arrive at this level. For example, Google Inc. [2], deemed that level 3 (human takes over control from the autonomous systems when the situation is too difficult) is not feasible since the driver is responsible to oversee the system, however may become complacent. Humans are proficient in active control tasks, but are not well-suited to monitoring tasks. Once the system works fairly reliably the user quickly begins to trust that it will always work. As a result, Google is developing a car directly for level 4. Tesla Motors, on the other hand, has released to market a system capable of level 3 autonomy and we have seen the first fatal accident resulting from complacency of the driver [3]. We propose an alternative approach to arrive at level 4, which is focused primarily on solving the *safety* problem associated with autonomous driving first. We have termed this level of autonomy "parallel autonomy", or "level 2.99", or "guardian angel". In this case, the human is still in control of the vehicle, but a full autonomy system is always running in the background and is responsible for preventing the human from causing an accident by minimally adjusting the commanded acceleration and speed. This approach has the added advantage that, if the system becomes unsure of the correct course of action, it can always choose to *do nothing* and consequently should be no worse than the human driver alone.

*This work was supported in part by the Singapore-MIT Alliance for Research and Technology (SMART) Future Urban Mobility Group (FM). The Toyota Research Institute (TRI) provided funds to assist the authors with their research, but this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity.

[1]F. Naser, D. Dorhout, S. Proulx, W. Schwarting, L. Paull, J. Alonso-Mora, S. Karaman, R. Tedrake, J. Leonard and D. Rus are with the Computer Science and Artificial Intelligence Laboratory (CSAIL), Massachusetts Institute of Technology (MIT), Cambridge, MA 02139, USA. {fnaser, ddorhout, wilkos, sertac, jleonard}@mit.edu, {steve, lpaull, jalonsom, russt, rus}@csail.mit.edu

[2]S. Pendleton, H. Andersen and Marcelo H. Ang Jr. are with the National University of Singapore (NUS). {scott.pendleton01, hans.andersen, mpeangh}@u.nus.edu

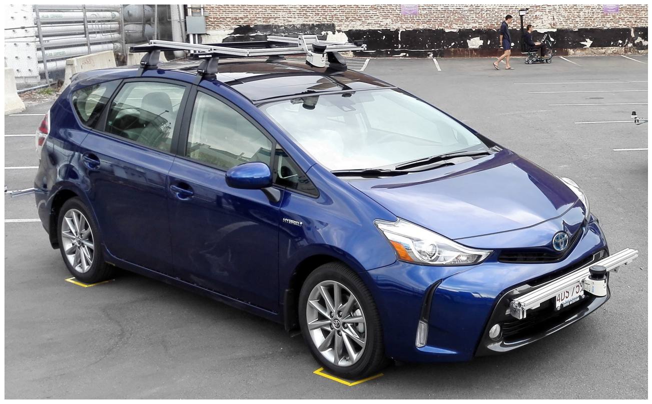


Fig. 1. The converted Toyota 2015 Prius V, a parallel autonomy research platform. The car is outfitted with sensors to enable fully autonomous driving, a drive-by-wire conversion and low-level controllers which are designed to fuse the desired input from the human with the autonomy software control output to provide safe acceleration and speed.

Existing work in both the academic and private sectors have attempted related approaches.

In the academic setting, the work of Anderson et. al. [4] demonstrated a constraint-based approach to shared control on an all-terrain vehicle. In [5], the shared control problem is formulated through envelopes and model predictive control, demonstrated on a research buggy. These works impressively presented shared control between the human and the autonomy system, but are not demonstrated on full-scale commercial vehicle hardware. The work of Gray et. al. also demonstrates a shared control [6], [7], with a focus on explicitly modeling the driver, but the hardware platform requires proprietary CAN bus information for actuation. Carnegie Mellon University's (CMU) autonomous driving

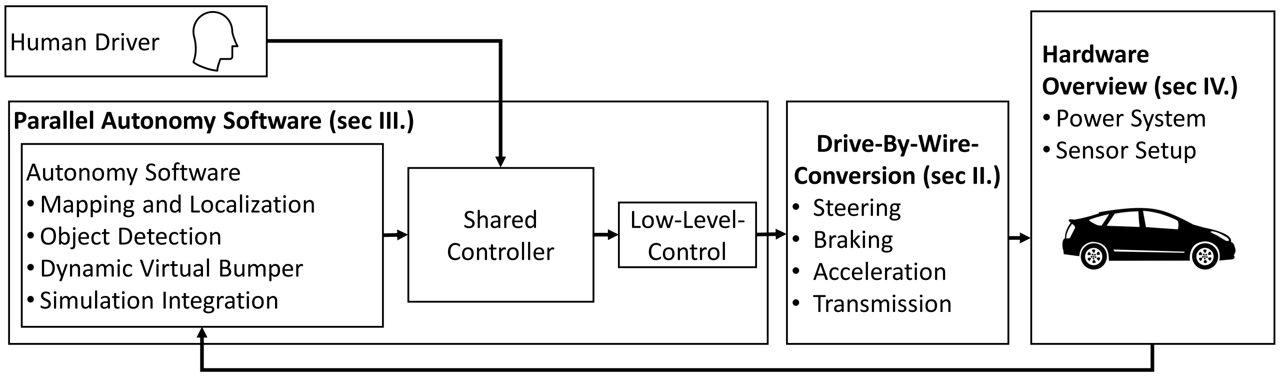


Fig. 2. Parallel autonomy research platform overview.

platform presented in [8] is capable of a wide range of autonomous and intelligent behaviors, but our drive-by-wire conversion method enables parallel autonomy and requires less modifications to control the vehicle. Although the work presented from ETH in [9] is using close-to-market sensors, but its actuation system depends on proprietary CAN bus information. The Autoware project by Kato et. al. [10] released an open autonomy software stack that relies on a third party converted car.

