2018-01-0035 Published 03 Apr 2018

Development of an Autonomous Vehicle Control Strategy Using a Single Camera and Deep Neural Networks

Anthony Navarro Udacity

Jendrik Joerdening Akka Technologies

Rana Khalil and Aaron Brown Udacity

Zachary Asher Colorado State University

Citation: Navarro, A., Joerdening, J., Khalil, R., Brown, A. et al., "Development of an Autonomous Vehicle Control Strategy Using a Single Camera and Deep Neural Networks," SAE Technical Paper 2018-01-0035, 2018, doi:10.4271/2018-01-0035.

Abstract

Autonomous vehicle development has benefited from sanctioned competitions dating back to the original 2004 DARPA Grand Challenge. Since these competitions, fully autonomous vehicles have become much closer to significant real-world use with the majority of research focused on reliability, safety and cost reduction. Our research details the recent challenges experienced at the 2017 Self Racing Cars event where a team of international Udacity students worked together over a 6 week period, from team selection to race day. The team's goal was to provide real-time vehicle control of steering, braking, and throttle through an end-to-end deep neural network. Multiple architectures were tested and used including convolutional neural networks (CNN) and recurrent neural networks (RNN). We began our work by modifying a Udacity driving simulator to collect data and develop training models which we implemented and trained on a laptop GPU. Then, in the two days between car delivery and the start of the competition, a customized neural network using Keras and Tensorflow was developed. The deep learning network algorithm predicted car steering angles using a single front-facing camera. Training and deployment on the vehicle was completed using two GTX 1070s since a cloud GPU computing instance was neither available nor feasible. Using the proposed methods and working within the competition's strict requirements, we completed several semi-autonomous laps and the team remained competitive. The results of the competition indicated that autonomous vehicle command and control can be achieved in a limited form using a single-camera with a short engineering development timeline. This approach lacks robustness and reliability and therefore, a semantic segmentation network was developed using feature extraction from the YOLOv2 network and the CamVid dataset with a correction for the unbalanced occurrence of the different classes. Currently 31 classes can be reliably detected and classified allowing for a more complex and robust decision making architecture.

Introduction

Modern vehicles are recently evolving into intelligent vehicles due to an increased understanding and implementation of computerization and software [1]. An intelligent vehicle is defined as a system that can sense the driving environment and provide information or vehicle control to assist the driver in improved vehicle operation [2]. Intelligent vehicles have the ability to sense the vehicle's own status and its environment, communicate with the environment, as well as plan and execute appropriate maneuvers [2, 3].

Currently there is a global need to improve vehicle safety. In the United States, 90% of vehicle crashes are due to driver error [4] resulting in 37,461 deaths in 2016 [5]. This danger has elicited safety as the number one goal of U.S. Department of Transportation [6]. But, intelligent vehicles have the capability to improve vehicle safety by assisting or removing the driver completely. Currently this has manifested itself in significant and recent improvements in Advanced Driver Assistance Systems (ADAS) [7] that are safety focused [8] and include a variety of sensors including cameras, radar, and ultrasonic detection [9]. But, it is projected that the current levels of driving assistance will be replaced by fully autonomous vehicles in the near future which not only drastically improve safety, but may positively affect congestion, parking, and economics [10].

Autonomous vehicle development was jump-started by the Defense Acquisitions Research Project Agency (DARPA) Grand Challenges. The first DARPA Grand Challenge took place in 2004, where researchers were tasked with

autonomously navigating a 142 mile course through the Mojave desert in no more than 10 h. None of the 100+ teams were able to navigate more than 5% of the course [11]. In 2005, the cash prize was doubled to $2 million and five teams were able to finish, the winner being the Stanford "Stanley" robot [11]. In 2007, the DARPA Urban Challenge was initiated where competitors were tasked with driving on roads, handling intersections and maneuvering in zones [12]. Six teams finished with Carnegie Mellon University winning due to their techniques of decomposing the problem into perception, mission planning, behavioral, and motion planning layer components [12]. Due to the advancement of autonomous technology from these competitions, numerous technology, transportation, and automotive companies have autonomous vehicle programs designed to transition modern vehicles with ADAS to semi-autonomous intelligent vehicles, and eventually to fully autonomous intelligent vehicles [10].

