Comparison of Hierarchical System Designs and End-to-End Learning for Autonomous Vehicle

Christoph Franzke
cfranzke@uwaterloo.ca

Abstract

Autonomous driving is a very complex task and many different approaches are deployed in order to solve these challenges. The typical system design of autonomous vehicles is hierarchical, and each layer provides the information needed for the next one. As a very important part of this architecture, trajectory planning determines the desired path for the vehicle and the trajectory tracking ensures that the vehicle follows this path with a minimal deviation. Latest research in this area introduce an alternative approach using end-to-end learning. A convolutional neural net is trained by pictures of the front camera and after the training is done the steering angle is determined only by the pictures generated by the front camera.

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 and within this domain a certain 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. After describing the application domain and the problem more detailed, different methods of the typical hierarchical system design of an autonomous vehicle are introduced. Especially, trajectory planning and trajectory tracking are emphasized. Afterwards, an end-to-end learning approach by NVIDEA is explained and eventually a comprehensive comparison between a hierarchical system design and end-to-end learning for autonomous driving completes this paper.

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 until the first fully autonomous vehicle is licensed.

Two previous 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 foundation 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. [3, p. 34, 3]

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 change 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 regarding licensing, security, insurance regulations and many more. [4, pp. 167-168]

Problem Description

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 possible 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 from the desired route. [3, 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 like it is done by NVIDEA [5]. 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.

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 proposed in the literature. A comprehensive overview over the different approaches cannot be presented within this project and thus it is recommended to use special surveys like [3]. 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 in this case 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 for the passengers. [3, 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 this problem. 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 in such a way that a graph is generated, and a minimum cost path can be found. Often variants of A* are used. In contrast to this, 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. [3, pp. 40-48]

Trajectory Planning for Bertha

After introducing trajectory planning in general, this paper will present a specific approach for trajectory tracking and look at the implementation in a more detailed way. The approach is designed by Ziegler, Bender et al. and is described in [2]. This local and continuous method is designed in a way 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 trajectory. As one advantage it 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 $\mathrm{x}(t) = (x(t), y(t))^T$, the curvature of the trajectory and the tangent angle by $\kappa(\mathrm{t})$ and $\psi(\mathrm{t})$ respectively. Both can be calculated with the derivates of the trajectory at time t [2, p. 451]. By minimizing the following integral

$$J[\mathrm{x(t)}] = \int_{t_0}^{t_0+T} L(x, \dot{x}, \ddot{x}, \dddot{x})dt,$$

with

$$\mathrm{L} = \mathrm{j_{offs}} + \mathrm{j_{vel}} + \mathrm{j_{acc}} + \mathrm{j_{jerk}} + \mathrm{j_{yawr}}.$$

an optimal trajectory can be found for the current time t_0 and the time horizon T. Every summand i in L contains an individual weighting factor w_i. The offset is considered by

$$\mathrm{j_{offs}(x(t))} = \mathrm{w_{offs}}|(\mathrm{d_{left}(x(t))} + \mathrm{d_{right}(x(t))}|^2.$$

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 d_{left} and d_{right} are signed and points left and right of a bound are positive and negative respectively. The next term

$$\mathrm{j_{vel}(x(t))} = \mathrm{w_{vel}}|\mathbf{v}_{\mathrm{des}}(x(t)) - \dot{\mathrm{x}}(t)|^2$$

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 d_{left} and d_{right}. 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

$$\mathrm{j_{acc}(x(t))} = \mathrm{w_{acc}}|\ddot{\mathbf{x}}(t)|^2$$

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

$$j_{jerk}(x(t)) = w_{jerk}|\ddot{x}(t)|^2$$

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 $\dot{\psi}$:

$$j_{yawr}(x(t)) = w_{yawr}\dot{\psi}(t)^2$$

The optimal trajectory not only minimizes the integral of L but also must hold several constraints. According to [2], 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:

