2019 IEEE Intelligent Transportation Systems Conference (ITSC)
Auckland, NZ, October 27-30, 2019

Online Path Generation from Sensor Data for Highly Automated Driving Functions

Tim Salzmann[1], Julian Thomas[2], Thomas Kühbeck[3], Jou-ching Sung[3], Sebastian Wagner[1] and Alois Knoll[1]

Abstract— **State-of-the-art autonomous driving systems rely on high precision map data. These map data are crucial to the driving function and therefore need to be validated during drive time. This work describes a probabilistic neural model inferring information about the road in front of an automated vehicle from sensory data. This problem is modeled as a pixel-wise classification problem. Thereby, the limitations of systems relying on pre-processed map data are overcome by replacing navigation related map data with online sensor information. The proposed model is trained based on recorded driving traces and allows the facilitation of the vehicle odometry as input label. Two use cases, namely *Path Planning* and *Map Validation* are presented and evaluated.**

Index Terms— **Path planning, Machine learning, Decision making, Autonomous systems**

I. Introduction

Toady's developed autonomous driving systems [1] [2] are often designed to rely on prior knowledge, namely accurate maps and precise localization. However, the accuracy and validity of both cannot always be guaranteed. The map can become outdated due to road changes followed by failure in localization based on inaccurate GPS measures. In most applications localization is performed sensor based by triangulating map features, therefore changes in the environment or weather-related degradation of the used sensor can lead to inaccurate extracted information. This degradation of information leads to two problems.

- Overall unavailability decreases the functional behaviour.
- Inaccuracies leading to wrong or misbehaviour of the automated driving function.

The second more complex problem demands the maintenance of the functionality of the automated driving function until the prior knowledge is available again or the car has come to a safe stop. For these problems, the requirement for an online road model from sensor data is inevitable in order to sustain an overall safe function behavior.

Pixel-wise segmentation and deep learning facilitate many online road model approaches which have been developed during recent years to segment camera images for the generation of road models [3] [4]. A large number of these algorithms rely on camera images as input signal which are subsequently segmented into drivable and not drivable space by algorithms. Cameras, however, being passive sensors are very sensitive to environment conditions. Caltagirone et al. [5] therefore moved the problem of road segmentation from an image to a grid space. They use a single grid representation of a LIDAR point cloud to detect free space around the vehicle. However, they train as well as evaluate solely on the human labeled KITTI dataset [6] (less than 300 data points). The result is a spatial representation of drivable space in the vehicle plane. However, no information on where the vehicle is allowed or should drive is inferred. Further, state-of-the-art algorithms in sensor based navigation rely on end-to-end approaches [7] leveraging the advantage of freely-labeled training data during drive-time. Extending this, Barnes et al. [8] use label efficient semi-supervised data generation to label the ego-vehicle path. While multiple sensors (camera and LIDAR) are used for automatic labeling, prediction again is exposed to the uncertainty of a single camera.

Definitions: In this work, knowledge about the road is also referred to as a road or path model as it represents the road and/or drivable paths in an abstracted way. Paths incorporating a width are called (driving) tubes. Navigating an automated driving system using mainly offline defined knowledge is referred to as global or high prior knowledge navigation information [9]. A typical manifestation of such is an HD (High Definition) map. In contrast, the term low prior knowledge if referred to if only offline defined routing information is used which itself would not be sufficient for navigation. Furthermore, this work is depicted along the terms of scenes and situations in autonomous driving as defined by Ulbrich [10].

Statement of Contributions: The contribution of this paper is twofold. First, a new approach to path planning is presented which uses a Fully Convolutional Network (FCN) inferring multiple possible paths for the vehicle based on a variety of input features. For robustness, it relies on fused data from multiple sensors. The FCN can be conditioned on low prior knowledge to only infer the most suited path based on a defined goal. The training data is automatically generated during driving without additional human labeling effort. Second, we demonstrate the usefulness of such a model illustrating post-processing algorithms for the FCN output for sensor based (and possibly map influenced) path planning and map validation.