In order to further the advancement of autonomous vehicle technologies, competitions are still held. One of the major sponsors for these competitions is the education focused company, Udacity [13]. In February 2017, Udacity selected a group of students to participate in the Self-Driving Cars event at Thunderhill Raceway in California [14]. Seventeen students from five different countries were tasked with design deep learning algorithms to make it around the Thunderhill track autonomously. The constraints of the competition were than only a single forward-facing Point Grey Blackfly camera and Swift Navigation GPS unit could be used for sensing.

Our team decided to use a novel approach of a single forward facing camera and an advanced artificial neural network to control the vehicle autonomously. The approach is advantageous since it is relatively simple and cost-effective compared to the sensor fusion of multiple cameras, radar, and lidar which is the current standard in autonomous vehicle technology [15]. If it can be demonstrated that this novel approach works in a high stress racing environment, it may be able to supplant the current standard.

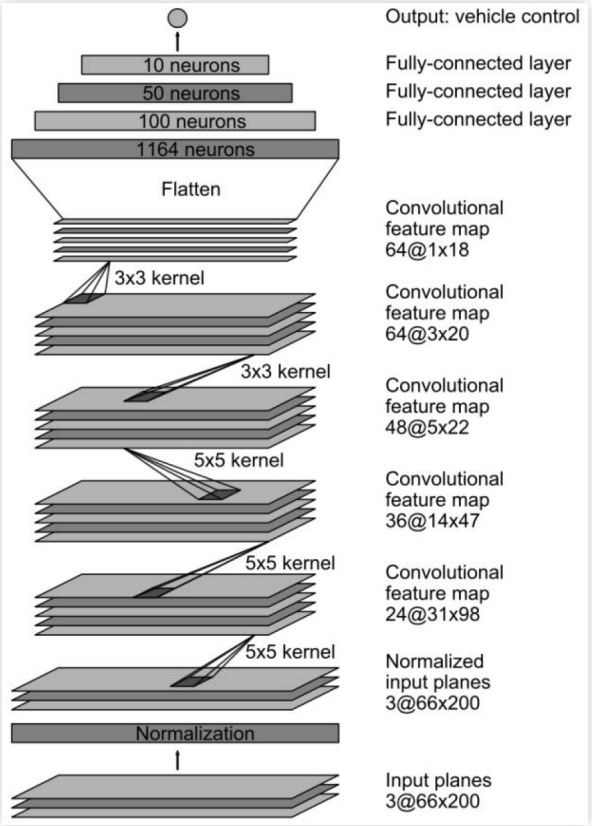
Methods

Approach

There are two approaches to autonomous vehicle control that are currently being researched in academia and deployed in industry. Traditional computer vision techniques are well understood and explainable however, their optimization and combination of techniques is still a topic of research. Another approach to controlling a system based on perception is to use machine learning and neural networks by allowing important features to automatically be learned.

The first practical application of an artificial neural network was in the 1950s [16] but did not experience mainstream application until understanding of statistical mechanics [17] and the development of back propagation in the 1980s [18]. Since then, artificial neural networks have been a powerful tool when using a large set of data where the important features are learned by the algorithms instead of being pre-programmed by humans [19]. There are numerous types

FIGURE 1 NVIDIA's PilotNet architecture for end-to-end vehicle control [23].



of artificial neural networks that can be used, but deep neural networks (DNNs) have a demonstrated applicability for autonomous driving [20, 21]. The team chose to implement an end-to-end deep learning solution due to the limited amount of sensors and recent experience performing a similar task in the Udacity Self-Driving Car Nanodegree program [22] in a simulated environment. The base DNN created for the race was based on the NVIDIA PilotNet model [23].

A completely different approach that was used during the race event and is well understood in the industry is called GPS waypoint following. GPS waypoint following is a simple technique where a GPS system is used to record multiple points along a path travelled. Once the points are recorded, the GPS system is used to localize to those points and the system is controlled to minimize the error from the current location to the next position that had previously been recorded. The team chose not to use this method since it did not present a technical challenge like a DNN and did not demonstrate a new or emerging application of technology. Many other teams on the track were using this method and were able to successfully navigate the track with it.

Simulator

Since the team was located worldwide, the physical vehicle was not available for most of the work performed. Instead, the team worked with Udacity to recreate a version of the Thunderhill track in the Udacity Simulator. This allowed the

FIGURE 2 Udacity simulator with Thunderhill course map.

team to begin testing algorithms before having access to the physical vehicle.