In the consumer realm, car manufactures such as BMW [11], Mercedes [12] and Tesla [13] have all designed systems for level 3. Google even started to build prototypes on their own for level 4 [2]. We built a full-scale research platform for level 2.99.

The main contributions of this paper are:

- a holistic approach to build a full-scale parallel autonomy research platform (Fig. 2),
- a detailed description of a reliable, fast, flexible and parallel autonomy enabling drive-by-wire conversion method based only on reverse engineering and without added external actuators,
- an experimental demonstration of the parallel autonomy framework on a full-scale vehicle.

Fig. 2 shows the overall system architecture. In Section III we explain each module of the parallel autonomy software stack. We have integrated our previous work as the level 4 system which always runs in the background, implemented a shared controller and interfaced the control output with the car. In section II and IV we present our vehicle platform and how we converted it. In Section V we show parallel autonomy experimental results. Finally we conclude the paper and propose future works in Section VI.

II. DRIVE-BY-WIRE-CONVERSION

We define a drive-by-wire conversion as a retrofit for an off-the-shelf car which enables control of steering, acceleration, gear selection and braking with software commands. There are four main drive-by-wire conversion approaches:

- CAN bus write access (e.g. Nvidia [14], BMW [11], ETH [9]): Requires propriety information about the CAN messages and/or modified Electronic Control Units (ECUs) from the Original Equipment Manufacturer (OEM).
- Sensor-signal-spoofing (e.g. DARPA Urban Challenge 2007: Princeton [15], Stanford Junior [16], VictorTango Odin [17]): After finding the primary control sensors the signals are spoofed.
- Motor-actuation (e.g. DARPA Urban Challenge 2007: CMU Boss [18], MIT [19], SMART SCOT): There are companies that provide motor based solutions to actuate a car e.g. Electronic Mobility Controls, the risk of low actuation speed is high.
- Hybrid approach (e.g. CMU [8]): Motors and sensor-signal-spoofing are used to actuate the vehicle.

We extended the sensor-signal-spoofing approach to enable not only full, but also parallel autonomy and present a reliable, fast and flexible method to add parallel and full autonomy functionality to an off-the-shelf car.

The Toyota 2015 Prius V, such as many other modern vehicles, is enabling this approach, because it uses extensive driver assistance and drive-by-wire systems for steering, braking, acceleration and transmission. These systems use sensors to detect the driver's inputs which are then read and interpreted by ECUs throughout the vehicle. The specific ECU then directly communicates with the appropriate actuation system. In normal operation, the vehicle's braking, acceleration and transmission systems are controlled purely by electronic signals and do not use any of the driver's force. Likewise, the majority of the effort to actuate the steering wheel is accomplished using a DC motor mounted to the steering column shaft and controlled by the ECU, not the driver. For this project, the preexisting drive-by-wire systems were reverse engineered and used to create a testing platform for parallel autonomy.

The parallel autonomy paradigm requires seamless transition between human and computer control of the vehicle. This can be achieved when the car operates and appears like a normal car to the human driver while the on-board computer system is able to read the driver's inputs and override, either to support or prevent unsafe action via a shared controller. The physical modifications are limited to splicing on extensions to the cables connecting the sensors to their associated ECUs so that they could connect to our interface board. No hardware was attached to the pedals, steering wheel, or transmission lever.

A. *Switching modes to enable parallel autonomy*

A system of mechanical relays was designed to allow the car to switch between the three operating modes shown in Fig. 3: (a) manual, (b) observational and (c) computer control. When the car is in "manual" operation, the relays are not energized and are in their normally closed position, allowing the signals from each sensor to pass through its relay and to the corresponding ECU. In this position, the circuit connecting the sensor to its ECU is identical to that of an unmodified car (Fig. 3-(a)). In "observational" mode, a relay is energized to pass the signals through, allowing the interface board to observe the communication between the sensors and the ECUs (Fig. 3-(b)). This allows the system to witness what the driver is communicating to the car. In "computer control" mode, the interface board communicates directly with the car's ECU while also receiving the signals from the different system's sensors (Fig. 3-(c)). This is accomplished by energizing the set of relays that redirect the car's sensors to the interface board and allow the interface board to masquerade as each sensor to the appropriate ECU. To the ECUs it appears that there is a driver turning the wheel, applying the brake, pressing the accelerator, or shifting the transmission lever when in reality it is the interface board spoofing those signals.

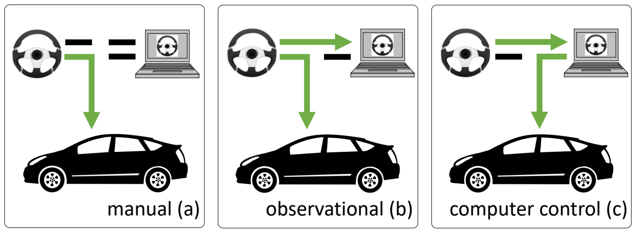


Fig. 3. Three modes of actuation: (a) manual, (b) observational and (c) computer control. Once operations are in computer control, we have again three options: (i) passing human readings unmodified through, (ii) passing autonomous system outputs through disregarding human readings and (iii) conditionally passing human readings through managed by the shared controller (Section III-E).