Organization: In Section II the design and training of neural models are explained which infer the knowledge about the road environment. Possible input features for such models are presented. Further post-processing algorithms using the model output towards certain use cases are portrayed. Sub-

Department of Informatics, Technical University of Munich, Germany[1]
BMW Group, Munich, Germany[2]
BMW Technology Office, Mountain View, CA, USA[3]
E-Mail: {Tim.Salzmann, Sebastian.Wagner}@tum.de, Knoll@in.tum.de
{Julian.Thomas, Thomas.Kuehbeck, George.Sung}@bmw.de
This work was supported by the Technology Office in Mountain View, CA of the BMW Group.

sequently, the result is presented and evaluated in section III. Then conclusions are drawn and future work is suggested in Section IV.

II. Approach

Independent of the use case the general approach of this work consists of three steps:

1) Pre-processing of raw sensor data being fused and derived into a grid representation.
2) FCN model is designed and trained for the desired use case.
3) Post-processing generates the output of the FCN path model for the respective use case.

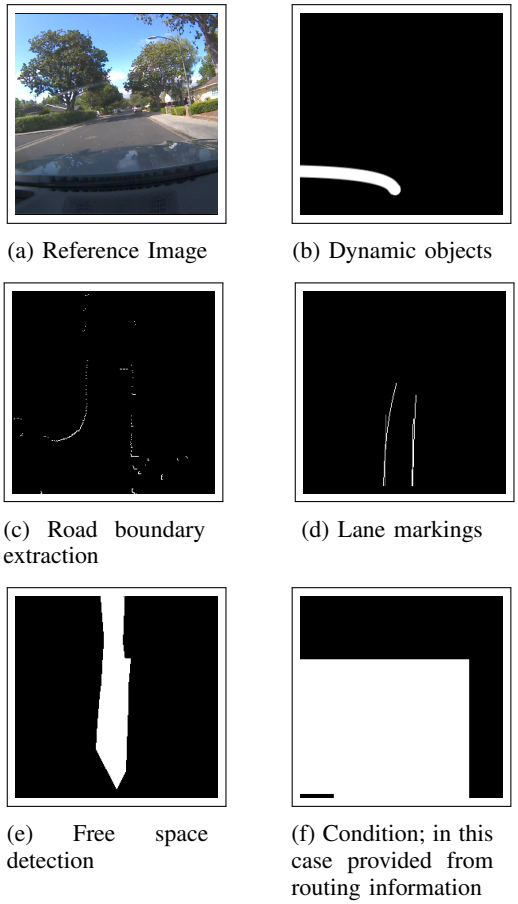
Accordingly to these topics, this section is structured starting with depicting the pre-processing steps which result in a variety of possible input grids. These generalize over all use cases. Then the model architecture, as well as the training procedure, is explained. Finally, use case specific post-processing steps are explained.

A. Pre-Processing

One of the main goals of this work is to increase the robustness against environmental influences including light or weather changes which influence the used types of sensors in different ways. Thus, we want to overcome one of the limitations of typically single sensor based end-to-end learning systems. Therefore, a focus is set on sensor data processing and fusion, where each input to the network can be created redundantly from at least two sensor types. The processed grids facilitate different resolution in their two axes. This enables the usage of grids of a reasonable size for all possible environments. For example, a highway environment requires more foresight in longitudinal than in lateral direction due to the higher speeds and straightened roads. This results in a decreased resolution in lateral direction (r). For proof of concept, the four following possible input grids are presented. However, all sensor data which can be represented as a two-dimensional grid around the vehicle are valid inputs to the model and may improve its capabilities. The model can be easily adapted to a varying number of inputs. Furthermore, we present a condition grid, which when fed as an additional input to the model can be seen as a conditional on the FCN model.