$$-\kappa_{max} \leq \kappa(t) \leq \kappa_{max}$$

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

$$\|\ddot{x}(t)\|^2 \leq a_{max}^2.$$

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 d to a polygon p. 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 p_1 and p_2 be two neighboring corner points of a polygon p and t_1 and t_2 the tangent vectors at this corner points, then a linear segment of a polygon is represented by $G = (\overline{p_1 p_2}, t_1, t_2)$. For every point

$$p_\lambda = \lambda p_1 + (1-\lambda)p_2$$

between the corner points ($\lambda \in [0,1]$), a tangent vector is defined by

$$t_\lambda = \lambda t_1 + (1-\lambda)t_2.$$

The value λ is calculated, see figure 1, such that the pseudo tangent t_λ and the pseudo normal vector $n_\lambda = x - p_\lambda$ are orthogonal and hence

$$n_\lambda t_\lambda = 0.$$

Furthermore, it is admissible to assume $p_1 = (0,0)^T$ and $p_2 = (l,0)^T$, thus $t_1 = (1,m_1)^T$ and $t_2 = (1,m_2)^T$. Then λ is calculated by

$$\lambda = (m_1 y + x)/(m_1 - m_2)y + l.$$

Finally, the pseudo distance d is determined by the length of the vector n_λ if $y > 0$ 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.

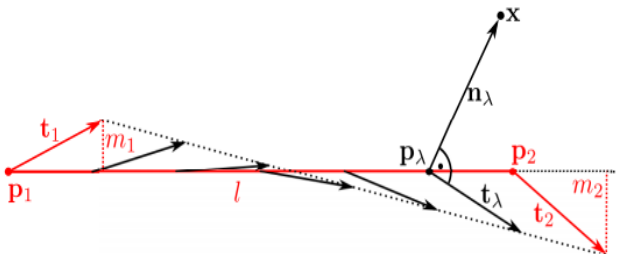

Figure 1: Calculation of the Pseudo Distance [1].

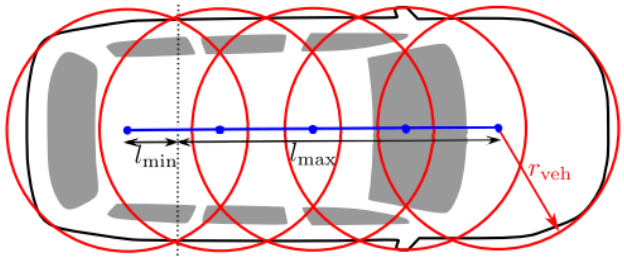

Figure 2: Representing the Shape of a Vehicle with Circles [2].

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 k circles with a radius of r_{veh}, see figure 2. Let $C_{maneuver}$ be the set of circle center points and P_i the set of all polygons, then the following constraints must be satisfied:

$$d(p, x_c) \geq r_{veh}, p \in P_i, x_c \in C_{maneuver}$$

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 [3, 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 [6, p. 2322]. A comprehensive, but not all-embracing survey of in the past used controllers is provided in [3].

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 [7, pp. 28-31].

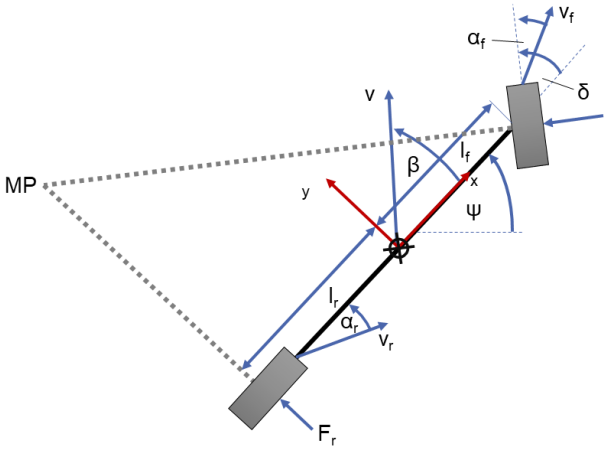

Figure 3: Dynamic Bicycle Model.