In order to predict how an autonomous car might behave on a real track, it's helpful to first test it in a detailed simulator that mimics the actual environment. A replica of the Thunderhill race course was created using the Unity game developer engine. Actual satellite image textures and geological terrain height maps were used to create the virtual track. Although this method is fast, the downside is the resolution for both the height map and color is low. While additional time could have been used to improve the fidelity of the simulator, the environment would never accurately mimic the real-world. The purpose of the simulator was to demonstrate a proof of concept and working model but not to get a fully-functioning and accurate solution. The simulator was able to interact with an autonomous driving script at a rate of 10 Hz for reliable control.

By extending the Udacity open-source simulator [24] with the Thunderhill course map, it was possible to have the user drive along the virtual course and record training data for how they would steer the car. This data was used to train a DNN to output predicted steering angles from image input alone.

Data Collection

The use of a deep learning algorithm requires a lot of training data to function properly. In this case, the data that was needed was images from the camera on the vehicle. This was easy to collect in the simulator by driving laps around the virtual track just as you would in a video game. However, while it is easy to collect this data, there are inherent issues with simulated data. In the real-world, the time of day affects many aspects of an image like glares, shadows, reflections, and more. Since the algorithms learn from the data, there is no way to tell the system to ignore things like a shadow or a glare. Humans easily understand that artifacts like this are not important when driving but an algorithm does not inherently know that. This is also why it is difficult to train on simulated data and then expect the inference on real-world data to perform as well as inference on simulated data. When the team had issues testing the simulator trained models, the first thing that was done was to collect real data with the vehicle. Data was collected using the PolySync vehicle while at Thunderhill to get accurate training data. Many methods were used while collecting data including driving the course with the vehicle in the center of the track, driving the race line on the track, and varying speeds from very conservative to the fastest you could push a Kia Soul. Over 36,000 images were collected and used in training the DNN. In addition to just collecting image data, the team also manipulated the data using various rotations and transformations to augment and increase the original set of data. This included shifts to the top and bottom, augmenting the steering to be weaker or stronger, respectively, and to the left and right adding a steering contradicting this shift. Finally, the images were rotated and a the steering corrected accordingly to be stronger to the right, if the image was rotated to the right and inversely as well. OpenCV was used to perform all image transformations.

There was some issues collecting data originally. Due to a previous bug, the system was not recording the throttle data of the vehicle. Because of limited time constraints, the team decided to make an attempt to derive this information instead of generating new data. While not directly correlated, velocity can be related to throttle and braking events. To try and make use of the data that had already been recorded, the team determined the first derivative of the velocity to gain acceleration information. This was done for both, the data containing throttle information and the ones without. This way a conversion could be found following:

$$k_{throttle} = \alpha \frac{dv}{dt} \tag{1}$$

This very simplistic conversion gave the results as visible in Figure 4. These are by far not perfect, but in the limited time range the most robust model available.

Architecture

The team used a hybrid architecture utilizing PolySync's custom build software and visualization studio called Core [25]. This system used DDS as a middleware to command and control the vehicle's steering, throttle, and brake. The car's

FIGURE 3 Example image data collected by Udacity race team. Notice the mirrored image and prevalent glare on the glass.

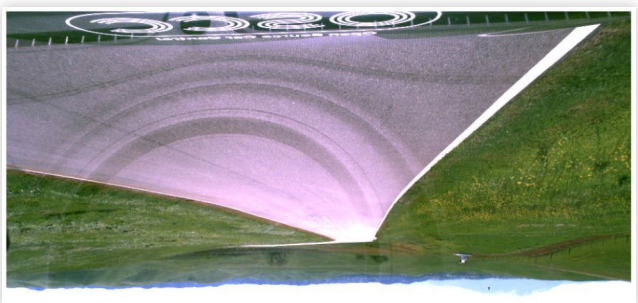
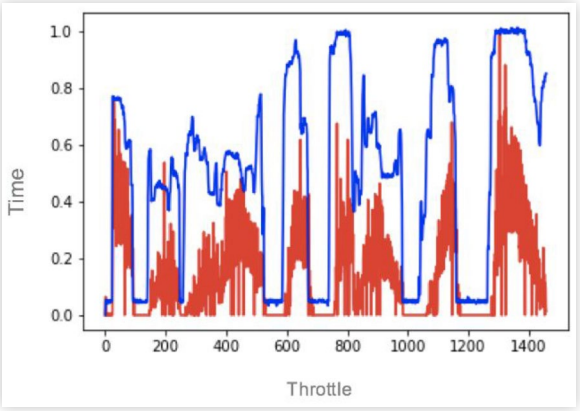


