iDriver - Human Machine Interface for Autonomous Cars

Arturo Reuschenbach, Miao Wang, Tinosch Ganjineh, Daniel Göhring
Artificial Intelligence Group
Computer Science Institute
Freie Universität Berlin
Germany

***Abstract*— Modern cars are equipped with a variety of sensors, advanced driver assistance systems and user interfaces nowadays. To benefit from these systems and to optimally support the driver in his monitoring and decision making process, efficient human-machine interfaces play an important part. This paper describes the second release of iDriver, an iPad software solution which was developed to navigate and remote control autonomous cars, to give access to live sensor data and useful data about the car state, as there are, e.g., current speed, engine and gear state. The software was used and evaluated in our two fully autonomous research cars "Spirit of Berlin" and "Made in Germany".**

Key Words: autonomous, car, iPad, remote control, driver assistance, spirit of berlin, made in germany

I. Introduction and Motivation

With modern technology evolving, cars are equipped with a variety of new sensors and functions nowadays. These systems give the driver new possibilities to control or to interact with the car and to assist the driver. As advanced driver assistance systems (ADAS) become more sophisticated, the vision of an autonomous car could become reality within the near future.

A. Research in Autonomous Cars

Research in autonomous cars have received a broader interest in recent years as they have unfolded many insights for general robot systems in areas like safety, machine learning and environmental perception. The industry, especially automobile manufacturers, are eager to improve advanced driver assistance systems, such as lane departure warning and intelligent speed adaptation systems while military is strongly interested in unmanned vehicles for its use in reconnaissance and combat operations. From a Computer Science perspective unmanned vehicles serve as a research platform for progress in a variety of fields as machine learning, computer vision, fusion of sensor data, path planning, decision making, control architectures and intelligent autonomous behavior.

While unmanned driving is the ultimate goal, in the development process of autonomous vehicles human drivers must be present in case of failures. A remote control as human-machine-interface is a perfect instrument to send commands to the vehicles as well as receiving status data from without actually remaining in the car. We choose to use an Apple iPad tablet PC as such a remote device.

B. iPad Human Machine Interface

With the idea of an Apple iPad human machine interface we have the following key features in mind, the iPad should supply:

- As a remote control it is a perfect instrument to send commands to the vehicles as well as receiving status data without actually remaining in the car.
- Within the development and testing process, visual debugging tools are necessary. Thus, the iPad can display real time raw sensor data as well as processed data, e.g., recognized obstacles from laser scanners or image processing results.
- As a human interface for the driver its touch display can be used to input the wanted target position, checkpoints that should be passed and other data that is necessary to define the desired mission.

In this paper a software solution is presented which enables us to use an Apple iPad as a remote control and as a display for live sensor data. The mobile application was named iDriver and is a further developed version of the iPhone version [1]. Generally, tablet computers provide the necessary computing capabilities and sensor functionalities to act as a remote control. We chose the Apple iPad over other tablet computers because of its advanced multitouch recognition and its sensor capabilities. The iPad, in comparasion with the iPhone used for the first release, has a bigger display, longer battery life, and enough power for wireless communication. iDriver has been tested with two autonomous cars "Spirit of Berlin" and "Made in Germany" from Freie Universität Berlin.

C. Motivation and Applications

The motivation for a remote control for an autonomous car is twofold. First, we want to receive sensor data directly from the car and use the iPad as a diagnostic frontend for on-board data. This is beneficial in the development process for autonomous systems without the need of a supervisior in the car itself. Secondly, we want to send out data and commands to the autonomous car. We do this on three different levels: full control of gas/brake and steering wheel, input of mission points for an autonomous route, and activating an emergency

brake. The latter is essential for autonmous testing where safeguards (i.e. a driver) are required ready to initiate the emergency brake in case something fails. These precautions and other actions can be done remotely with a mobile device as described later.

D. Paper structure

The paper is structured as follows: After giving a short introduction to related work (Chapter II) within the field of autonomous cars and to earlier work on human-machine interfaces we will present key software features of the iDriver software in Chapter III. In Chapter IV we will introduce the testing ground and experiments we performed, a short conclusion and future work are given in Chapter V.

II. Related work

There are various research teams working on autonomous cars with the goal of unmanned driving in daily urban scenarios. Although driving without an actual human pilot is legally yet an open issue, the technology has reached a level where it can be safely field tested in urban scenarios alongside with regular human traffic participants.

