Don't Delete Carriage Returns Above. The Title of Your Paper Goes Here Using the Paper Title Style

Author Names Go Here Using the Author Name Style

Affiliations and Addresses Go Here Using the Affiliation and Address Style  
publications15@aaai.org

Abstract

This is an example abstract. It goes here in 9 point Times New Roman type.

Introduction

This project is part of the course CS 686 Introduction to Artificial Intelligence by Kate Larson at the University of Waterloo. A for us interesting application domain should be chosen for this project and within this domain a problem is supposed to be identified. The goal of the project is to analyze which artificial intelligence techniques or other approaches are used to tackle this problem. This project will deal with autonomous driving and different approaches to obtain the steering angle of a self-driving car.

Application Domain

The application domain of this work is autonomous driving. Autonomous driving has been part of many movies like Batman (1966 to 1968) or Knight Rider (1982 to 1986) and seemed to be a utopian idea of driving. But within the last 10 to 15 years a lot of research was done on this topic and great progress was made. Self-driving cars become more and more realistic and it seems to be only a matter of time when the first fully autonomous vehicle is licensed.

Two previously milestones were the PROMETHEUS project in 1980 and the DARPA Grand Challenge in 2004. Due to the enormous research on this topic within the first mentioned project a vehicle was able to drive 95 % of the 1,600-km long drive autonomously. Later in 2004, the goal of the challenge was to drive a 150-mile off-road track as fast as possible without any human interaction with the car. This was the inception for many further events like the Hyundai Autonomous Challenge in 2010 or the drive of Daimler with an autonomous vehicle on the historic Berta-Benz route. Especially the Google self-driving car and Tesla’s autopilot gain a lot of attention. [1, 1, p. 34]

The increasing research and progress in this area are mostly connected to the different advantages of self-driving cars. Firstly, driverless cars could increase safety and reduce the number of deadly crashes caused by human faults. Moreover, it is an option to enable mobility for disabled or older persons. Less traffic jams are predicted as well as an increasing road capacity and less fuel consumption. Furthermore, it could be changing our relation to vehicles. Instead of owning a car by yourself you could share a ride or a vehicle whenever you need one. However, there also critical and yet unanswered questions about licensing, security, insurance regulations and many more. [2, pp. 167-168]

This is a Subsubsection Heading

In a typical self-driving car, the different processes are ordered hierarchically. At first, different sensors like LIDAR, cameras and GPS units are used to locate the vehicle itself, to perceive its environment and measure important values of the vehicle. In combination with prior information about the road network, e.g. due to a digital map, it is necessary to plan a route through this network. In the behavior layer several decisions are made. These decisions will lead the car to the destination and ensures that the vehicle satisfies the traffic rules. Afterwards, motion planning determines the desired trajectory of the vehicle. In the last layer the self-driving car uses a controller to calculate a steering angle in order to keep the vehicle on the desired trajectory or to minimize the deviation of the desired route. [1, p. 35]

Some of the latest research suggest a different architecture and approach for the trajectory planning and tracking. Instead of separating these two steps, end-to-end learning is used by NVIDEA [3]. This system, known as PilotNet, is based on a neural-network and can output the steering angle only by the pictures of the front camera of the vehicle. In order to reach this goal, the network was trained with road images that are linked to a steering angle generated by a human driver.

The remainder of this structured as follows: Firstly, different approaches for trajectory planning and tracking are introduced. Secondly, methods using end-to-end learning to determine the steering angle will be presented and ….

Typical methods of trajectory planning and trajectory tracking

As mentioned above, the typical architecture of a self-driving car is separated in hierarchical processes. Two of them are trajectory planning and trajectory tracking. For each of these tasks are several different approaches used in the literature. A comprehensive overview cannot be presented within this project and thus it is recommended to use special surveys like [1]. Nevertheless, particular approaches are chosen to illustrate what trajectory planning and trajectory tracking means and how it is achieved.