1) Objects Grid: A lot of information can be inferred from other road participants especially from those in front of the ego vehicle. They are included as an object grid (Figure 1b). In addition, not only the last position of an object but also its interpolated history is displayed on the grid. This results in a driving tube for each object on the road. This is achieved by keeping track of the absolute position of all objects around the ego vehicle.

2) Boundaries Grid: Other important input features are the boundaries of the road as well as static obstacles along the road. These can be extracted by any kind of active sensors including computer vision algorithms on camera images. Each sensor produces detections (for example point cloud of a LIDAR system) at every point in time. Detections which do not change their position in consecutive timesteps are assumed static and represent boundaries [11]. These static detections are transformed into the coordinate system of the ego vehicle and projected onto the grid as boolean or probability values of occupancy (Figure 1c).

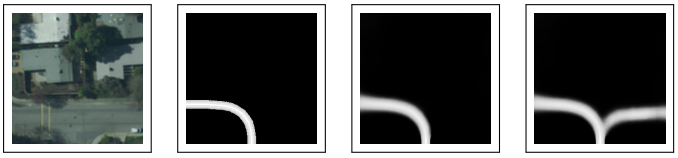


(a) Reference Image (b) Dynamic objects (c) Road boundary extraction (d) Lane markings (e) Free space detection (f) Condition; in this case provided from routing information

Fig. 1: Picture of a scene from the reference camera (a) and different extracted features depicted as grids ((b) - (f)). It can be seen in (a) that there are no actual lane markings present, the change of color on the road on the left as well as the curb on the right side are detected as markings (d). Even though such detections are noisy they can still be used. (f) shows routing information inferred from a navigation system. Though there is an option to go straight as seen in (a) the routing determines the left option as goal.

3) Lane Markings Grid: Lane markings are commonly extracted by vision systems though current research exists extracting these out of the street painting reflectivity through LIDAR [12]. Therefore, this grid also fulfills the redundancy requirement. The detected lane markings are approximated using a polynomial fit. This polynomial is then plotted onto a grid (Figure 1d). If a certain line type is detected by the camera system, it is depicted in the grid using varying intensities. For example, double solid lines are displayed with higher intensity than single dashed lines.

4) Free Space Grid: The most abstract input feature considered is a free space estimation around the ego vehicle (Figure 1e) which can be created by deep learning and



(a) (b) (c) (d)

Fig. 2: Comparison between training label grid and output grid of an (un-)conditioned FCN path model. (a) Orthoimagery for scene. (b) Label for a scene: Driving tube along the driven odometry points. (c) Output of FCN model conditioned on taking a left turn. (d) Output of unconditioned FCN model where both possible driving tubes are inferred. The blurring on the edges of the tube in (c) and (d) exemplifies the uncertainty of the model (c)

semantic segmentation on a calibrated camera or by extracting convex surfaces from LIDAR point clouds [13]. The outcome of a free space detection is a 2D plane extending the sensors field of view indicating free drivable space. This plane is represented as either points, describing the convex boundaries, or as pairs of distance d and angle azimuth ν from a fixed origin and is depicted as a 2-D surface on the grid.

5) Condition Grid: Some use cases require a certain minimum level of prior knowledge. Navigation, for example, becomes redundant if no goal for the vehicle exists. Therefore, this grid provides a way to include low prior knowledge as input to the FCN model and can be seen as a conditional to the FCN model outputting the best path conditioned on the information on this grid. It can have multiple manifestations. For example, a certain marking on the grid depicting an intermediate goal is possible. However, we had good experiences implementing this grid as over-approximated routing information. Thereby, the planned routing by a navigation system is plotted as straight lines. The width of these lines incorporates the inaccuracy of the GPS localization and the map itself (Figure 1f).