Autonomous vehicles have received a huge research boost by the US Defense Advanced Research Projects Agency (DARPA) that has organized three challenges for unmanned land vehicles in 2004, 2005 and 2007. Over 100 teams participated in the first challenge, but none of them managed to complete the whole distance to win the price money. The best performance was accomplished by Carnegie Mellon Red Team's robot Sandstorm with 7.36 miles before crashing with a road obstacle [2].

The same challenge was repeated in 2005, with 195 applicants whereas 43 where chosen for a National Qualification Event (NQE) and 23 teams made it to the finals. All but one of the finalists surpassed the distance of Sandstorm in the preceding year. Five vehicles successfully completed the race with the winner robot Stanley from Stanford Racing Team that finished the course in under 7 hours [3].

In 2007, DARPA moved to an urban scenario: a 60 mile course in urban area in less than 6 hours including obeying all traffic laws while negotiating with other traffic participants and obstacles and merging into traffic. The winner was Tartan Racing (Carnegie Mellon University and General Motors Corporation) with their vehicle Boss and a finishing time of 4 hours and 10 minutes [4].

While the 2004 and 2005 events were more physically challenging for the vehicles, because the robots only needed to operate in isolation with focus on structured situations such as highway driving, the 2007 challenge required engineers to build robots able obey all traffic regulations and make intelligent decisions in real time based on the current situation.

No urban challenge was organized after 2007, which urged researchers to move to real traffic, testing autonomous vehicles along side with human road users. Most recent accomplishments include autonomous Toyoto Prius' cars from Google developed by a joined venture from researchers of Carnegie Mellon and Stanford University [5] in 2010. Seven test cars have driven 1,000 miles without human intervention and more than 140,000 miles with only occasional human control. Simultaenously, "Leonie", a Volkswagen Passat from Technische Universität Braunschweig, Germany, finished a round course in Braunschweig autonomously. Similary, "Made in Germany"of Freie Universität Berlin demonstrated crossing behaviour, traffic light detection and obstacle avoidance in urban situations.

A. Spirit of Berlin & Made in Germany

For tests we use two of our autonomous cars "Spirit of Berlin" and "Made in Germany" of Freie Universität Berlin (see Figure 1).

Fig. 1. The autonomous testing vehicles "Spirit of Berlin" (left) and "Made in Germany" (right).

"Spirit of Berlin" was the participating robot of Team Berlin in the 2007 DARPA Urban Challenge [6]. It finished as one of the 35 semifinalists. The 2007 team was a joint team of researchers and students from Freie Universität Berlin, Rice University, and the Fraunhofer Society working together with American partners. The vehicle was a retrofitted Dodge Caravan with drive-by-wire technology, modified so that a handicapped person could drive using a linear lever for brake and gas (the lever controls all intermediate steps between full braking and full acceleration), and a small wheel for steering the front wheels. The rest of the car's components can be controlled through a small contact sensitive panel or using a computer connected to A/D converters. Several sensors are used and mounted on top of the car: Two GPS antennas give information about the position and direction of the car. An IMU and an odometer provide temporal positioning information when GPS signal is lost. Two video cameras are mounted in front of the car for stereo-vision modules to detect lane markings and roadside. One of the two cameras broadcasts its video stream to the iPad to provide a view of where the car is heading. Three laser scanners are used to sweep the surroundings for any obstacles that need to be evaded. All sensor data are collected and fusioned into one state model about the vehicle itself and its surroundings that is representing the perception of the car.

A blade server from IBM provides the necessary computing power for the running software modules, the actual intelligence of the vehicle. Here, the fusioned sensor data are used to make decisions on what action needs to be executed next given the current situation. The necessary commands are then transmitted to actuators in the steering wheel, gas and brake pedals to execute the made decision. When the iPad

acts as a remote control in manual mode, the intelligence in the car is turned off, and the commands given from the phone are directly transmitted to the actuators.

"Made in Germany" is a successor to "Spirit of Berlin", a volkswagen Passat which drive-by-wire interface was factory-opened for us to control the car. Therefore no A/D converters were necessary to control the car, instead communication is established via ethernet directly to the Controller area network (CAN). The Passat has six Ibeo Lux laser scanners build in for a 360 degree view of its surroundings. Moreover, a Velodyne laser scanner on the roof enhances its perception by another 64 laser beams sweeping around the car. In addition to "Spirit of Berlin" the car also includes radar sensors from TRW and SMS that enables us to detect obstacles in close proximity with even higher resolution. This is indispensable for parking or shunting maneuvers.

