FREIE UNIVERSITÄT BERLIN

Remote controlling an autonomous

car with an iPhone

Miao Wang

B-10-02
March 2010

FACHBEREICH MATHEMATIK UND INFORMATIK
SERIE B · INFORMATIK

Remote controlling an autonomous car with an iPhone

Miao Wang
Free University of Berlin
Department of Computer Science
14195 Berlin, Germany
mwang@mi.fu-berlin.de

Abstract

This paper describes iDriver, an iPhone software to remote control "Spirit of Berlin". "Spirit of Berlin" is a completely autonomous car developed by the Free University of Berlin which is capable of unmanned driving in urban areas. iDriver is an iPhone application sending control packets to the car in order to remote control its steering wheel, gas and brake pedal, gear shift and turn signals. Additionally, a video stream from two top-mounted cameras is broadcasted back to the iPhone.

1 Introduction and Motivation

Autonomous cars are robots capable of unmanned driving in an unknown urban area. Research in autonomous cars have received a broader interest in recent years and unfolded many insights for robot systems in different applications. The industry, especially automobile manufacturers, are eager to improve advanced driver assistance systems (ADAS), such as lane departure warning and intelligent speed adaptation systems while the military is strongly interested in unmanned drones for military purposes. From a Computer Science perspective autonomous vehicles function as a research foundation for progress in machine learning, computer vision, fusion of sensor data, path planning, decision-making 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 is a perfect instrument to send commands to the vehicles as well as receiving status data from without actually remaining in the car. In this paper a solution is described where an Apple iPhone is used as a remote control. Mobile phones like the iPhone possess the necessary computing capabilities and sensor functionalities to act as a remote control. The remote can toggle between completely autonomous mode, where the phone acts as a mobile monitoring panel, and manual mode, where the operator can use the phone to steer and drive the car.

The mobile application was named "iDriver" and tested with the autonomous car "Spirit of Berlin" from the Free University of Berlin which also participated in the 2007 DARPA Urban Grand Challenge. We tested the remote software on the abandoned airfield in Berlin Tempelhof, where a human operator sent commands for the desired position of steering wheel, gas and brake pedal as well as gear shift and turn signals to the car. Additionally, a video stream from two mounted cameras on top of the car is broadcasted back to the iPhone as feedback.

A remote control is typical for a client-server-architecture whereas 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 remainder of this paper is structured as follows. In Section 2, we give information on the background of autonomous cars in the DARPA Grand Challenge and also describe the architecture of "Spirit of Berlin". Section 3 explains the user interface of the remote software iDriver. The following two sections will describe the communication between the phone and the car: Section 4 will describe the communication from the phone to the car, whereas section 5 will describe the communication back from the car to the phone. Finally, Section 6 presents our conclusions and potential future use of this work.

2 Related Work

2.1 DARPA Grand Challenge

The US Defense Advanced Research Projects Agency (DARPA) has organized three challenges for unmanned land vehicles in 2004, 2005 and 2007.

In 2004, the goal was to drive a predefined route of 150 miles within 10 hours in Barstow, California. Price money of 1 million dollar was awarded for the team who finished first. Over 100 teams participated in this 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 [1].

DARPA repeated the same challenge 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 [2].

In 2007, DARPA moved to a urban scenario. The Urban Challenge in November 2007 took place at the site of the now-closed George Air Force Base in Victorville, California. The goal involved finishing 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 [3].

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.

2.2 Spirit of Berlin

"Spirit of Berlin" was the participating robot of Team Berlin of the Free University of Berlin in the 2007 DARPA Urban Challenge [4]. It finished as one of the 35 semifinalists. The 2007 team was a joint team of researchers and students from Free University of 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 antenna give information about the position and direction of the car. An IMU unit 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 camera broadcasts its video stream to the iPhone 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 iPhone 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.