B. *Steering*

The 2015 Prius V Electric Power Steering (EPS) system uses a torque sensor and DC motor located within the steering column assembly to assist the driver with turning the steering wheel (Fig. 4). The torque sensor contains two sensors (T1 and T2) that indirectly detect the twist of a torsion bar within the steering column. These sensors then output two voltages (VT1 and VT2) to the EPS ECU which communicates the direction and speed of rotation of the steering wheel. The EPS ECU then uses that information along with the speed of the vehicle to determine how much effort the DC motor should apply to the steering wheel column to make it turn the correct amount at the appropriate speed.

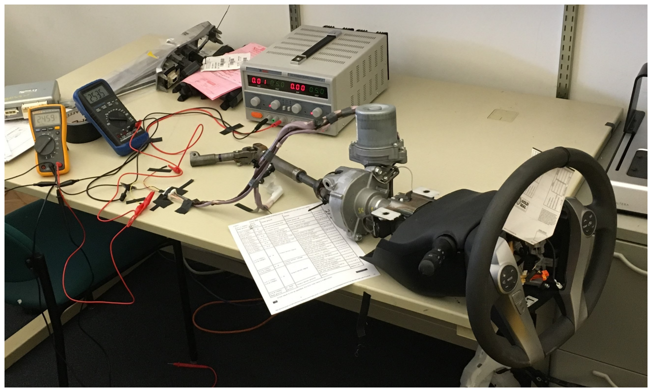


Fig. 4. Steering column setup to read torque sensor signals. We used salvaged parts from an identical vehicle to test the actuation method first.

For example, when the steering wheel is turned to the left, VT2 increases and VT1 decreases proportionally to each other (Fig. 5), then when the steering wheel is turned to the right, VT1 increases and VT2 decreases proportionally (Fig. 5). The manner in which VT1 and VT2 reflect each other is used by the EPS ECU to detect if there is a fault in the circuit such as if VT1 increased to 3.5V while VT2 remained at 2.5V. This event would cause the EPS ECU to reduce or eliminate the power steering motor output, allowing the driver to continue safely maneuvering the car manually (albeit with greater effort) via the direct mechanical linkage of the steering wheel column to the rest of the vehicle.

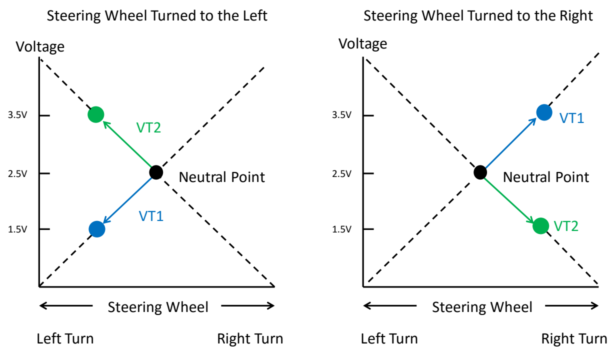


Fig. 5. This figure shows how the torque sensor readings are related to each other when steering wheel is turned left or right.

C. *Braking*

The braking system of the 2015 Prius V is collectively known as the "driver assisted braking system" and during normal operation it supplies all of the braking force applied to the wheels. It consists of several subsystems related to braking including the Anti-lock Brake System (ABS), Electronic Brake Force Distribution (EBD), Traction Control (TR(A)C), etc.. The vehicle receives braking information input from the driver through three sets of sensors, the master cylinder pressure sensor, the brake pedal stroke sensor and the brake light sensor.

1) Master Cylinder Pressure Sensor: In the 2015 Prius V, the master cylinder pressure sensor is located within the actuator portion which is in between the Skid Control (SK) ECU and the hydraulic brake booster. When the driver presses the brake pedal, the pressure of the hydraulic fluid within the hydraulic brake booster increases, which is detected by the master cylinder pressure sensor. It is then converted to a single electrical voltage between 0-5V which is sent to the SK ECU.

2) Brake Pedal Stroke Sensor: The brake pedal stroke sensor is attached to the pivot shaft of the brake pedal and communicates its rotational position to the SK ECU using two voltages (SK1 and SK2) between 0-5V. SK1 and SK2 both increase and decreases proportionally. This feature is again used by the SK ECU to detect if there is a fault in the circuit in the same way that the two sensor values from the steering torque sensor are compared to detect a fault.

3) Brake Light Sensor: The brake light sensor is a normally closed switch attached to the brake pedal assembly near the pivot point. In its normal state, the brake pedal assembly depresses the switch interrupting the circuit. The brake light switch assembly varies between models depending on if the Prius is equipped with dynamic cruise control or not. The switch appears to operate as a simple, binary switch that when released, powers the brake lights and communicates with the ignition and transmission system, letting them know when the brake pedal is being depressed.

4) Master Cylinder Pressure and Brake Pedal Stroke Sensor Dependencies: Both the master cylinder pressure sensor and the brake pedal stroke sensor are required for actuating the brakes. Those sensor values are related to each other and monitored by the pre-collision system. We recorded data points with Techstream [20] to approximate the underlying function. A value mismatch will result in a malfunction being detected and the braking system will switch to manual control.