B. Related Remote Control Systems

Related work on remote systems to control autonomous cars is rare. We have shown two methods controlling an autonomous car with a HED4 eyetracking system by SMI [7] and with an Apple iPhone [1]. The latter has become our ground work for this project.

Another approach has been done by Institute of Transportation Systems of German Aerospace Center DLR, where they ported their autonomous development and testing framework DOMINION directly onto the Apple iOS platform [8]. They were thus able to control autonomous cars, but limited it to gas/brake and steering only.

Other projects involving autonomous cars make do with an E-Stop system as a wireless link to initiate an emercency stop in case the car fails or goes critical [3].

III. iDriver Software

The iDriver software (Second Release) was developed for the Apple iPad and tested under iPad software version 3.2.2. Our approach involves a software solution to remote control an autonomous car and to display raw sensor and navigation data as well as recognized and tracked objects transmitted from the car.

The iPad tablet platform comprises all technologies for the development process of an autonomous car remote control. The big multitouch display, accelerometer and the powerful wifi processor allow a fluent transmission and easy control of the autonomous car. Nowadays, there are also tablet computers based on other operating systems like Windows Mobile or Linux. But because of the iPad's high market distribution there are more developement resources, guidelines and support than on others platforms. The iOS programming language (Objective-C) is based on C which enables us to use core system calls for TCP/IP communication. Other frameworks included in iOS simplify development in OpenGL, 2D or 3D animation and graphics.

A. Architecture

iDriver is developed for a client-server architecture where the client software resides in the remote sending out control signals to the server system to be controlled. The server is responsible to accept or reject these commands and execute them accordingly. As of a feedback service the server may return an answer back to the client.

The main data flow between iPad and car is outlined in Figure 2. The server is directly connected to the car via ethernet and is thus able to retrieve can messages, and laser sensor data (in our case: six Ibeo Lux sensors and one Velodyne scanner). Camera data are read via Firewire 400. Another micro controller acts as an emergency backup system that is also directly connected to CAN. The controller is needed to initiate an emergency stop in case the server fails.

The iPad receives its data over Wi-Fi using UDP packets for real-time data (e.g. camera data, sensor data and gps locations). Emergency stop signals, status data and commands to the server are transmitted over TCP for more reliability. For testing purposes the Wi-Fi network was left unsecure, however for general purposes a security policy and encryption is desirable.

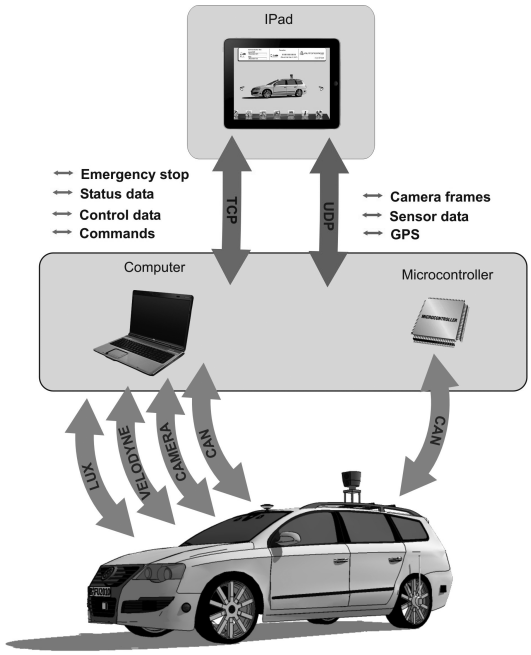

Fig. 2. Data flow chart: communication between iPad and main computer in the car

The multi-touch screen of the iPad can be used to capture multiple commands with more than one finger simultaneously, e.g. giving gas and steering to the left. As a safety measurement at least one touch needs to remain on the iPad to transmit data. The built-in accelerometer allows us to read out the current pose of the iPad, so that tilting to the left and right can be mapped to left and right steering actions.

B. Features

The application start view is divided into three areas. At the top status information is displayed such as Wi-Fi

connection with the server and the reception level with the round trip delay in seconds. In the middle of the screen there is a 3D Model of the autonomous car with a short description of all the sensors when touched. At the bottom, a scroll view with various icons gives access to all the features that are implemented. The main implemented features are emergency stop, dashboard, sensors, obstacles, track, mission, pick me up and drive.