Trajectory Planning

Motion planning or trajectory planning has the task to define a trajectory from the current position of the vehicle to the goal position. In comparison to a path, a trajectory describes the status of a vehicle over time. This trajectory must fulfill certain criteria. Firstly, the trajectory has to be collision free and safe. But it is also important that the trajectory is feasible regarding the dynamics and kinematics of the vehicle. Finally, the planned trajectory should be comfortable. [1, p. 39]

To solve a trajectory planning problem, it is possible to use certain variational methods or to transform the problem into a path planning problem. In general, there exist three different kind of methods to tackle these problems. Non-linear optimization is used for variational methods and hence calculating a trajectory is done by defining and solving an optimization problem. For path planning problems known graph search methods are available. This is done by discretizing the state space such that a graph is generated, and a minimum cost path can be found, often variants of A\* are used. Finally, incremental methods scan the state space and design a reachability graph incrementally. The distinctive property of this method is that the size of the graph grows incrementally until a solution is found that satisfies the requirements. [1, pp. 40-48]

**Trajectory Planning for Bertha**

After introducing trajectory planning in general, this paper will introduce specific approaches and look at the implementation in a more detailed way. The first approach is designed by Ziegler, Bender et al. and is described in [4]. This local and continuous method is designed such that there are no local maxima and the solution will be the global maximum due to a special specification of the constraints. Both the optimization problem and the inequality constraints are nonlinear.

Ziegler, Bender et al. use a global coordinate system to plan the trajectories. As one advantage is mentioned that offline maps can be used easily and important information like the right and left driving bound can be obtained. This information is useful to create a so called ‘driving corridor’, an area in which the vehicle is allowed to drive. To detect objects and obstacles vision control and radar sensors are used. Common methods of preprocessing like sensor fusion and validation stage are applied to improve the accuracy of the data.

The rear axle center point of the vehicle is considered as the point that follows the designed trajectories. A trajectory is represented by , the curvature of the trajectory and the tangent angle by and respectively. Both can be calculated with the derivates of the trajectory at time [4, p. 451]. By minimizing the following integral

,

with

|  |
| --- |
| . |

an optimal trajectory can be found for the current time and the time horizon . Every summand in contains an individual weighting factor . The offset is considered by

|  |
| --- |
| . |

Due to this term trajectories that run through the middle of the driving corridor get a lower L-value and are hence preferred. The distance functions and are signed and points left and right of a bound are positive and negative respectively. The next term

|  |
| --- |
|  |

minimizes the deviation between the desired velocity and the velocity of the trajectory. The absolute value of the desired velocity is the current speed limit and is obtained by the behavior generation with the offline map. The velocity vector of the trajectory is orthogonal to the gradient of and . This causes the velocity vector to be parallel to the bounds of the driving area. Thus, the vehicle drives along the desired direction. The remaining terms will establish smoothness for the driving dynamics and comfort. Firstly, the term

|  |
| --- |
|  |

avoids extreme accelerations, both in longitudinal and lateral direction. Furthermore, a minimum of changes in the acceleration is achieved by

|  |
| --- |
| . |

However, fast changes of the driving direction of the vehicle and thus of the steering wheel angle are still possible. Therefore, the last term is introduced to minimize the yaw rate :

|  |
| --- |
|  |

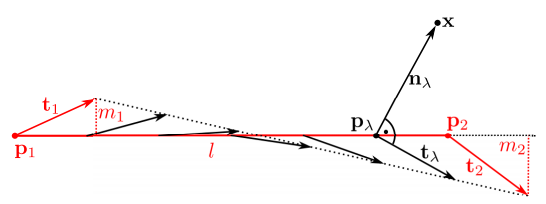
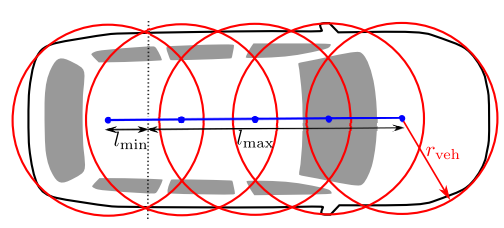
 The optimal trajectory not only minimizes the integral of (1) but also must hold several constraints. According to [4], constraints can be divided into two classes, internal and external constraints. Internal constraints satisfy the requirements of the vehicle dynamics and kinematics. For instance, at lower speeds the curvature of a trajectory must be restricted due to the steering system of the vehicle. The front wheels of the car can only be turned until a certain angle, depending on the vehicle, and all trajectories that require a higher steering angle are not admissible. According to the geometry of the steering system, limits for the curvature are calculated:

Figure 2: Representing the Shape of a Vehicle with Circles [].