D. Acceleration

The accelerator pedal uses two identical sensors located at the pedal's pivot point that detect the rotational angle of the pedal and outputs two 0-5V signals (VPA1 and VPA2) to the Power Management Control (PMC) ECU. VPA1 and VPA2 increase and decrease in a parallel fashion with VPA1 always being 0.8V greater than VPA2. This is different than what was observed in the steering and stroke sensor systems. A mismatch between VPA1 and VPA2 will result in a malfunction being detected by the PMC ECU.

E. Automatic Transmission

The Power Management Control (PMC) ECU uses a "select" and a "shift" sensor to determine the position of the shift lever (Fig. 6). Each sensor contains a main and sub circuit that are identical and when operating properly, they output nearly identical voltages as redundancy for safety checks.

1) Select Sensor: The "select sensor" is composed of two magneto resistive sensors, which output voltages (VSX3, VSX4) between 0-5V that the PMC ECU uses to determine the shift lever's horizontal position in the panel. As the shift lever is moved from the right to the left, VSX3 and VSX4 both increase from around 1.4V to 3.5V at the same time.

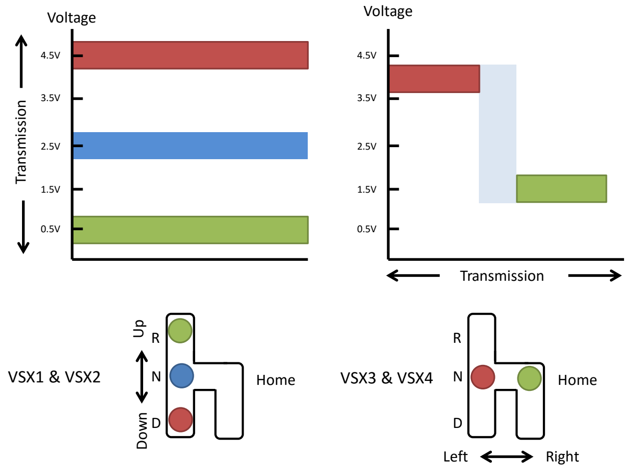


Fig. 6. Schematic of how the shift and select sensor reflect shift lever movements.

2) Shift Sensor: The "shift" sensor uses two hall sensors that output a voltage (VSX1, VSX2) between 0-5V that the PMC ECU uses to determine the shift lever's vertical position in the panel. As the shift lever vertically moves up, the voltage that VSX1 and VSX2 output decreases from around 2.5V to 0.5V. The ECU places the voltage inside three buckets to determine if the lever is in "drive" (high voltage), "neutral" (middle voltage), or "reverse" (low voltage).

F. Override Systems

There are three systems that act as layers to prevent unintended actions by the vehicle: The vehicle's native fault detection system, an external watch dog circuit and manual overrides. These three systems work together using independent and unique methods for error detection that each results in the vehicle being placed in a "safe" mode and allows the safety drive to regain control of the vehicle preventing a single point of failure.

1) Native Fault Detection System: The 2015 Prius V is equipped with fault detection circuits and programs that monitor the driver assistance programs including the systems mentioned above. The driver assistance systems have each been designed so that when there is a fault, the systems fails into a safe mode that allows the driver to safely maneuver the vehicle, e.g. make use of the mechanical linkage for the steering wheel or the hydraulic brake system.

2) Watch Dog: The Watch Dog circuit monitors the cable connections and the heartbeat of both the interface board micro-controller and the on-board vehicle control computer. It has its own independent micro-controller and is completely electrically isolated from the vehicle until it starts to monitor the cables. During power-up the watch dog circuit checks to see if all of the cables connecting the interface board to the car are connected. If the cables are all present, the watch dog then turns on the power to the interface board and begins monitoring both the interface board and the on-board control computer to ensure that the programs are operating correctly by observing a “heartbeat” from each system. If the watch dog determines an error, it cuts the power to the interface board. This places the vehicle into manual mode, allowing the safety driver to safely control the vehicle and sends an error message to the vehicle’s control computer. The watch dog also communicates with the driver via a dashboard light to indicate in which mode the vehicle is.

3) Manual Override: There are three types of manual override that return the vehicle to manual operation. The first override system is only used when the car is driving fully autonomously. It monitors the sensors from the vehicle’s steering wheel, brake and accelerator pedals. If the system detects an input from the driver, such as pressing the brake, the software then switches the vehicle back to manual control. This allows the driver to naturally and intuitively assume control of the vehicle as needed. The second manual override system functions in a similar software-based method but uses a button as input instead of the steering wheel or pedals. The third manual override system uses a button that physically turns off the power to the interface board, resulting in the vehicle returning to manual mode.

III. PARALLEL AUTONOMY

Enabling parallel autonomy requires a fully functional autonomous system to act as one of the two inputs to the shared controller as shown in Fig. 7. In this section, we will briefly describe the existing autonomy system that we use, which is based on our previous work as part of the Singapore-MIT Alliance for Research and Technology (SMART) [21] and open source tools such as those built into the Robotic Operating System (ROS) [22].