Fig. 3. iDriver layout interface

1) Emergency stop: The Emergency stop feature provides the same function as an emergency stop switch that users activate to initiate the complete shutdown of a machine. In our case when the user pushes the emergency stop button the autonomous car will stop immediately and wait for new commands. We use this function to prevent malfunctions during the tests or to abort if anything goes wrong. When the user pushes the button, it generates a message with an special ID and it sends over a secure connection to the micro controller installed in the autonomous car. Afterwards, the micro controller can directly communicate with the CAN bus to initiate a emergency brake. Thus, the stop process is separated from the server and remains functional even if the server malfunctions.

2) Dashboard: The Dashboard feature is a digital instrument to wirelessly show various information of the autonomous car which would be available on the real dashboard. This enables the user to check on speed, revolutions per minute, heading and turn signals when the car is in autonomous mode. When the user starts this function the car encapsulates the local dashboard information in a message and sends it to the iPad in periodically short time intervals.

3) Sensors: In the Sensors view, raw data from every sensor is visualized for the user. Using a segmented control the user can switch between data from the cameras, Velodyne laser scaner, Lux (TM) laser scaner, TRW (TM) radar and SMS (TM) radar sensors. The raw data is processed by the server and projected in a 3D simulator around the car. A image capture of this simulator is then transmitted to the iPad using a Bayer filter.

4) Obstacles: The Obstacles feature displays objects that have been recognized and tracked by the sensors from their raw data. With the camera raw data we can recognize static objects like traffic lights or traffic signs and moving objects like vehicles or pedestrians that are in front of the car and in the field of view of the camera. With the Velodyne and Lux sensors we can scan 360 degrees around the car and recognize and track objects like vehicles, pedestrians, trees, walls, etc. In contrast to camera data, the laser scan data incorporates position estimation and therefore a distance measurement to the car's chassis. With the radar sensors we can also scan for incoming obstacles but with higher precision and independent of weather conditions. All this objects are also projected in a 3D simulator and thereafter broadcasted as a 2D texture that is send line by line to the client using a Bayer filter.

5) Track: The track feature displays a map of the actual positon and trajectory of the autonomous car since the user activated the function. The GPS position of the car is periodically sent to the client and visualized as a path on the map. It is useful to know the car's position at any time when in autonomous mode.

6) Mission: The mission feature allows the user to send a sequence of checkpoints to the server that the autonomous car has to process. When the user starts the function the server sends all the available checkpoints for the given region to the client. The user can then start a mission in a 2-step workflow: First, the user selects a sequence of checkpoints for the car to process. The checkpoints are visualized on a map along side the cars chosen trajectory. Secondly, once the mission is confirmed by the user it is transmitted to the car. When the server confirms the new sequence the user can start the mission pushing a button and the autonomous car has to drive to all checkpoints in the same sequence as the user has put in.

7) Pick me up: The pick me up feature allows the user to call the autonomous car and get picked up. The client sends its GPS positon to the server where the server projects this position to the closest street reachable from the car's current position. A mission is generated for the car to drive from its current to the user's location. Thus, when the user starts this function you can initially see where the car is located, after that the user needs only to touch the pick me up button to iniate the mission.

8) Drive: In drive mode the car hands over control to the iPad allowing the remote user to steer, brake and accelerate the car with touches and tilt movements. All necessary control information (desired gas or brake position, steering wheel position, desired gear and turn signal) is packed into one single UDP packet and then send out to the server. The gas or brake position consists of a float value between -

1.0 (maximum brake) and 1.0 (maximum gas), the default value for brake is set to -0.11 allowing a smooth deceleration until a full stop is reached. Similarly, the gas value is set to 0.3 by default to limit the maximum speed in a test scenario to about 15 km/h. Another float value holds the steering wheel position with -1.0 for maximum left and 1.0 for maximum right. As a safety precaution to prevent rapid turns the steering wheel position is limited to -0.7 till 0.7. The desired gear is saved as an integer value with 1 = park, 2 = reverse, 4 = neutral and 8 = drive. The turn signals are also stored in an integer with 0 = both signals off, 1 = left signal on, 2 = right signal on, 3 = both signals on. Both float values for gas/brake and steering and both integer values for gear and turn signal are packed into one 16-byte UDP packet shown in Table I.

TABLE I
STRUCTURE OF OUTGOING UDP CONTROL PACKETS

gas/brake	steering wheel	gear	turn signal	horn
4 bytes	4 bytes	4 bytes	4 bytes	4 bytes

When in driving mode, UDP packets are only sent out when at least one finger is touching the screen. If no fingers are touching the screen no UDP packets are sent out, causing the car to perform an emergency stop after a short time interval. This is a safety measure to ensure that the car comes to a full stop when the connection between remote and car is lost or if the car is out of Wi-Fi range.