Figure 1: Calculation of the Pseudo Distance [].

|  |
| --- |
|  |

However, at higher speeds the forces at the tire are the limiting factor. If a vehicle drives a turn, the radial force induced by the lateral acceleration can exceed the friction force between the road and the tires. As a consequence, the vehicle leaves the road. One way to depict this constraint is by fulfilling the restriction

|  |
| --- |
| . |

External constraints result from the map, which provides the driving corridor, and from the obstacles and objects which are perceived by the sensors. Each of these external constraints are represented by polygons. To establish the equation system, it is necessary to calculate for every point the distance to a polygon . Since Ziegler, Bender at al. use a Newton-type optimization method, it is important that the constraints are continuously differentiable. Therefore, a pseudo distance is used because non convex polygons do not ensure that this is true. Let and be two neighboring corner points of a polygon and and the tangent vectors at this corner points, then a linear segment of a polygon is represented by . For every point

|  |
| --- |
|  |

between the corner points (), a tangent vector is defined by

|  |
| --- |
| . |

The value is calculated, see figure 1, such that the pseudo tangent and the pseudo normal vector are orthogonal and hence

|  |
| --- |
| . |

Furthermore, it is admissible to assume and , thus and . Then is calculated by

|  |
| --- |
| . |

Finally, the pseudo distance is determined by the length of the vector if and the negative value of the distance otherwise. To figure out the distance to a complete polygon, the distance to every line segment is calculated and the smallest value is chosen.

Because the trajectory is created for the rear axle center point, the shape of the vehicle must be taken into account. The vehicle shape is represented by circles with a radius of , see figure 2. Let be the set of circle center points and the set of all polygons, then the following constraints must be satisfied:

|  |
| --- |
|  |

Trajectory Tracking

After the trajectory planning determined an optimal path, it is the goal of the trajectory tracking module to stabilize the vehicle on this desired path by minimizing the deviations from this path. These deviations arise because of inaccurate vehicle models [1, p. 36] and are the input for the feedback controller, which calculates the commands like the steering wheel angle for the actuators of the vehicle [5, p. 2322]. A comprehensive, but not all-embracing, survey of in the past used controllers is provided in [1].

**Dynamic Bicycle Model**

Because vehicle models play an important role regarding the accuracy of the controller, the most used model will be briefly introduced in this paper. A representation of the model can be seen in figure 3. The here presented vehicle model can be found in [6, pp. 28-31].

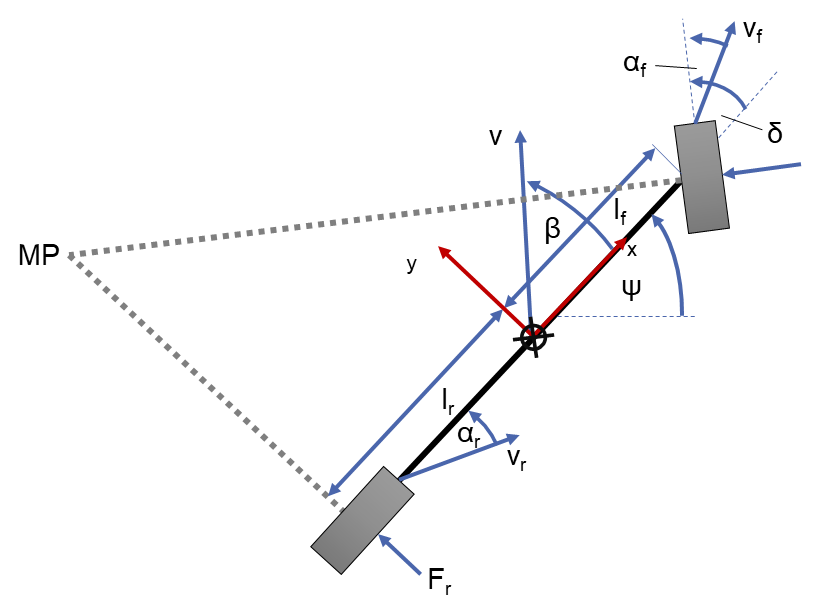
 Due to the tremendous complexity of the dynamics and movement of vehicles, it is necessary to make certain assumptions. Important assumption of the dynamic bicycle model are that the center of gravity is on the ground, hence movement is only considered in the plane, and that the vehicle is symmetrical. Using Newton’s equation of motion for the lateral direction and for rotation leads to the two equations

Figure 3: Dynamic Bicycle Model.

|  |
| --- |
|  |
| with and   |  | | --- | | . | |

Using further equation of the dynamics of vehicles, e.g. for the slip angle and lateral forces, Snider builds a system of equations for the lateral acceleration and the yaw rate ,which needs to be linearized if linear control methods are used. An extensive explanation of the vehicle model, however, would exceed the scope of this work.