In order to train the model, a desired output for the model needs to be designed. Therefore, two additional grids are derived: The label for the training of the FCN and its output:

1) Training Label Grid: The labels used to train the model are automatically generated by keeping track of the ego vehicles odometry during data collection. This odometry trace is projected as a path, with an orthogonal gradient in intensity away from the center, into a grid. However, this leads to the training label only containing one possible driving tube. Specifically, the one which was actually driven during data collection. Therefore, this grid is called a training label and not the "ground truth" as often done in machine learning applications. For unconditioned FCNs the aim is to train models which are able to generalize to output multiple driving tubes even though the training labels only contain a single one. The gradient on the path is set discrete: For a width of 0.5 meters around the center of the vehicle, the intensity of 1 is drawn on the grid. This also represents a probability of 1 of being part of a driving tube. For the remaining width of the vehicle, a probability of 0.9 is used. Additionally, for the immediate surrounding of the vehicle of 1m the probability of 0.8 is used (Figure 2b). Thus, every pixel in the grid contains a label $L(c, r)$ resembling the probability of being part of a driving tube $y_{c,r} = 1$. The gradient is important for path extraction along the center of driving tubes for the navigation use case further described in Section II-C.2.

2) Output Grid (FCN Path Model Grid): The inferred output from the model contains possible driving tubes from the perspective of the ego vehicle. The grid values represent a probability of the pixel being part of a driving tube or not $\hat{y}_{c,r} = \{0, 1\}$. Even though the training data only consists of a single driving tube per data point, the output of the model will generalizes over all possible paths. Figure 2 shows a comparison between the training label grid and desired output grid without conditioning. While the driver during data collection took a left turn (b) all possible paths (d) for the scene are the output of the FCN if unconditioned. If conditioning is applied the condition has to be consistent with the drivers' decision and the conditioned output of the FCN would only be the driving tube to the left (c).

B. Model & Training

This problem is modeled as a pixelwise classification problem, assigning each pixel of the output grid a probability of being part of a driving tube. Naturally, there are many similarities with classic pixelwise segmentation approaches. However, there are some major differences too.

Different input features resulting out of diverse sensors and fusion techniques add to the problem space of the input vector. Furthermore, as described in Section II-A, the labels for each pixel are not binary class labels but the probabilities of these ($[0, 1]$). Furthermore, the recorded path model labels only represent part of the ground truth. If the driver took a left turn at a crossing only this driving tube is represented in the label. For unconditioned models the main difficulty is to train the model in a way that it generalizes to infer all possible driving tubes even though it was only trained on a single one per scene. Also, very sparse information is fed into the network. The information to identify a pixel as being part of a driving tube might come from a large spatial distance compared to classical image segmentation where the information necessary to classify a pixel on the output is normally given in close proximity around the corresponding pixel location in the input image.

Two concepts, downsampling, and dilation, originating from pixelwise segmentation are transferred to this problem. The most popular implementation of the first concept is called U-Net [14]. It is originally used for biomedical image segmentation and is designed to segment a single input image with one or three (colored) channels. However, in this work, a segmentation of the environment around the vehicle is inferred from multiple input grids. For the FCN path model all of these grids are single channel and have the same size ((b) - (f) from Figure 1). Therefore, the U-Net architecture is altered (Figure 3) to have multiple downsampling feature lines which get concatenated at the maximal depth. The