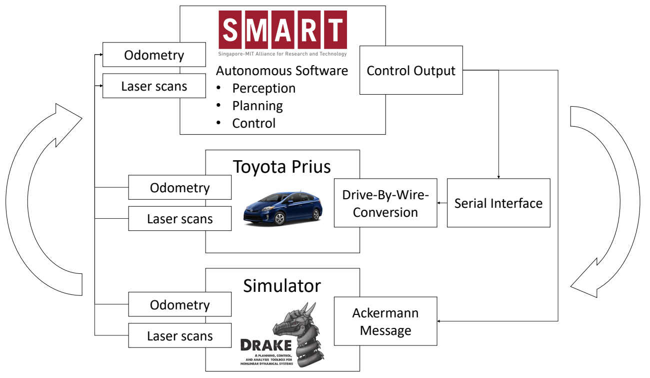


Fig. 7. The parallel autonomy system uses the core libraries developed as part of the SMART project. The software is architected to be easily configurable between real vehicle hardware and the Drake simulation environment for rapid development and testing.

A. Mapping and Localization

The localization and mapping system is based on our previous work. First, the vehicle is manually driven around the environment and sensor readings are recorded. Then pose Simultaneous Localization And Mapping (SLAM) is used to build a consistent map of the environment [23]. The localization is performed based on this built map. Synthetic LiDAR, a specific sensor model, is used to perform Adaptive Monte-Carlo Localization [24]. The synthetic LiDAR makes use of the normals in a 3D rolling window as the main features that provide a unique fingerprint of the environment.

B. Moving Object Detection

To ensure safe navigation of the autonomous vehicle, a moving object recognition algorithm was developed to detect and recognize other human agents in the shared environment [25]. The algorithm utilizes the spatial-temporal features of object clusters extracted from a planar LiDAR and performs object recognition using a supervised learning method of Support Vector Machine (SVM). Once moving pedestrians are recognized, their motion information (speed and direction) is calculated based on their centroid positions from consecutive measurements in the spatial-temporal clusters. The positions and the speeds of the recognized pedestrians are then passed on to the Dynamic Virtual Bumper module for vehicle speed control.

C. Dynamic Virtual Bumper

A Dynamic Virtual Bumper (DVB) [21] is used to generate the advisory speed for the vehicle’s safe navigation in the presence of both static and moving obstacles. The DVB is defined as a tube zone with its center line as the vehicles local path and its width w_t and height h_t as linear functions to the vehicle’s speed v_t at time t:

$$\begin{aligned} w_t &= w_0 + \alpha * v_t^2 \\ h_t &= h_0 + \beta * v_t^2 \end{aligned} \tag{1}$$

where w_0 and h_0 are the static buffers and α and β are the coefficients that the side lengths grow together with the vehicle’s speed, which reflects the bumper’s dynamic nature. LiDARs are used to detect obstacles in the vicinity. When an obstacle o_i enters DVB, the vehicle will generate an advisory speed of the new desired DVB, whose boundary is marked by the position of the obstacle.

D. Simulation

In order to test new algorithms first in a simulator and then in the real car to ensure safety and reduce development time, we integrated our software stack in Drake [26]. Drake is a very advanced and flexible planning, control and analysis toolbox for nonlinear systems that has been released under open source license. As shown in Fig. 7 we have standardized the interface between the simulator and the real vehicle hardware to ease switching between the two.

The car localizing itself and driving autonomously in the simulator is shown in Fig. 8. To further homogenize the user interface between the steering wheel and the car we have interfaced a gaming steering wheel and pedals to the simulator (Fig. 9).

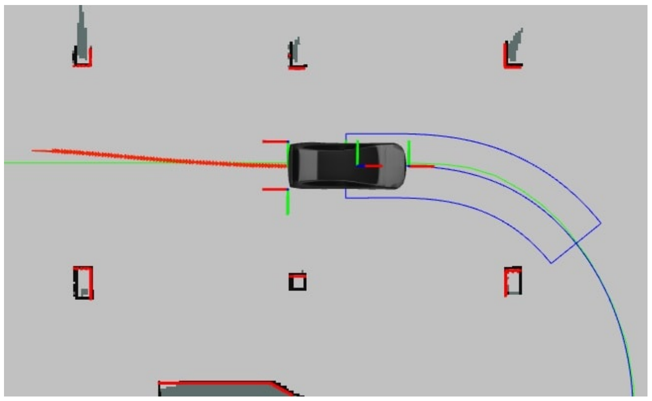


Fig. 8. Software stack running in Drake: Grey map in background, red arrow line visualizes odometry messages over time, red points on obstacles are the LiDAR output, the blue line is the control path from the pure-pursuit controller, the curved blue rectangle is the DVB and the green line the predefined path.

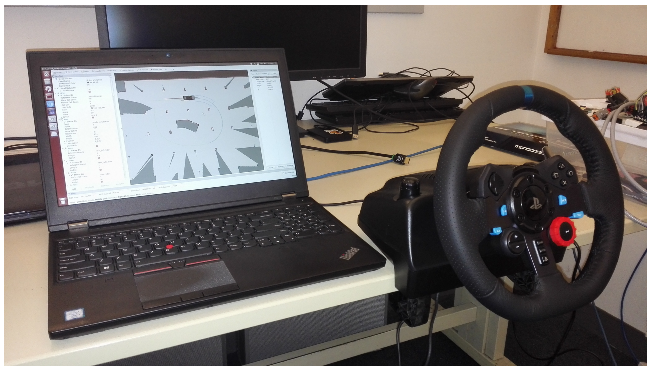


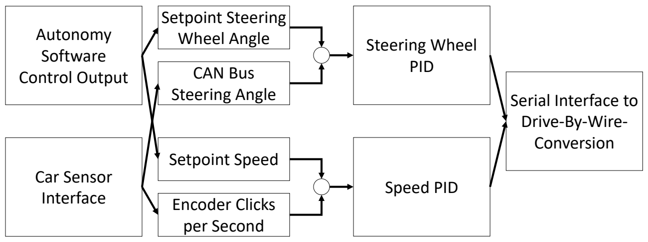
Fig. 9. A steering wheel and pedals (not shown) are interfaced to provide a standard and uniform user interface to test parallel autonomy with human inputs in Drake.