**Trajectory Tracking by Immersion and Invariance**

One method to keep the vehicle on the desired path is presented by Tagne, Talj et al. in [5]. The proposed method has the advantage, in comparison to model predictive control, that the computation time is very low and thus suitable for real-time operations [5, p. 2322]. Due to my work experience with lateral controllers for trajectory tracking during my bachelor thesis, it is an additional advantage that the controller can handle different velocities and is, unlike model predictive control methods, configured for a specific velocity.

The inventors of this method use the dynamic bicycle model, too. Their formulation of the model results in two equations for the side slip angle and the yaw rate :

|  |
| --- |
|  |
|  |

Additional values are the velocity in longitudinal direction and the steering angle . The remainder of the values are vehicle parameters.

Moreover, the lateral error dynamics are represented by

|  |
| --- |
| . |

The two values, and , represent respectively the lateral acceleration of the vehicle and the desired acceleration of the vehicle. With

|  |
| --- |
|  |

and

|  |
| --- |
|  |

the dynamic lateral error can be rewritten as

|  |
| --- |
|  |

and using the equation for results in:

This equation is used to write an equation system in the form of

|  |
| --- |
| . |

After defining the equilibrium point, new variables and several mathematical transformations, the calculations result in one final equation for the steering angle

with two new parameters and . The angle , the curvature , the yaw rate and the longitudinal velocity are measured. The constants and must be determined but can be obtained by test drives or a single measurement. The lateral error will be calculated by the current position and the desired position of trajectory. Additionally, the derivative is obtained by the formula, see [7, p. 34],

|  |
| --- |
|  |

with as the angular difference between the orientation of the vehicle and the angle of the tangent vector of the reference point on the desired trajectory. The values of the constants and are determined by simulation or testing.

Given this equation, it is possible to calculate a steering angle at every moment that keeps the vehicle as close as possible to the desired trajectory. This angle is the input for the actuator, which moves the front wheels into the according position.

End-to-End Learning for Self-Driving Cars

Besides obtaining the steering wheel angle with a modular method, which consist amongst others of trajectory planning and trajectory tracking, different approaches using end-to-end learning were in the latest focus of the news. Especially, NVIDEA’s method, which uses a convolutional neural network (CNN), gained enormous attention.

Even if the idea of using CNN is not new – first implementations were made 20 years ago – latest developments influenced their success. On the one hand, large data sets are available and can be used for adopting the CNN. On the other hand, the process of learning is faster due to the implementation of these algorithms on GPUs. [8, p. 2]

Before end-to-end learning with CNN for self-driving cars is introduced, it is necessary to give a brief overview of CNN and how they work.

Figure 5: The Process of Convolution []

Basic Principles of CNN

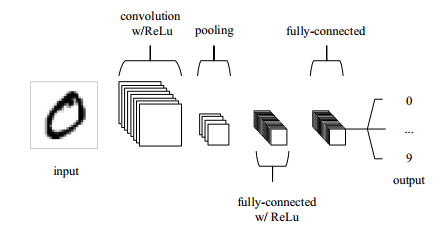
This chapter will present the most important information about CNN. The architecture of a CNN is illustrated in figure 4 for the recognition of handwritten numbers.