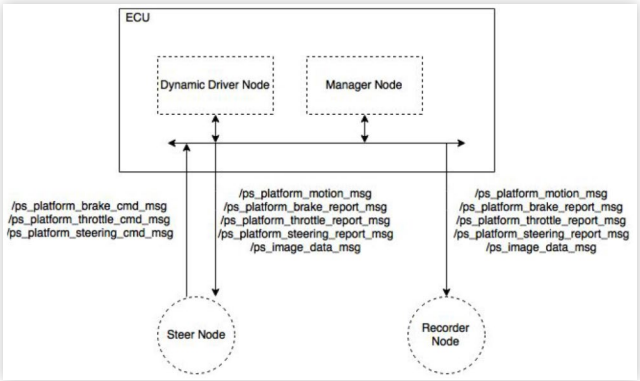
FIGURE 4 Comparison of throttle and scaled acceleration.



software architecture handled data received and transmitted at high rates from both the car's platform and sensors over the DDS bus to enable processing and intelligence on the higher level Polysync Pub-Sub and ROS stack. The data was being transmitted at a range of 60 to 100 HZ on the DDS bus to a ROS bridge to publish the data to other nodes for image and data processing and augmentation.

The ROS architecture composed and held most of the deep learning and path planning algorithms to autonomously steer the car in real-time based on data processed from the car and the sensors mounted as well. One of the main challenges encountered during the race was making sure the different pieces of the architecture were testable and performing at an appropriate rate to match how fast the car is navigating. The team programmed the deep learning algorithms in Python while the rest of the architecture in C++, which meant there was a need for a C++ to Python bridge. Degraded performance was encountered during testing which led to delayed model predictions as well as delayed command issuance in the rest of the drive-by-wire stack. The team found the most accurate way of testing and predicting the results was to use the Root Mean Square Error (RMSE) resulting from our ground truth predictions. The tools built for RMSE calculation helped speed up models development, tuning and iteration.

FIGURE 5 Software architecture communicating with PolySync Core.



Data Interpolation and Exploration

Steering, Throttle and Brake Data was collected in two different ways during the race. Information based on the car driving a conservative path in the middle of the road was used to maximize the performance and accuracy. Conversely, data was recorded where the driver drove the "race line" and had harder braking and acceleration points to maximize lap time.

During the data recording sessions, it was interesting to study the relationships between how steering would look differently at a variety of speeds and brake mechanisms.

What might seem quite intuitive to a driver, is a learning experience to an autonomous vehicle. From Figure 8, it seems during a specific steering angle, the driver needs to render high brake values, while at medium steering angles at less sharper edges, our brake values average and stay up to a certain braking threshold. While a driver might intuitively guess and have an instinct based on experience on how much to apply brake to continue steering while traveling at a high speed, the correlation is necessary for the vehicle to learn and render a decision at fast turning points around the track.

Looking through the relationship between the speed and steering in Figure 9, the team noticed a different sort of relationship where steering and speed during the moments where the driver needs to apply light steering adjustments, almost a circular continuous graph is plotted showing easing into smaller changes in speed. On the other hand, steeper steering angles in clockwise and counterclockwise show different levels of reduced speed, which makes quite sense since the track had steeper counterclockwise turns in the middle of the track vs. a couple gentle clockwise turns in the middle of the track and in the beginning of the track.

GPS Data for Localization and Steering The PolySync vehicle was equipped with an MTi-G-710 GPS sensor which was used to collect ground truth position information to fuse into the PoseNet model for training during the race.

A simple visualization script was developed to determine the error in the recorded data. It was found that an error of 1-2 meters existed but it was determined that this would be

FIGURE 6 Visualizing the relationship between steering and brake while racing around the track.