The car is functioning as a server for the remote and is sending back two kinds of packets: feedback and camera packets. Feedback packets are sent for every incoming control packet described above, containing 16-byte information for packet version, current speed, set gear and set turn signal mode. Table II shows the structure of feedback packets. The speed value is displayed on screen whereas gear and turn signal information are used by the client to verify the last sent desired gear and turn signal commands.

TABLE II
STRUCTURE OF INCOMING UDP FEEDBACK PACKETS

version number	current speed	gear	turn signal
4 bytes	4 bytes	4 bytes	4 bytes

Camera packets are constantly sent from the car to the iPad so the remote user has a visual feedback of where the car is heading. Each packet contains data for several rows of the camera image encoded with a Bayer filter and additional meta information. The meta information consists of three short values for which row is sent, the maximum number of rows and how many rows are sent. The raw image data follows the meta information as the payload of the packet. Table III shows the structure of camera packets.

The remote software receives these camera packets and updates part of its OpenGL ES texture accordingly which is displayed in the main screen. With the use of a Bayer filter the bandwidth of camera data is reduced to one third compared to raw RGB data. iDriver then uses a pixel and fragment shader as an efficient demosaic filter on the GPU described by McGuire [9].

TABLE III
STRUCTURE OF INCOMING UDP CAMERA PACKETS

row	maximum rows	number of rows	image data
2 bytes	2 bytes	2 bytes	variable

IV. TESTS

The iPad remote control has been extensively tested with our autonomous vehicles “Spirit of Berlin” and “Made in Germany”. When the iPad was connected to the car and its CAN bus, the CAN messages were retrieved with a frequency of 100 Hz and sent to the iPad via an ad-hoc wireless network. Packet losses due to UDP were minimal and noncritical for display purposes at 100 hertz. Sensor data are retrieved and transmitted as UDP packets with different frequencies: 12.5 Hz for Ibeo Lux sensors, 15 Hz for Velodyne sensor and 25 Hz for camera data. This is sufficient for visualisation and debugging tasks. GPS packets for tracking the cars position are also broadcasted with 10 Hz.

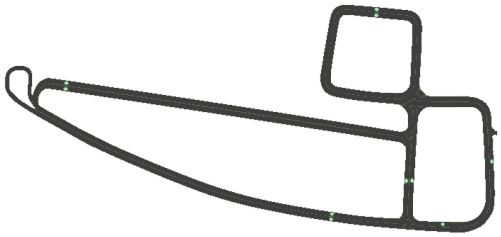

Fig. 4. Testing ground on Central Airport Berlin-Tempelhof

Long-run driving tests have been performed on the now defunct Central Airport Berlin-Tempelhof, Germany. A round course was designed including various left and right turns, 3-way and 4-way crossings and long straight high-speed segments (see Figure 4). 11 mission points have been incorporated into the course, from which different mission scenarios were sent from the iPad to the car. Additional challenges have been included: traffic lights, other traffic participants at crossings, unexpected obstacles passing the road like pedestrians and static obstacles forcing a overtaking maneuver. The car executed all missions flawlessly and came to a full stop once a mission has been completed.

We first measured the range of simple Wi-Fi connection between iPad and car and then the delay and rate of packet loss for different ranges. The iPad kept an stable connection with the car up to 200 meters in open area and up to 150 meters in urban areas. For higher ranges a switch to other wireless technology, such as WiMAX or UTMS is required.

The delay between iPad and the server was about 100-150 ms. It took another 100 ms for the signals to reach the gas/brake and steering engines from the computer. Thus, we had a delay of 200-250 ms to control the car. This was sufficient for all our main features.

Packet loss was a greater challenge as UDP does not provide reliability or data integrity. However, as a real-time system we are prefer dropping packets rather than waiting for them. For control packets we had an average packet loss of 9% which was negligible as control packets are sent with a high frequency. It was more crucial for camera data and sensor data packets as they have a much higher payload. We had about 19% packet loss in close proximity and up to 41% packet loss at maximal range. Packet loss was compensated by compressing image data with a Bayer filter and avoiding repeated transmission of unchanged sensor data.