Due to the tremendous complexity of the dynamics and movement of vehicles, it is necessary to make certain assumptions. Important assumptions 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

$$m(\ddot{y} + v_x\dot{\Psi}) = F_f\cos(\delta) + F_r$$

$$\text{with } a_y = \ddot{y} + v_x\dot{\Psi} \text{ and}$$

$$I_z\dot{\Psi} = l_f F_f \cos(\delta) - l_r F_r.$$

Using additional equations for the dynamics of vehicles, e.g. for the slip angle and lateral forces, Snider builds a system of equations for the lateral acceleration a_y and the yaw rate $\dot{\Psi}$,which needs to be linearized if linear control methods are applied. 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 [6]. 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 [6, p. 2322]. Due to my work experience with lateral controllers for trajectory tracking during my bachelor thesis, it can be said that 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 β and the yaw rate $\dot{\Psi}$:

$$\dot{\beta} = -(C_f + C_r)\beta/(mv_x)$$
$$-(1 + (L_fC_f - L_rC_r)/(mv_x^2))\dot{\Psi} + C_f\delta/(mv_x)$$

$$\ddot{\Psi} = -(L_fC_f - L_rC_r)\,\beta/I_z$$
$$-(L_f^2C_f + L_r^2C_r)\dot{\Psi}/(I_zv_x) + (L_fC_f)\,\delta/I_z$$

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

Moreover, the lateral error dynamics are represented by

$$\ddot{e} = a_y - a_{y,ref}.$$

The two values, a_y and $a_{y,ref}$, represent respectively the lateral acceleration of the vehicle and the desired acceleration of the vehicle. With

$$a_y = v_x(\dot{\beta} + \dot{\Psi})$$

and

$$a_{y,ref} = v_x^2\,\kappa$$

the dynamic lateral error can be rewritten as

$$\ddot{e} = v_x\left(\dot{\beta} + \dot{\Psi}\right) - v_x^2\,\kappa$$

and using the equation for $\dot{\beta}$ results in:

$$\ddot{e} = -\frac{C_f + C_r}{m}\beta - \frac{L_fC_f - L_rC_r}{mv_x}\dot{\Psi} - v_x^2\kappa + \frac{C_f}{m}\delta$$

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

$$\dot{x} = Ax + B_1\,\delta + B_2\kappa.$$

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

$$\delta = -\frac{m(K+\lambda)}{C_f}\dot{e} - \frac{mK\lambda}{C_f}e + \frac{C_f + C_r}{C_f}\beta + \frac{L_fC_f - L_rC_r}{C_fv_x}\dot{\Psi} + \frac{mv_x^2}{C_f}\kappa$$

with two new parameters K and λ. The angle β, the curvature κ, the yaw rate $\dot{\Psi}$ and the longitudinal velocity v_x are measured. The constants C and L must be determined but can be obtained by test drives or a single measurement. The lateral error e will be calculated by the current position of the vehicle and the desired position on the trajectory. Additionally, the derivative $\dot{e}$ is obtained by the formula, see [8, p. 34],

$$\dot{e} = v_y + v_xe_\Psi$$

with e_Ψ 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 K and λ are determined by simulation or testing.

Given this equation, it is possible to calculate a steering angle 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. [9, p. 2]

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

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.

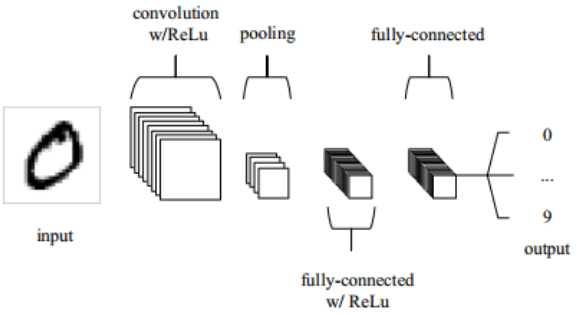

Figure 4: A Simplified Architecture of CNN [1].

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 kind of input. More specific, the input is an order 3 tensor. For an image with $H \times W$ pixels the input is a tensor with H rows, W columns 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 w_i exists, then the update rule for this parameter is

$$w_i \leftarrow w_i - \alpha(\partial z/\partial w_i)$$

with the learning rate α and the loss z. 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. [10, 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.