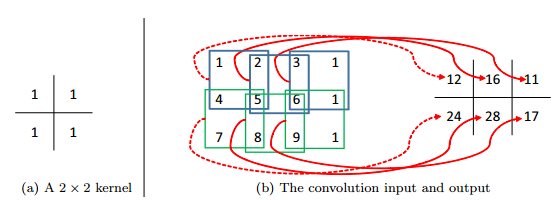
Figure 4: A simplified architecture of CNN []

CNNs are often used for pattern recognition in pictures. Therefore, the input is likely to be images and this is why the architecture of CNNs needs to be best suited for this specific kind of input. More specific, the input is an order 3 tensor. For an image with pixels the input is a tensor with rows, images and three channels for the colors red, green and blue. In general, a convolutional layer, a pooling layer and a fully-connected layer are the three typical layers of a CNN. However, an additional layer is added at the end to perform the learning and adopt the parameter. This is called backward error propagation and a common method to do this is stochastic gradient descent. Assuming a parameter exists, then the update rule for this parameter is

|  |
| --- |
|  |

with the learning rate and the loss . The last part of the equation is called gradient. The learning rate should not be too large because otherwise it could be possible that the step in this direction was too far and the loss function increases. [9, pp. 1-7]

How convolution works is illustrated in figure 5. On the left side, a so-called kernel is defined. By applying the kernel to the input matrix we obtain another matrix as the result.



By overlapping the kernel and the input matrix, multiplying each of the values and summing them up, we calculate the values of the right matrix. Afterwards, the kernel is shifted along the input matrix until every possible position was visited. Typically, different kernels are used in CNNs. In this example the kernel is moved to the next pixel. In this case the stride is 1. By setting the stride to 2, the kernel would skip one pixel location. [9, pp. 11-13]

If a kernel or ‘filter’ sees a feature at a spatial region, higher values are found in the output matrix at this pixel location and an activation for feature detection took place [10, p. 6]. Features could be edges and in the deeper layers of the network even complex patterns [9, p. 14].

Afterwards, the new output is the input for the ReLu layer. The Rectified Linear Unit is applied on every element in the input and is just the maximum of zero and the value of this element. As a result, the non-linearity of CNNs growths. This needs to be done because images are highly non-linear as well. If a pattern is not recognized in a region, the values are zero or negative. Applying ReLu will transform this values to zero and the new value will be only greater than zero if there is an activation for a specific pattern at this region. This operator requires no parameter. [9, pp. 10-11]

In the next step, the pooling layer reduces the complexity, the number of parameters and the dimension. Often a max-pooling layer is used with a kernel of the size 2x2 and stride 2. As a consequence, four values are reduced to just a single value. [10, p. 8]

Finally, the last layers are fully connected layers, this means that there is a connection between all neurons from the previous layer and all neurons from the current layer. Moreover, in case of image recognition the high-level features, as an output of convolutional and pooling layers, are used to classify the input. Mostly, it is also a simple method to combine these features in a non-linear way. [11]

NVIDEA’s PilotNet

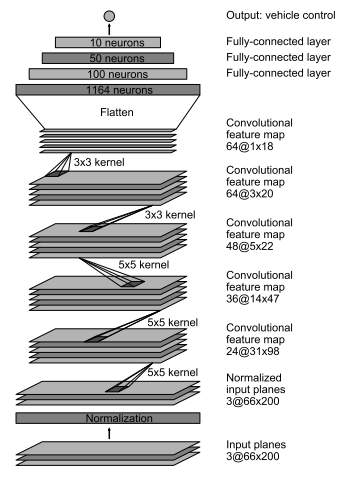
After introducing the basic principles of CNNs, this chapter will focus on the method of NVIDEA, later called PilotNet [12], that uses CNN to determine the steering angle for an self-driving car only by the images of a front camera [8].

Figure 7: []

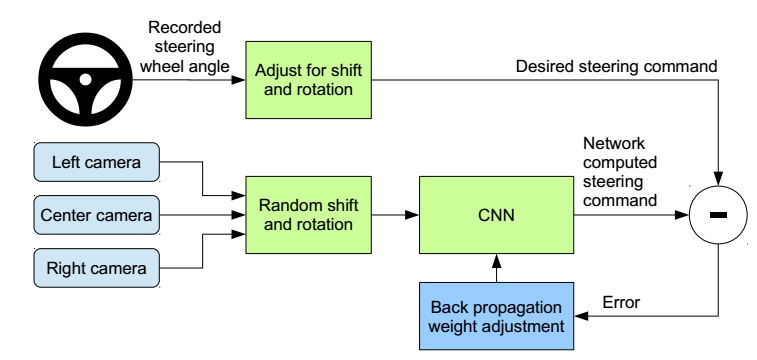
 In order to train the network, the steering angle of the vehicle needs to be recorded as well as the images of the front camera. To achieve independence from the steering system of the vehicle, the steering command is represented by with as the turning radius in meter. In contrast to use as the steering command, this approach involves the advantage of avoiding a singularity when driving a straight. Furthermore, it is essential that the network learns how to react after mistakes and guides the vehicle to the center of the lane again instead of slowly leaving the road. For training purposes, additional videos are recorded by the left and right front camera. These images are used to displace the vehicle from the center of the lane and to rotate it such that the vehicle direction does not correspond to the road direction anymore. Since the vehicles is moved and turned, the steering angle does not fit to the images. Consequently, the steering angle is adopted in a way that it would guide the vehicle to the center of the lane and with the correct orientation within two seconds. Using backpropagation of the Torch 7 machine learning package determines the parameter of the CNN. The procedure of training can be seen in figure 6.