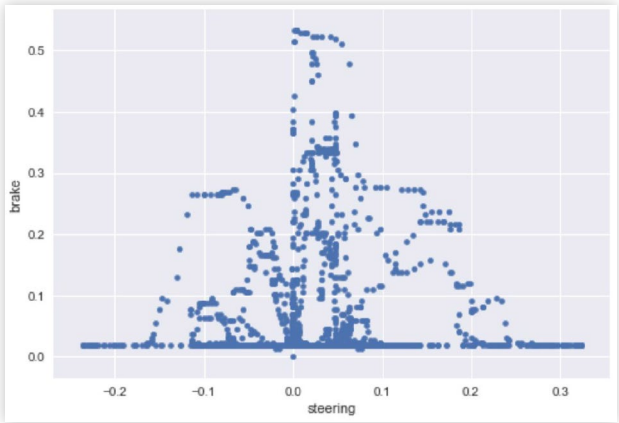


FIGURE 7 Visualizing relationship between steering and speed of the vehicle while skid steering around the track.

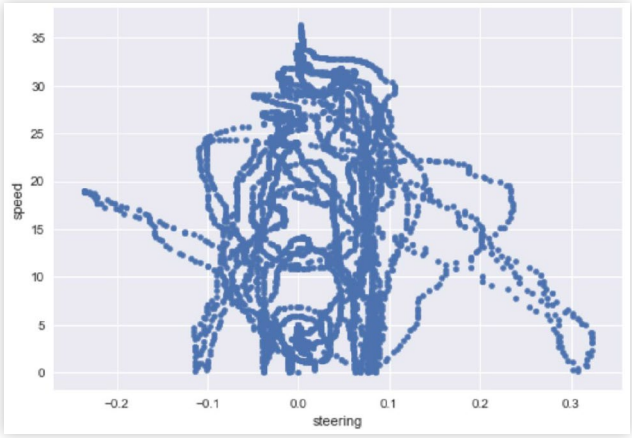
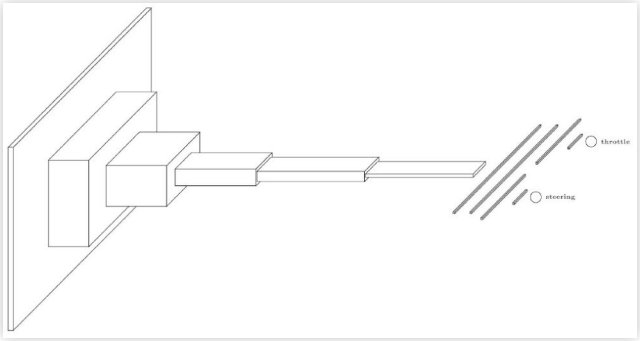


FIGURE 8 GPS data collected around thunderhill track visualization.

FIGURE 9 Simple DNN architecture for feeding images and predicting steering and throttle.



an acceptable range for the race. The GPS sensor had less than 2 ms latency, which provided the architecture with an adequate and timely flow of position information to predict and navigate around the track.

Models

The team used multiple variants of DNNs to solve the challenge. One commonality that all models shared was the utilization of a Convolutional Neural Network (CNN) which is a multi-layer deep neural network that is trained with a version of the back-propagation algorithm. CNNs are designed to recognize patterns and are robust to distortions and simple geometric transformations [26]. CNNs have demonstrated a 15.3% error rate in classifying 1.2 million images into 1000 different classes through convolutional layers and "dropout" regularization [27].

In this case, a CNN followed by a dense DNN was utilised to first extract features from the image then learn the throttle and steering angle commands based on these features.

This approach performed with promising results in a simulated environment. Transferring it to a real driving situation was not amount of data available and the utilised augmentations. Therefore, two other architectures were utilised investigated and utilized.

The first approach utilized a recurrent neural network (RNN) with a Long short-term memory (LSTM) cell after the convolutions to predict the commands based on the current as well as the previous features. This approach led to a lot smoother commanding, but at the same time it put too much emphasis on previous driving situations, which is not optimal if accurate turns are expected to be performed.

The second approach is a modification to the previous CNN with two changes: the current speed was added as an input to the model and a normalized position should be predicted as an output. The normalization of the position was performed such that it was given as two features between −1 and 1. For this, the GPS was converted to UMC and a Min-Max-Scaler was utilised transforming it using:

$$UMC_i = 2\frac{UMC_i - UMCi,min}{UMC_{i,max} - UMC_{i,min}} \quad (2)$$

This modification gave the DNN information about the current speed which was influencing both the steering as well as the throttle command. At the same time, it had to determine its position during training time. This forced the neural network to be aware of the concept of positioning of the race track and allowed it to determine the steering as well based on these activations. This enables the DNN take the image data it is receiving from the camera at the current time and augment some form of position data to it.