E. Shared Controller

In autonomous mode, we have a pure-pursuit steering controller [27] to follow a predefined path. In the presence of a road blockage, e.g. by pedestrians, the vehicle waits until the path is clear before proceeding. We extended our previous work with a basic shared controller for speed and steering. The shared controller takes human driver inputs such as steering wheel angle and pedal statuses and compares them to the autonomous system output. If the speed for the current path is above the DVB advised speed v_{max} or if the user's selected steering wheel angle deviates from the autonomous system recommendation by more than a certain threshold, the system takes over until it is again steering towards the path and below the calculated safe speed. This works when the system is in computer control mode (Fig. 3-(c)).

F. Low-Level-Control

We decided to minimize logic and computing tasks on the micro-controller level because it is hard to debug and error-prone. We integrated a standard ROS PID controller that takes control set points and current system states as inputs and outputs voltage values to spoof the sensor signals. This enables us to send a steering angle and speed in order to control the vehicle. A serial interface communicates between the micro-controller and ROS, publishing the sensor readings as custom messages (encoder clicks, brake position, steering wheel angle and accelerator voltages).



Autonomy Software Control Output
Car Sensor Interface
Setpoint Steering Wheel Angle
CAN Bus Steering Angle
Setpoint Speed
Encoder Clicks per Second
Steering Wheel PID
Speed PID
Serial Interface to Drive-By-Wire-Conversion

Fig. 10. Low-Level-Control Interface Overview: One PID controller regulates the speed and the other one the steering wheel angle.

To close the control loop for the steering wheel angle we read the CAN bus message, which contains the current steering wheel angle. For that we integrated an open source library for communicating with the vehicle network tools developed by Intrepid Control System in ROS and used the only reverse engineering based work presented in [28] to decode the CAN bus steering wheel angle message (Fig. 11).

$$\phi = \begin{cases} (b_0 * 16^2 + b_1 - \sigma) * \frac{360}{240} \\ \text{if } b_0 * 16^2 + b_1 - \sigma < 2048 \\ (b_0 * 16^2 + b_1 - \sigma - 4096) * \frac{360}{240} \\ \text{if } b_0 * 16^2 + b_1 - \sigma \geq 2048 \end{cases} \tag{2}$$

Fig. 11. Equation to compute the steering wheel angle, where b_0 and b_1 are the values of the first two bytes of the CAN message with the ID $0x25$ and σ is the offset from the zero position.

On the micro-controller level we only read and write the sensor signals depending on the mode the car is in (Fig. 3). A multi-core architecture enables fast reading of the sensor signals and publishing in to the remainder of the software system.

IV. HARDWARE OVERVIEW

The Toyota 2015 Prius V is our vehicle base platform and we retrofitted necessary sensing, computing and power systems to enable parallel and autonomous driving. Fig. 12 shows the mounted sensors and Section IV-A the power system.

A. Power System

All system electronics utilize the vehicle's auxiliary 12VDC battery as the source of power. This low voltage battery sources its power from the vehicle's high voltage source via a DC-DC voltage converter. Using DC-DC converters and a DC-AC inverter, the low voltage battery's 12VDC is converted to a range of voltages appropriate to power requirements of the system's components. Each leg of the power system has overcurrent protection and power distribution. Additionally, sensitive components have individual overcurrent protection, per manufacturer specification.

B. Sensor Setup

Table I lists the sensors we integrated. We leveraged from open source drivers from the ROS community to interface with our software system.

Two incremental encoders are mounted on the vehicle, one at each rear wheel. An Inertial Measurement Unit (IMU) is mounted inside the car along the center axis to provide attitude and heading of the vehicle. A Global Positioning System (GPS) unit is mounted on the roof. Encoder, GPS and IMU readings are fused to provide the vehicle's odometry information in six degrees-of-freedom using an unscented Kalman filter [29].

Four 2D LiDARs and a webcam are used to sense the environment, where we use the webcam currently only for a better scene understanding after recording data sets. One SICK LMS151 LiDAR is mounted at a tilted down angle onto the front of the vehicle roof (ca. 15 deg) for mapping and localization. A second SICK LMS151 is mounted horizontally in the lower front of the vehicle for obstacle detection. Two SICK TiM551 LiDARs are mounted at the rear corners of the vehicle to provide all around obstacle detection. The sensor setup is shown in Fig. 12.

As with the drive-by-wire conversion method, we employed a minimally invasive and lean mounting of the sensors to the car. We achieved this by using for rear and front LiDAR mountings the existing towing holes and for the GPS and top LiDAR an off-the-shelf roof-rack as a mounting base. Only for the two encoders we had to drill holes in the chassis.