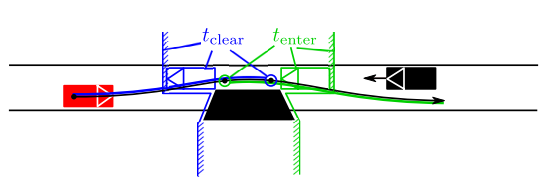
Figure 6 []

Once the network is trained, only one front camera is necessary. According to the authors, the training data was collected under different weather circumstances including good conditions but also from foggy, snowy situations or during the night. The drivers were told to drive as usual but with high alertness. As labels for the training data are the road type, the weather conditions and the driver’s activity used. The latter could be switching or following a lane or turning.

During the training the network should minimize the deviation between the computed steering angle and the desired steering command. In total, the network exists of one layer for normalization, five convolutional and three fully-connected layers, see figure 7. Three of the convolutional layers have a kernel and a stride. The other convolutional layers are non-strided and use a kernel. The configuration of the layers is reasoned with experience from different experiments. Finally, the fully-connected layers calculate a steering command.

 Before on-road tests were conducted, a drive simulator was used to validate the network. The variable is used to assess the performance and is defined by:

This definition assumes that it takes a driver six seconds to guide the car to the center of the line and an intervention occurs if the vehicle is more than one meter from the center of the lane away. According to [8] PilotNet achieved a of 98 %. Additionally, it is shown in [12] which features or elements in an image mostly influence the steering angle. The results show that the network is able to learn important features by itself. More precisely, it learned to detect the boundaries of the road or parking vehicles at the side of the road, but the system was never told to learn to detect these features. Moreover, these features contribute the most to the steering angle.

Evaluation of the Different Approaches

After introducing two different approaches for the determination of a steering angle for a self-driving car, this paragraph of the paper focuses on the evaluation of both. More specifically, priority is given to the individual advantages and disadvantages of the approaches. Moreover, the approaches will be assessed how well they achieve the overall goal to keep the vehicle on a lane or make the vehicle following a desired path. Conclusively, the inclusion in the complete system architecture of an autonomous vehicle will be considered.

Figure []

Evaluation of Modules within the Typical Hierarchical System Design of Autonomous Vehicles

Hierarchical system designs of self-driving cars consist of a path or trajectory planning module and a trajectory tracking module. The first one creates the desired path for the latter one. At first, the methods for both of these tasks are evaluated separately and afterwards an assessment of both modules in the hierarchical system design will be done.

**Evaluation of the Trajectory Planning for Bertha**

The introduced method for trajectory planning by [4] is applied on the Bertha-Benz-Memorial-Route, which connects two cities in Germany and has a total distance of 100 km.

The authors themselves concluded that this method could handle all the problems during the drive. For instance, the system reduces the speed in sharp turns or in tight bends. For the passengers, the driving style imitated a natural behavior and seemed to them very pleasant. In spite of this, the authors highlighted some limitations of the method. Problems in the context of combinatorial aspects occur. Given these aspects just little importance at the beginning of the project, the authors figured out that combinatorial skills are necessary to tackle all the problems during a drive. Figure X shows a situation in which these skills must be used. Due to the obstacle the street is narrowed. The local method has calculated the trajectory, the black curve, but it is important to consider the other vehicle in black. The own vehicle, in red, can either enter and pass the obstacle before the other vehicle enters the section or after the other vehicle passed it. This means that two discrete classes of solution exist and according to the authors the local method is not able to guarantee that it will choose the correct class. Additionally, it is possible that the method does not find a way out of an invalid state. During the drive, ad hoc decisions for one of the classes were made to solve these problems, considering different measurements. Further work needs to be done, however, on these discrete decisions that were also used for merging. To handle all the different situations during a drive, a combinatorial method must complement the local method. [4, p. 457]