3 User Interface

The remote software was developed for the Apple iPhone 3GS. The multi-touch screen of the iPhone can be used to capture multiple commands with more than one finger simultaneously, e.g. giving gas and steering to the left. The user interface of iDriver is similar to many available race games for the iPhone: Two touch sensitive buttons represent gas and brake control, touching them will send out commands either to give gas or to brake. If both buttons are pressed, the brake control overrides the gas control. The steering wheel is controlled by the built-in accelerometer sensors: Tilting the phone to the left will cause the car to steer left; tilting to the right will cause the car to steer right. The operator can activate the steering wheel control by touching the wheel. The steering wheel control is inactive by default to prevent continuing steering commands being send out when the user accidentally drops the phone. Two arrows on the left and right represent controls for the turn signals, where an operator can turn either left, right or both signals on or off. A flashing of the arrow control depicts the activity of the corresponding turn light. The top bar shows information about the steering and gas/brake value currently send out to the car as well as the current vehicle speed. An info button toggles the settings view, in which the default value for gas and brake can be set up. The gear can also be changed via the settings menu. Since the "Spirit of Berlin" is an automatic, the gear can only be switched between reverse, park, neutral and drive. Holding the brake is required for a gear shift. The iPhone possesses a 120 MHz PowerVR-MBX graphics processor, capable of rendering OpenGL ES 2.0. This graphics processor is used to

display the camera image from the video camera of the car onto the main screen.

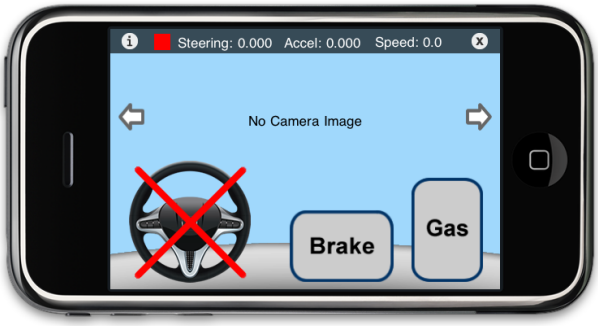

Figure 1: Graphical User Interface of iDriver

4 Communication from iPhone to car

The communication between iPhone and car is based on simple Wi-Fi. The transmission from the phone to the car requires the values for the desired gas or brake position, steering wheel position, desired gear and turn signal activity. All this information is packed in one 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 -0.44 allowing a smooth deceleration until a full stop is reached. Similarly, the gas value is set to 0.58 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 1.

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

Table 1: structure of outgoing UDP control packets

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 send 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.

5 Communication from car to iPhone

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 send for every incoming control packet described above, containing 16-byte information for packet version, current speed, set gear and set turn signal mode. Table 2 shows the structure of feedback packets. The speed value is displayed in the top bar whereas gear and turn signal information are used by the client to verify the last sent desired gear and turn signal commands.

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

Table 2: structure of incoming UDP feedback packets

Camera packets are constantly send from the car to the iPhone. 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 send, 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 3 shows the structure of camera packets.

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

Table 3: structure of incoming UDP 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 comapred to raw RGB data. iDriver then uses a pixel and fragment shader as an efficient demosaic filter on the GPU described by McGuire [5].

6 Conclusion

This paper has described the architecture and mechanics of the iDriver remote software. Completely autonomous driving as well as manual driving with a remote like a mobile phone will probably not be accessible for everyone to use in everyday's traffic for the next few years. 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.

Nevertheless there are still areas where a car remote can be used in a beneficial way. We will briefly describe three possible applications for the future. The first is commercial use of the car remote for specific maneuvers where jurisdiction can be clearly stated. For example, instead of opening a garage door every morning to drive out the car and then closing the door again, all can be executed with a remote control right before leaving the front door. The garage door will automatically open while the car drives out before the door closes up again. This option can save time and is comfortably to use. The same can be imagined for parking in a parking space or parking structure. The second usage was already outlined in the beginning of this paper: When testing for autonomous cars safeguards are required like people sitting in the car ready to hit the emergency brake in case something fails. These precautions and other actions can be done remotely with a mobile device. Mobile devices can also act as a mission control overview where checkpoints can be defined for an autonomous route. Lastly, another commercial use of mobile devices would be to let them act as a diagnostic frontend to visualize what is wrong with a car. Most vehicles today have much more diagnostic information stored than is shown on the dashboard. On-Board Diagnostics (OBD) has also been standardized for many years to refer to the car's self-diagnostic and reporting capability. A mobile remote device can read out this information to display the necessary steps to repair the vehicle or to offer the option to call a more specialized mechanic.

We see our work on iDriver as a proof-of-concept on which more applications can be built upon. As of future work we would like to incorporate a security protocol to our communication between car and iPhone and try to integrate more on-board diagnostic information. For a spinoff project we also envision the visualization of laser scan data and mission control for autonomous cars.

References

[1] 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.

[2] 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.

[3] 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.

[4] J. Rojo, 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.

[5] 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.