Hardware

The vehicle used during the race was a 2014 Kia Soul that had been modified by PolySync. Using the Open Source Car Controller (OSCC) system [28] they developed, the car was controllable by wire for the steering, braking, and throttle.

FIGURE 10 PoseNet fully connected dense layers.

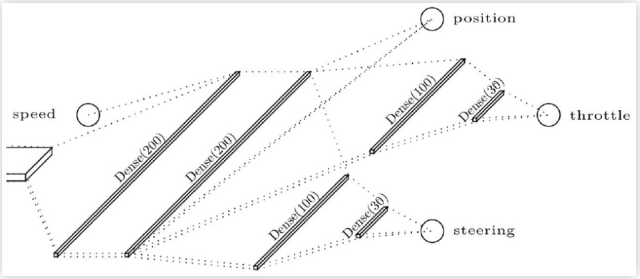


FIGURE 11 PolySync Kia Soul used by Udacity race team.

The vehicle was equipped with multiple types of sensors used by autonomous vehicles including LIDAR, radar, camera, IMU, and GPS. During the race, only the single forward facing Point Grey Blackfly camera and Swift Navigation MTi-G-710 GPS system were used. To power the autonomous algorithms, an Alienware computer equipped with two NVIDIA GTX 1070 graphic cards was used. The graphics cards are critical for controlling a DNN based system since images have to be processed extremely fast in order to allow for the vehicle to react and accurately navigate the track.

The purpose of the OSCC system is to provide a cheap and accessible solution for drive-by-wire system on the 2014+ Kia Soul. To provide complete control of the vehicle, the OSCC exploits the throttle-by-wire and steer-by-wire capabilities of the Kia Soul. For the braking system, a slight modification to the vehicle is needed by adding an electronic brake controller from a Toyota Prius to the Kia Soul braking system. To ensure safe controllability, three modules are created: a CAN gateway module to translate Kia specific CAN messages onto a new Control CAN bus, and a power distribution system that has emergency stop capability are necessary. [29].

FIGURE 12 Open source car controller architecture [28].

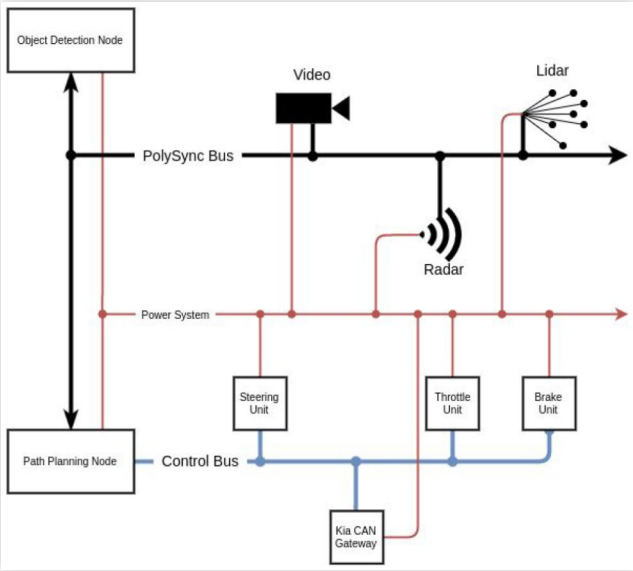


FIGURE 13 Image of the trained semantic network.

Results

Only four autonomous vehicle teams that took part in the 2017 Self-Racing Cars event obtained a fully autonomous lap. They were able to accomplish this by utilizing GPS waypoint following. By the final round of the race, the Udacity team unable to successfully make it around the track fully autonomously. However, during the final runs while using the PoseNet approach, the team was able to complete every turn on the track autonomously just not all during the same lap. Since time on the track was limited and data collection difficult once the race began, this limited the team in exploring the results further. Given a day or two of additional time, the team would have had the opportunity to record additional training data and fine tune models.

The Udacity team spent six weeks preparing for the race, only four of those days with the car, 48 hours participating in the race, and only about two hours actually racing on the track. During this short period of time, the team was able to implement a semi-autonomous end-to-end, deep neural network controlled vehicle using only a single camera and GPS unit. While unable to complete a fully autonomous lap, the team demonstrated that deep learning showed promise as awas a valid method for vehicle control and could be implemented to assist self-driving cars, on the race track or off.