The discussion of the authors of the local trajectory planning method shows that the proposed method is not perfect. Even if they claim that the method works excellent in most of the scenarios, in which no combinatorial decision is needed, handling the critical situations well is crucial for the success of autonomous vehicles and for the safety of every involved person. Path planning in form of solving an optimization problem can be a computational expensive task because the solution space can be very large, and it should be prevented that the methods gets stuck in local optima. Setting up the constraints for an optimization problem in such a way, like it is done in [4], that the only optimum is the global optimum needs a lot of expertise and experience in this field of research. However, once it is ensured that this is achieved, the optimization problem can be solved in short period of time. This is essential for programs that have to run within a fraction of a second and must ensure that it finds a good solution within this time. Neglecting the critical situations in which combinatorial decisions are needed, the introduced method is successful in most of the driving scenarios. Planning the path and considering the velocity is very complex but useful to avoid accidents. The different constraints of the method also result in smooth and natural trajectories. This is significant because people have to trust the autonomous vehicle, and no one will enter a car that acts strange from the passenger’s perspective and conveys a insecure feeling.

**Evaluation of the Trajectory Tracking**

For tracking the desired trajectory, a method by [5] using the principle of Immersion and Invariance was introduced. The goal of the controller is to keep the vehicle as close as possible to the desired path and to minimize the deviation between the angle of the road and the yaw angle of the vehicle.

Simulation results provided by the authors show that the controller performs well in different kind of situations. However, different steps of improvement were necessary to achieve this level of performance. The last version of the controller, limits the lateral error from the desired path to less than 5 cm for two specific driving scenarios. In the first scenario the speed is constant (13.5m/s) and the maximal lateral acceleration is 4m/s2. The latter has a huge influence on the dynamic behavior of the vehicle. Until a value of 3 to 4m/s2 the vehicle dynamics are linear and the equations for the used vehicle models are still valid. If the lateral acceleration exceeds this value, it is not possible to describe the vehicle dynamics with relative simple equations. The behavior is more unpredictable, and the vehicle models loses validity and the controller could struggle with keeping the vehicle on the road. Other controllers, like most of the model predictive controllers, have to be designed for a certain velocity and deviations from the desired path increase when the actual velocity differs more from the initial velocity. Because of these two reasons, the second test track is simulated with a velocity between 5 and 25m/s and a maximal lateral acceleration of 5 m/s2.

Evaluation of End-to-End Learning

Systems using End-to-End Learning seem very promising because after setting up the structure of the convolutional neural network it is only training data required. Though, a very large amount of data is needed to train the network and [8] has shown that is necessary to preprocess this data. Otherwise the self-driving car would not be able to come back to the middle of the lane after a mistake. Furthermore, it remains unanswered how much training data is needed to train the network sufficiently and how diverse it has to be. End-to-end learning is only useful if the network also learns to react correctly in situations it has never seen before because it is not possible to include every possible traffic scenario in the training data.

Considering the time for designing the architecture of the CNN, end-to-end learning has the potential to reduce the development time and with this the costs in comparison to typical approaches. There is no need to design or think of specific features. Moreover, CNN can learn complex features that are not very understandable from a human’s perspective. The results of [12] have shown that the network has learned to detect important features in the images by itself without being ever explicitly told what it should learn. Illustrating which parts of the image contribute the most to the resulting steering angle, e.g. lane markings or construction vehicles on the road, are comprehensible. Achieving an autonomy value of 98 % during the tests of NVIDEA is a good result. But this also means, according to their definition of the autonomy value, that it is possible that the deviation from the center lane could be almost one meter in a lot of cases. In the case of autonomous vehicle, high accuracy is required and deviations up to one meter are much too high and could cause lethal accidents. Furthermore, it should be verified how accurate CNNs can be when they learn from human drivers which are not capable of keeping the vehicle exactly in the middle of the lane or very close to the desired path. The accuracy of PilotNet could be a critical disadvantage, though further information of how well the system performs are necessary. Nevertheless, the results have shown that it is possible to build and train CNNs in such a way that they can keep the lane and calculate an appropriate steering angle. According to [spectrum.ieee], a further advantage of end-to-end learning is the robustness of the system against sensor failure.