V. Conclusion

This paper has described the architecture and mechanics of the iDriver remote software on an Apple iPad. It is used as a human-machine interface for autonomous vehicles to display live sensor data and useful data about the car state.

We have described each of the main features of the software in detail, especially features to observe sensor data or the car's state: dashboard, sensors, obstacles. Other features allow the user to remote control the car by calling out to be picked up or sending out distinct missions. Furthermore it allows a mode where manual driving with the remote is possible.

The software has been tested on the now defunct Berlin-Tempelhof airport with two of our autonomous cars: "Spirit of Berlin" and "Made in Germany".

A. Future Work

Besides incorporating features of new sensors, in future works it might be interesting to analyze, in how far location based services are applicable with car autonomy. E.g., it is imaginable that the driver uses the iPad to find the next gas station or repair facility and the car will plan and find its way to it. Other features, as a visual status watchdog showing if something in the car went wrong, or, classically traffic jam warnings could be feasible extensions.

As of future work we would also like to incorporate a security protocol to our communication between car and iPad and try to integrate more on-board diagnostic information.

Although it would be technically feasible as the DARPA Urban Challenge or iDriver obviously demonstrated, there are many legal issues to be worked out. For example, if two robot cars crash into each other in traffic, it is not resolved who will be responsible for the accident. Will it be one of the owners of the car or will it be the manufacturer or will it even be the company that built the failing module that was responsible for the software error? It is obvious, that manufacturers and legal institutions are not eager to discuss the terms of completely autonomous or remote controlled driving in the near future. However, we believe that autonomous features, as some of them are described in this paper, will eventually be brought into series-production readiness as part of more sophisticated advanced driver assistance systems because they may provide necessary information for drivers or even prevent accidents.

References

[1] M. Wang and T. Ganjineh, "Remote Controlling an Autonomous Car with an iPhone," tech. rep., Free University of Berlin, 2009.

[2] C. Urmson, J. Anhalt, M. Clark, T. Galatali, J. P. Gonzalez, J. Gowdy, A. Gutierrez, S. Harbaugh, M. Johnson-Roberson, H. Kato, P. Koon, K. Peterson, B. Smith, S. Spiker, E. Tryzelaar, and W. Whittaker, "High speed navigation of unrehearsed terrain: Red team technology for grand challenge 2004," Tech. Rep. CMU-RI-TR-04-37, Carnegie Mellon University, June 2004.

[3] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Stanley: The robot that won the darpa grand challenge," *Journal of Field Robotics*, vol. 23, pp. 661 – 692, September 2006.

[4] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. N. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. M. Howard, S. Kolski, A. Kelly, M. Likhachev, M. McNaughton, N. Miller, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski, B. Salesky, Y.-W. Seo, S. Singh, J. Snider, A. Stentz, W. Whittaker, Z. Wolkowicki, J. Ziglar, H. Bae, T. Brown, D. Demitrish, B. Litkouhi, J. Nickolaou, V. Sadekar, W. Zhang, J. Struble, M. Taylor, M. Darms, and D. Ferguson, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, vol. 25, pp. 425 – 466, July 2008.

[5] "Google cars drive themselves, in traffic." New York Times, October 2010.

[6] R. Rojas, K. Gunnarsson, M. Simon, F. Wiesel, F. Ruff, L. Wolter, F. Zilly, N. Santrac, T. Ganjineh, A. Sarkohi, F. Ulbrich, D. Latotzky, B. Jankovic, G. Hohl, T. Wisspeintner, S. May, K. Pervoelz, W. Nowak, F. Maurelli, and D. Droeschel, "Spirit of Berlin: An Autonomous Car for the DARPA Urban Challenge - Hardware and Software Architecture," tech. rep., Free University of Berlin, June 2007.

[7] M. Wang and D. Latotzky, "Driving an Autonomous Car with Eye-Tracking," tech. rep., Free University of Berlin, 2009.

[8] S. Montenegro, F. Dannemann, L. Dittrich, B. Vogel, U. Noyer, J. Gacnik, M. Hannibal, A. Richter, and F. K"oster, "(spacecraft-buscontroller+automotiveecu)/2=ultimatecontroller," *LNCS GI Software Engineering 2010 / ENVISION2020*, no. 160, pp. 103–114, 2008. ISSN: 1617-5468.

[9] M. McGuire, "Efficient, high-quality bayer demosaic filtering on gpus," *Journal of Graphics, GPU, & Game Tools*, vol. 13, no. 4, pp. 1–16, 2008. ISSN: 2151-237X.