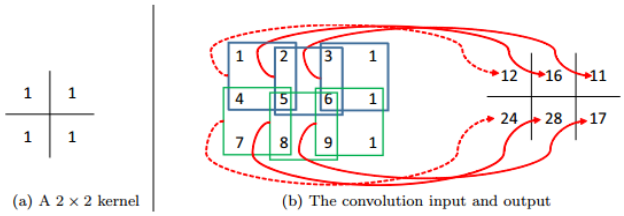

Figure 5: The Process of Convolution [10].

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. [10, 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 [1, p. 6]. Features could be edges and in the deeper layers of the network even complex patterns [10, 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. [10, 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. [1, 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 [9].

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 $1/r$ with r as the turning radius in meter. In contrast to use r 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. Backpropagation of the Torch 7 machine learning package is used to determine the parameter of the CNN. The procedure of training can be seen in figure 6.

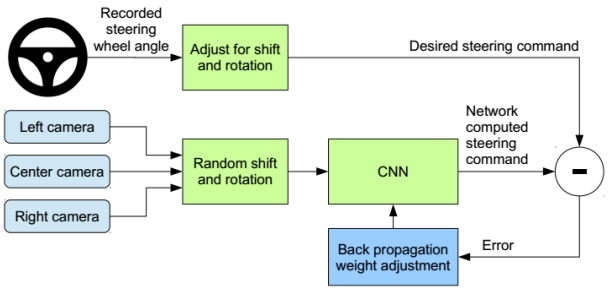

Figure 6: Scheme of the Training Method [9].

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. Labels for the training data are the road type, the weather conditions and the driver's activity. 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 5×5 kernel and a 2×2 stride. The other convolutional layers are non-strided and use a 3×3 kernel. The configuration of the layers is reasoned with experience from different experiments. Finally, the fully-connected layers calculate a steering command.

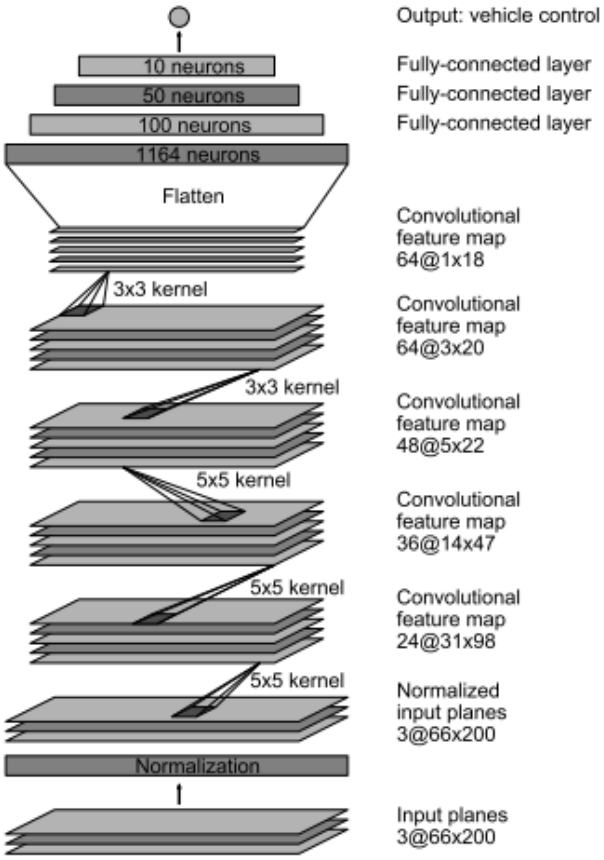

Figure 7: Structure of the CNN [9].

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

$$autonomy = \left(1 - \frac{(number\ of\ interventions) \times 6\ seconds}{elapsed\ time}\right) \times 100$$

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 [9] PilotNet achieved a $autonomy$ 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. The approaches will be assessed on 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.

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

Hierarchical system designs of self-driving cars consist not only but also of a path or trajectory planning module and a trajectory tracking module. The first one creates the desired path for the latter one. In this chapter both introduced methods are evaluated.