TABLE I

TOYOTA 2015 PRIUS V SENSOR PART LIST

Sensor	Part Number
2 Front LiDARs	SICK-LMS151
2 Rear LiDARs	SICK-TiM551
2 Wheel Encoders	TR1-U1R4-100NV1RHV-F00
IMU	MicroStrain 3DM-GX4-25
USB Camera	Logitech HD Pro Webcam C920
GPS	Garmin-18x-GPS-Navigator-Unit
CAN bus Reader	neoVI RED
Computer	ThinkPad P50

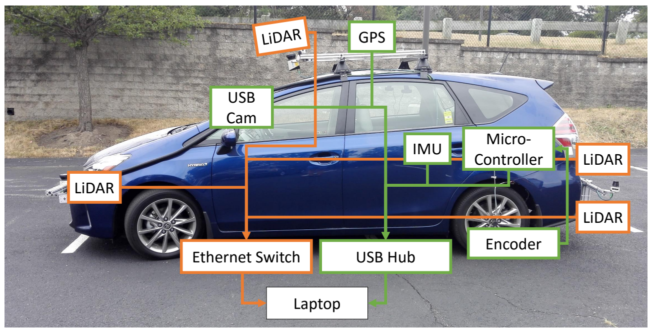


Fig. 12. We equipped the parallel autonomy research platform with relatively inexpensive sensors (Table I).

V. EXPERIMENTAL RESULTS

We have shown the flexibility and modularity of our software stack by reproducing the SMART autonomous driving capabilities [21] on a new hardware platform and with a different drive-by-wire conversion method[1]. We have ensured high reliability and safety with intensive test drives on closed loop test fields. We also extended our previous work by implementing a basic shared controller as explained in Section III-E, enabling shared control for speed and steering wheel angle at the same time. This video shows that the driver can't leave the path and can't go above the safe speed calculated by the parallel autonomy system[2].

A. Shared controller applied to speed

As a first step towards parallel autonomy functionality, we tested a "safe speed function." This system is more advanced than a standard "emergency brake assistant" since the autonomous system is using the dynamics of the vehicle and the current reference path to determine the safe maximum speed. If the human driver goes faster than the control output is suggesting, the systems takes over until the speed is again in the safe speed zone and then gives back control to the driver (see Fig 13). This works similarly when an obstacle is obstructing the path and the human does not pay attention, because then the control output would be either 0 m/s or close to 0 m/s depending on how the obstacle enters the DVB explained in Section III-C.

In the attached video the clicking sound from the microcontroller relay marks the takeover points in the plot. In future work, to reduce the number of takeovers we will implement a hysteresis filter. Nevertheless, we have demonstrated that with our chosen drive-by-wire conversion method we can read the human input at all times and can if needed seamlessly takeover control. The inertia of the system slows control effects down, e.g. after the first takeover point the autonomy systems applies brakes, but it takes around 0.5 secs to reduce the speed to v_{max}.

B. Shared controller applied to steering wheel angle

Using the shared controller for the steering wheel angle operates similarly as applied to the speed. If the human driver steers the car too far from the suggested control output, the parallel autonomy systems takes over control until the vehicle is within the safe zone (Fig. 14).

VI. CONCLUSIONS

A full-scale parallel autonomy research platform was developed based on our holistic approach and extensively tested. We demonstrated that our research platform enables parallel and full autonomy. The proposed drive-by-wire conversion method is allowing fast actuation, seamless takeover between human and autonomous system and operates in three modes without requiring proprietary CAN bus information or external motors. In combination with the relatively inexpensive sensor setup, it provides a lean and minimally invasive setup for an autonomous driving research platform.

As future work, we will integrate a more advanced shared controller [30] and run user studies. We will extend the

[1]Autonomous driving `https://youtu.be/9XDfz2PiNOk`
[2]Parallel autonomy `https://youtu.be/aPDx-wqV7Gk`

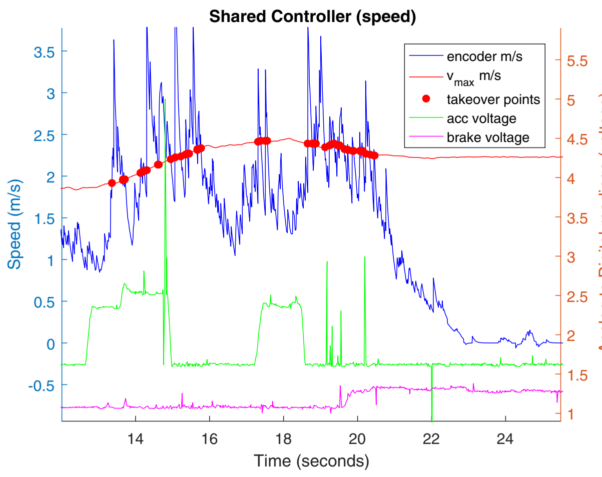


Fig. 13. Shared controller applied to speed: The red line shows the DVB advised speed for the current path. The blue line is the encoder speed measurement. Every takeover point is marked with a red dot. When the human is pressing the accelerator pedal the green line increases likewise the magenta line when the brake pedal is pressed.

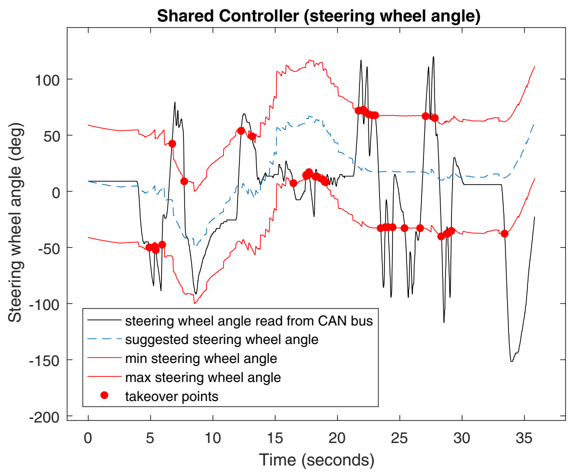


Fig. 14. Shared controller applied to steering wheel angle: When the human tries to turn too far to the left or right, the system takes over control. The blue line shows the advised steering wheel angle, the black line the CAN bus reading of the actual angle and the takeover points are marked with red dots again.

sensor setup and improve localization in non-urban environments. We will also test the system more on closed test fields and analyze the vehicles behavior statistically.