To improve on these results, a second approach is being developed that will use a semantic segmentation network along with path planning and control algorithms. Semantic segmentation is a process where a CNN is used to "paint" every pixel in an image to identify the classification of it. The network being used is based on YOLOv2 network [30] and the CamVid dataset [31, 32]. Once a drivable surface can be accurately identified, a path planning algorithm can be applied to find the time minimising trajectory and this can be followed using classical control algorithms like a model predictive controller.

This race, and other research from NVIDIA and others, demonstrate that it is possible to create end-to-end vehicle control in an autonomous vehicle. This method has a lot of advantages with the speed of programming and relying on feature identification from the computer. However, while it might make sense to have a system on the race track that relies on DNNs, when autonomous vehicles are deployed in for public use, there will most likely be a number of solutions incorporated in order to ensure a reliable and fault tolerant system.

Conclusions

Committing to using ROS as part of our self-driving car architecture will enable us to utilize other localization, path planning, visualization and perception tools and support which will speed up our development and testing process. While our architecture might need a fair amount of tuning and adding some safety functionalities in the future, it will enable us to quickly prototype and iterate over our end to end architecture to autonomously navigate the race track. Once the current architecture is proven reliable, ROS 2.0 will be investigated and implemented. There are many advantages to ROS 2.0 including DDS communication like PolySync Core. Many companies are looking at specialized version of ROS 2.0 to enhance safety and reliability.

References

1. Charette, R.N., "This Car Runs on Code," *IEEE Spectrum* 46:3, 2009.
2. Bishop, R., "Intelligent Vehicle Technology and Trends," trid.trb.org, 2005.
3. Gusikhin, O., Filev, D., and Rychtyckyj, N., "Intelligent Vehicle Systems: Applications and New Trends," . In: *Informatics in Control Automation and Robotics*. (Berlin Heidelberg, Springer, 2008), 3-14.
4. National Highway Traffic Safety Administration, "*National Motor Vehicle Crash Causation Survey*," (U.S. Department of Transportation, 2008).
5. National Highway Traffic Safety Administration, "*2016 Fatal Motor Vehicle Crashes: Overview*," (U.S. Department of Transportation, 2017).
6. U S House of Representatives, "MAP-21 Conference Report to accompany H.R. 4348," 2012.
7. Bengler, K., Dietmayer, K., Farber, B., Maurer, M. et al., "Three Decades of Driver Assistance Systems: Review and Future Perspectives," *IEEE Intelligent Transportation Systems Magazine* 6:6-22, 2014.
8. Mahajan, H.S., Bradley, T., and Pasricha, S., "Application of Systems Theoretic Process Analysis to a Lane Keeping Assist System," *Reliability Engineering and System Safety* 167:177-183, 2017.
9. Kukkala, V.K., Tunnell, J., Pasricha, S., Bradley, T., "A Survey of Advanced Driver Assistance Systems and Current Challenges. In Review".
10. Fagnant, D.J. and Kockelman, K., "Preparing a Nation for Autonomous Vehicles: Opportunities, Barriers and Policy Recommendations," *Transportation Research. Part A, Policy and Practice* 77:167-181, 2015.
11. Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D. et al., "Stanley: The Robot that Won the DARPA Grand Challenge," *Journal of Field Robotics* 23:661-692, 2006.
12. Urmson, C., Andrew Bagnell, J., Baker, C.R., Hebert, M. et al., "Tartan Racing: A Multi-Modal Approach to the DARPA Urban Challenge," 2007.
13. Cameron, O., "Race Self-Driving Cars With Udacity! - Udacity Inc - Medium," https://medium.com/udacity/race-self-driving-cars-with-udacity-42ae12e545c1.
14. Gundling, C., "Our Very Own Grand Challenge - Udacity Inc - Medium," https://medium.com/udacity/our-very-own-grand-challenge-b004a9863024.
15. Sun, Z., Bebis, G., and Miller, R., "On-Road Vehicle Detection: A Review," *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28:694-711, 2006.
16. Rosenblatt, F., "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain," *Psychological Review* 65:386-408, 1958.
17. Hopfield, J.J., "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," *Proceedings of the National Academy of Sciences of the United States of America* 79:2554-2558, 1982.
18. Rumelhart, D.E., McClelland, J.L., and PDP Research Group, editors, "*Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1: Foundations*," (Cambridge, MIT Press, 1986).
19. Demuth, H.B., Beale, M.H., De Jess, O., and Hagan, M.T., "*Neural Network Design*," (USA, Martin Hagan, 2014).
20. Sallab, A.E.L., Abdou, M., Perot, E., and Yogamani, S., "Deep Reinforcement Learning Framework for Autonomous Driving," *Electronic Imaging* 2017:70-76, 2017.
21. Ohn-Bar, E. and Trivedi, M.M., "Looking at Humans in the Age of Self-Driving and Highly Automated Vehicles," *IEEE Transactions on Intelligent Vehicles* 1:90-104, 2016.
22. "Self Driving Car Engineer Nanodegree," https://www.udacity.com/course/self-driving-car-engineer-nanodegree--nd013.
23. Bojarski, M.Del Testa, D., ,Dworakowski, D., Firner, B. et al., "End to End Learning for Self-Driving Cars," http://arxiv.org/abs/1604.07316, 2016.
24. Udacity, "Udacity's Self-Driving Car Simulator," https://github.com/udacity/self-driving-car-sim.
25. PolySync Core, https://polysync.io/core/.
26. LeCun, Y., Others, "LeNet-5, Convolutional Neural Networks," http://yann. lecun. com/exdb/lenet, 2015.
27. Krizhevsky, A., Sutskever, I., and Hinton, G.E., "ImageNet Classification with Deep Convolutional Neural Networks," . In: Pereira F., Burges C.J.C., Bottou L., and Weinberger K.Q., editors. *Advances in Neural Information Processing Systems 25*. (Curran Associates, Inc., 2012), 1097-1105.
28. PolySync. "Open Source Car Control," https://github.com/PolySync/OSCC/wiki.
29. PolySync, "Hardware," https://github.com/PolySync/oscc/wiki/Hardware-Main.
30. Redmon, J. Farhadi, A. and, "YOLO9000: Better, Faster, Stronger," arXiv preprint arXiv:1612. 08242, 2016.
31. Brostow, G.J., Fauqueur, J., and Cipolla, R., "Semantic Object Classes in Video: A High-Definition Ground Truth Database," *Pattern Recognition Letters* 30:88-97, 2009.
32. Brostow, G.J., Shotton, J., Fauqueur, J., and Cipolla, R., "Segmentation and Recognition Using Structure from Motion Point Clouds," . In: *Computer Vision - ECCV 2008*. (Berlin, Heidelberg, Springer, 2008), 44-57.