Evaluation of the Trajectory Planning for Bertha

The introduced method for trajectory planning by [2] 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 8 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. [2, p. 457]

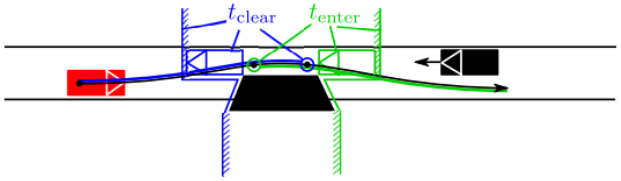

Figure 8: Illustration of a Critical Section [2].

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 method gets stuck in local optima. Setting up the constraints for an optimization problem in such a way, like it is done in [2], 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 a 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 an insecure feeling.

Evaluation of the Trajectory Tracking

For tracking the desired trajectory, a method by [6] 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 [6, pp. 2326-2327] 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 5cm for two specific driving scenarios. In the first scenario the speed is constant (13.5m/s) and the maximal lateral acceleration is 4m/s^2. The lateral acceleration has a huge influence on the dynamic behavior of the vehicle. Until a value of 3 to 4m/s^2 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 5m/s^2. Achieving a deviation less than 5cm in both cases is a very good result. Controller using different methods lead to a higher deviation on easier tracks, compare [7]. Different tests were conducted by the authors to proof the robustness of the controller. Several parameters of the final control equation are vehicle specific, not easy to determine and can vary during a drive. But even if these parameters vary in a certain range, the controller achieves a good performance. Finally, even simulations with a lateral acceleration of 8m/s^2 does not lead to a poor performance of the controller. [6, pp. 2326-2327]

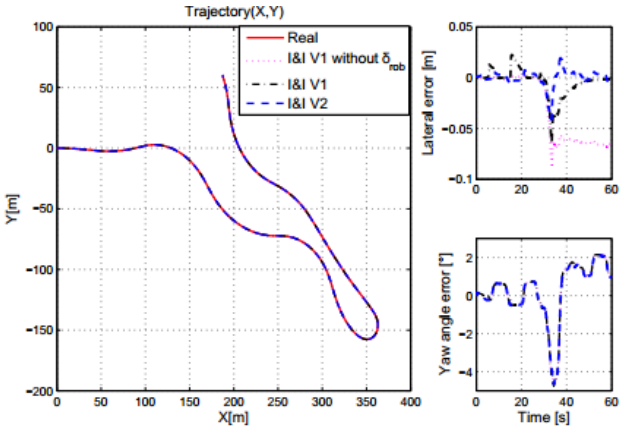

Figure 9: Simulated Test Track and Results [6].

The simulation results show that the controller is capable of guiding the vehicle along the desired path with a very high accuracy. Even in extreme situations the results are still good. However, the controller is just tested in simulations and not with a vehicle on an actual test road. The authors mention that this will be done in future work. It can be assumed that these tests will show a higher deviation from the desired path caused by e.g. the actuators. In general, the introduced method requires a comprehensive knowledge about control theory and a lot of experience in this field. The way the final control equation is achieved is very complex and finding the proper values for the parameters within this equation is a difficult task.

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 [9] 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, CNNs 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 large 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 vehicle on the lane and calculate an appropriate steering angle.

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.

Comparison of the Different Approaches

Finally, the typical hierarchical approach with a separated module for trajectory planning and trajectory tracking is compared with the latest approach of end-to-end learning.

Regarding the complexity of each of the approaches, both demand a comprehensive knowledge in their field of research and setting up the modules is very complex. In contrast to end-to-end learning, the use of separated modules with explicitly defined functions allows improvements of the system easier. The system is not like a black box and weaknesses of the methods can be found with average effort. For instance, both the method for trajectory planning and trajectory tracking figured out disadvantages of their versions and how they can improve them. Using a CNN, it needs a lot of effort just to understand, or try to understand, why the network calculated this specific steering angle and which objects contributed the most. Furthermore, the hierarchical approach covers more problems that occur during autonomous driving. For example, the trajectory tracking module determines a desired velocity in order to keep the vehicle on the road or to avoid accidents with other traffic participants. How the velocity is or will be considered in end-to-end learning is unclear. Will a second network be used and train to follow the desired velocity? Or will the velocity and the desired velocity be an additional input of the current network?