References

[1] N. H. T. S. Administration, "Preliminary statement of policy concerning automated vehicles," *Washington, DC*, pp. 1–14, 2013.

[2] G. S. D. C. Project. (2017, January) On the road. [Online]. Available: https://waymo.com/ontheroad/

[3] A. Singhvi and K. Russell. (2016, July) New York Times: Inside the self-driving Tesla fatal accident. [Online]. Available: https://www.nytimes.com/interactive/2016/07/01/business/inside-tesla-accident.html

[4] S. J. Anderson, S. B. Karumanchi, and K. Iagnemma, "Constraint-based planning and control for safe, semi-autonomous operation of vehicles," in *Intelligent Vehicles Symposium (IV)*. IEEE, 2012, pp. 383–388.

[5] S. M. Erlien, S. Fujita, and J. C. Gerdes, "Shared steering control using safe envelopes for obstacle avoidance and vehicle stability," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 2, pp. 441–451, 2016.

[6] A. Gray, M. Ali *et al.*, "A unified approach to threat assessment and control for automotive active safety," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1490–1499, 2013.

[7] A. Gray, Y. Gao *et al.*, "Stochastic predictive control for semi-autonomous vehicles with an uncertain driver model," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. IEEE, 2013, pp. 2329–2334.

[8] J. Wei, J. M. Snider *et al.*, "Towards a viable autonomous driving research platform," in *Intelligent Vehicles Symposium (IV)*. IEEE, 2013, pp. 763–770.

[9] P. Furgale, U. Schwesinger *et al.*, "Toward automated driving in cities using close-to-market sensors: An overview of the v-charge project," in *Intelligent Vehicles Symposium (IV)*. IEEE, 2013, pp. 809–816.

[10] S. Kato, E. Takeuchi *et al.*, "An open approach to autonomous vehicles," *IEEE Micro*, vol. 35, no. 6, pp. 60–68.

[11] M. Aeberhard, S. Rauch *et al.*, "Experience, results and lessons learned from automated driving on Germany's Highways," *IEEE Intelligent Transportation Systems Magazine*, vol. 7, no. 1, pp. 42–57, 2015.

[12] A. Daimler. (2014, September) Mercedes-Benz Future Truck 2025: World premiere of the spectacular study of tomorrows trucks-autonomous driving into an exciting future. [Online]. Available: http://media.daimler.com/marsMediaSite/ko/en/9918575

[13] R. Bradley. (2016, April) Tesla Autopilot. [Online]. Available: https://www.technologyreview.com/s/600772/10-breakthrough-technologies-2016-tesla-autopilot/

[14] M. Bojarski, D. Del Testa *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[15] A. Atreya, B. Cattle, and B. Collins, "DARPA Urban Challenge Princeton University Technical Paper," *Princeton University*, 2007.

[16] M. Montemerlo, J. Becker *et al.*, "Junior: The Stanford entry in the urban challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 569–597.

[17] A. Bacha, C. Bauman *et al.*, "Odin: Team VictorTango's entry in the DARPA urban challenge," vol. 25, no. 8, pp. 467–492. [Online]. Available: http://doi.wiley.com/10.1002/rob.20248

[18] C. Urmson, J. Anhalt *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," vol. 25, no. 8, pp. 425–466. [Online]. Available: http://doi.wiley.com/10.1002/rob.20255

[19] J. Leonard, D. Barrett *et al.*, "Team MIT urban challenge technical report," 2007.

[20] T. M. Sales. (2016, January) TIS (toyota technical information system). [Online]. Available: http://toyota.us/2nuF1WZ

[21] S. Pendleton, T. Uthaicharoenpong *et al.*, "Autonomous golf cars for public trial of mobility-on-demand service," in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE/RSJ, pp. 1164–1171.

[22] M. Quigley, K. Conley *et al.*, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[23] Z. Chong, B. Qin *et al.*, "Mapping with synthetic 2d lidar in 3d urban environment," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 4715–4720.

[24] ——, "Synthetic 2d lidar for precise vehicle localization in 3d urban environment," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2013, pp. 1554–1559.

[25] B. Qin, Z. Chong *et al.*, "A spatial-temporal approach for moving object recognition with 2d lidar," in *Experimental Robotics*. Springer, 2016, pp. 807–820.

[26] R. Tedrake, "Drake: A planning, control, and analysis toolbox for nonlinear dynamical systems," 2014. [Online]. Available: http://drake.mit.edu

[27] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," DTIC Document, Tech. Rep., 1992.

[28] M. Krucker, R. Triebel, and R. Bitzi, "Communication with a Toyota Prius," 2009.

[29] T. Moore. (2016, September) Package summary ROS robot localization. [Online]. Available: http://wiki.ros.org/robot_localization

[30] W. Schwarting, J. Alonso-Mora *et al.*, "Parallel Autonomy in Automated Vehicles: Safe Motion Generation with Minimal Intervention," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*.