Contact Information

Anthony Navarro
Udacity
2465 Latham St.
Mountain View, CA 94040
anthony.n@udacity.com

Acknowledgments

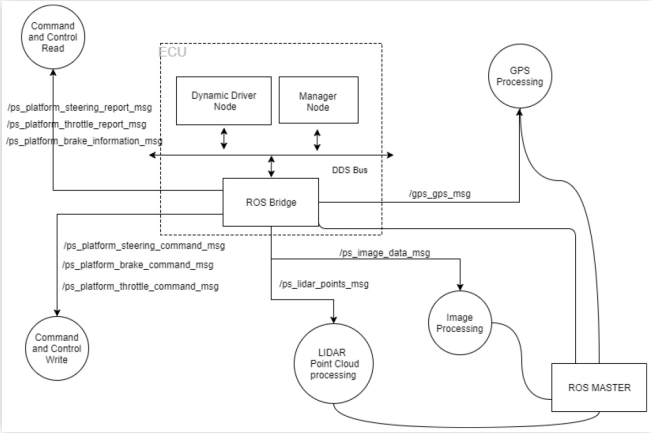
The accomplishments here were achieved by many great individuals and with the support of amazing companies. Every member of the Udacity Race team deserves high amounts of praise (Vlad Burca, Dean Liu, Chris Gundling, Maruf Maniruzzaman, John Chen, Chandra Sureshkumar, Nadia Yudina, Kiarie Ndegwa, Christy Cui, Harinando Andrianarimanana, Jacob Thalman, Karol Majek, Claudio Salvatore De Mutiis, Jerrick Hoang, and.

Definitions/Abbreviations

ADAS - Advanced Driver Assistance Systems
CamVid dataset - This is that one dealy????
CNN - Convolutional Neural Network
DARPA - Defense Advanced Research Project Agency
DNN - Deep Neural Network
IMU - Inertial Measurement Unit
GPS - Global Position Satellite
GPU - Graphics Processing Unit
LSTM - Long Short Term Memory
RNN - Recurrent Neural Network

Appendix

FIGURE 14 Embedded layer communication with the higher-level ROS architecture.



ISSN 0148-7191