The parameter used for trajectory tracking vary from vehicle to vehicle and even from drive to drive. This makes the process of reaching the goal to follow the desired path more complicated and can lead to different deviations. Additionally, a vehicle model is needed in the introduced approach. For trajectory tracking and planning certain assumptions have to be made, a lot of restrictions considered and details be defined explicitly. On the other hand, end-to-end learning has the advantage that it does not have to deal with vehicle parameters and can avoid thus these difficulties. There is no underlying vehicle model necessary or no constraints have to be defined explicitly. In general, end-to-end learning does almost not require any knowledge about the vehicle and its dynamics. There are no experts needed for optimization or control theory, only for CNNs. This is the reason why end-to-end learning seems to be so promising. Forgetting about a lot of difficult aspects, once the network structure is set up and the network is trained, all the work is done. Implementing the network seems to be less effort than designing the different modules of the hierarchical approach. But so far, end-to-end learning neglects some important tasks that are necessary for autonomous driving and it is not obvious how they can be implemented in CNNs or the overall system. The end-to-end learning system of NVIDEA is only able to keep the vehicle on the middle of the lane and to avoid obstacles. More critical situations, like the one in figure 9, where two vehicles approach an obstacle from both lanes, are not trained yet because the velocity of the other vehicle is not considered. It is not ensured, that there will be no collision when both vehicles pass the obstacle. Because of the isolation of the system, the surrounding of the vehicle is not perceived and e.g. vehicles from behind are not considered. This could cause problems as well as untypical behavior of other vehicles or pedestrians. Is the CNN trained well enough to avoid an accident if another car from the opposite lane slowly drifts on the own lane? How will the network react in case an accident is not avoidable? At least the first question can be answered positively for the hierarchical approach. The sensors perceive the movement of this car and consider it in the path planning. It is more likely, that this situation will not result in an accident.

At last, the performance of the different approaches has to be evaluated. The trajectory planning module struggles with critical situations in which combinatorial decisions are necessary, but it can deal with this kind of situations at least. The path planning module is almost universal and designs smooth and natural trajectories. The trajectory tracking showed excellent results in the simulations, even at highly dynamic and nonlinear behavior. Nevertheless, tests with a vehicle on an actual test track are essential for a final evaluation. The fact that for the tasks of planning a trajectory and tracking a trajectory, a lot of different approaches are currently used, indicates that none of the introduced methods perform well enough. PilotNet of NVIDEA is able to keep the vehicle on the middle of the lane and avoids obstacles in most of the situations. Considering that the network is trained by data from human drivers, it is very likely that the lateral error from the middle of the lane is much larger than 5cm. In general, it should be questioned if it is the right goal to imitate a human driver on the road when they cause accidents, are inaccurate and machines can perform better than humans in this case.

Conclusion

Within this work different approaches to determine the steering angle for a self-driving car were compared. On the one hand, a typical hierarchical approach, consisting of trajectory planning and trajectory tracking, and on the other hand, a method that uses CNNs and end-to-end learning was introduced. End-to-end learning has the promising advantage that only comprehensive knowledge about CNNs is needed and no expertise about vehicle dynamics or optimization is required. Additionally, no assumptions or constraints have to be defined explicitly. Almost like a magic black box, the steering angle is obtained by the images of

the front camera and the trained network. Training the network properly is very difficult, though, and it is not clear if every possible situation will be handled correctly by the network. Also, it is necessary that the extend surrounding of the vehicle is perceived and considered by the module to avoid accidents. The hierarchical approach already takes this into account and covers more of the difficult tasks during an autonomous drive. Both approaches have shown that they are not sufficient to design a self-driving car which does not need a security driver anymore. To reach complete autonomy, further research needs to be done in both areas.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

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

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

[9] 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.

[10] Jianxin Wu, *Introduction to Convolutional Neural Networks.* [Online] Available: https://cs.nju.edu.cn/wujx/paper/CNN.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.