Regarding to the architecture of the network, it is still very difficult to choose a good design and thus expert knowledge is required. So far, no tool exists to determine the optimal structure of the network and once the design is chosen, there is no flexibility and only the parameters can be trained. The training itself needs a lot of computational power and for this expensive GPUs are necessary. Another aspect to mention is the black box behavior of CNNs. It is not easy to understand how a convolutional neural network works in detail and why it e.g. computed this particular steering angle. This is especially for debugging purposes a disadvantage. Additionally, there are no safety standards defined yet, which are necessary to guarantee a safe behavior during a drive.

Is whole system considered?

Margins

Papers **must** be printed in two-column format on 8-1⁄2 x 11 inch paper. The margins if you use 8-1/2 x 11 inch paper must be **exactly** as follows:

• Top margin: 3⁄4 of an inch

• Left margin: 3⁄4 inch

• Right margin: 3⁄4 inch

• Bottom margin: 1-1⁄4 inches

The actual image size for your paper (minus margins) is 7

x 9 inches (17.78 x 22.86 cm). Papers that deviate from these measurements will not be published.

Fonts

All fonts must be embedded for your paper to be published. No exceptions.

References

Engelmore, R., and Morgan, A. eds. 1986. *Blackboard Sys­tems.* Reading, Mass.: Addison-Wesley.

References

[1] B. Paden, M. Cap, S. Z. Yong, D. Yershov, and E. Frazzoli, “A Survey of Motion Planning and Control Techniques for Self-driving Urban Vehicles,” in vol. 1, *IEEE Transactions on Intelligent Vehicles*, 2016, pp. 33–55.

[2] D. J. Fagnant and K. Kockelman, “Preparing a nation for autonomous vehicles: Opportunities, barriers and policy recommendations,” *Transportation Research Part A: Policy and Practice*, vol. 77, pp. 167–181, 2015.

[3] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, *End to end learning for self-driving cars.* [Online] Available: https://arxiv.org/pdf/1604.07316.pdf. Accessed on: Nov. 08 2017.

[4] J. Ziegler, P. Bender, T. Dang, and C. Stiller, “Trajectory planning for Bertha — A local, continuous method,” in *2014 IEEE Intelligent Vehicles Symposium (IV)*, MI, USA, pp. 450–457.

[5] G. Tagne, R. Talj, and A. Charara, “Immersion and invariance control for reference trajectory tracking of autonomous vehicles,” in *2013 16th International IEEE Conference on Intelligent Transportation Systems - (ITSC 2013)*, The Hague, Netherlands, pp. 2322–2328.

[6] J. M. Snider, *Automatic Steering Methods for Autonomous Automobile Path Tracking*. Pittsburgh, 2009.

[7] R. Rajamani, *Vehicle Dynamics and Control*. Boston, MA: Rajesh Rajamani, 2012.

[8] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, *End to End Learning for Self-Driving Cars.* [Online] Available: https://arxiv.org/pdf/1604.07316.pdf. Accessed on: Nov. 13 2017.

[9] Jianxin Wu, *Introduction to Convolutional Neural Networks.* [Online] Available: https://cs.nju.edu.cn/wujx/paper/CNN.pdf. Accessed on: Nov. 14 2017.

[10] R. N. Keiron O'Shea, *An Introduction to Convolutional Neural Networks.* [Online] Available: https://arxiv.org/pdf/1511.08458.pdf. Accessed on: Nov. 14 2017.

[11] Ujjwal Karn, *An Intuitive Explanation of Convolutional Neural Networks.* [Online] Available: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/. Accessed on: Nov. 14 2017.

[12] Mariusz Bojarski, Philip Yeres, Anna Choromanaska, Krzysztof Chromanski, Bernhard Firner, Lawrence Jackel, Urs Muller, *Explaining How a Deep Neural Network Trained with End-to-End Learning Steers a Car.* [Online] Available: https://arxiv.org/pdf/1704.07911.pdf. Accessed on: Nov. 13 2017.