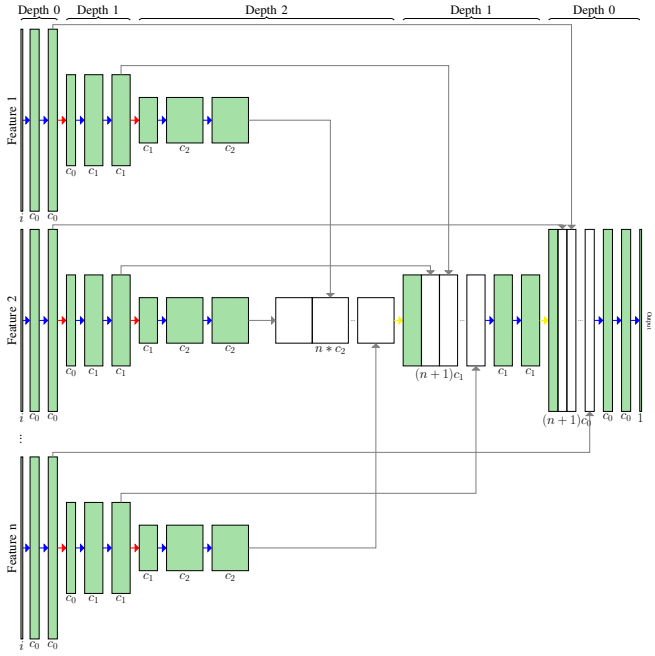


Fig. 3: Adapted U-Net architecture with multiple feature lines. Arrows: blue - Convolution (or Parallel Dilated Convolutional layer as shown in Figure 4), red - Downsampling, yellow - Upsampling, grey - copy and concatenate. c_x: number of convolution filters on depth x.

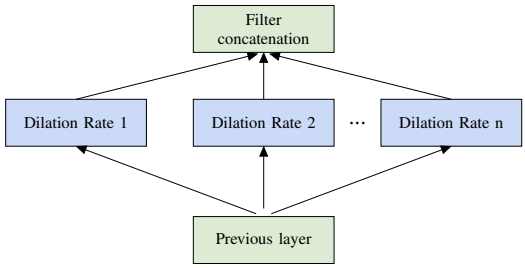


Fig. 4: Parallel dilated layer parallelizing convolutions with different dilation rates. (Adapted from [15])

single upsampling line is adapted so that on each depth level features in the original resolution are bridged (grey arrows) from each feature line.

This concept of downsampling is combined with the concept of dilated filter masks. Without dilation, the adapted U-Net can only increase the receptive field by increasing depth which implies an increase in the size of the network. Dilated convolutional masks are a way to decrease the size of the network while keeping its functionality. To overcome the limitation of fixed dilation rates, a new type of convolutional layer introduced in [15] is used. Similar to the inception layer [16], which uses parallel convolutions with different filter sizes, a layer is used which incorporates parallel convolutions with different dilation rates (Figure 4).

The final layer of the network uses a pixelwise *sigmoid* activation to output values in the range of $[0, 1]$ for each pixel with 0 indicating that a pixel is not part of a driving tube and 1 indicating that a pixel is part of a driving tube. Therefore, the output values represent a probability of the pixel being

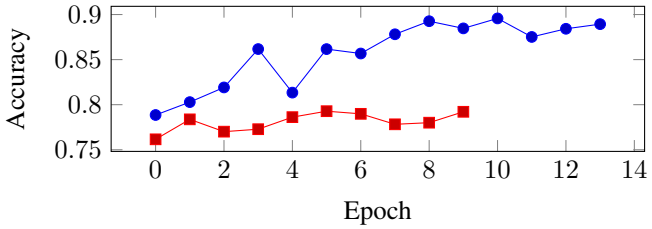


Fig. 5: Comparison between adapted U-Net using parallel dilated convolutional layers (blue) and no dilation (red) for small depth of 2 downsampling steps. Both models are trained until the accuracy converges.

part of a driving tube.

The cross-entropy loss function is selected as loss function to optimize the FCN model towards matching the distribution of the training data. Additionally a weighting factor between the two classes $\hat{y}_{c,r} = \{0, 1\}$ is introduced as most pixels in an output grid are not part of the road the $\hat{y}_{c,r} = 0$ class overweights. To counteract, a factor ω_1 is introduced to increase the weight on the $\hat{y}_{c,r} = 1$ class which results in the finally used loss function

$$\mathcal{L} = \sum_{c=0}^{C} \sum_{r=0}^{R} -(\omega_1 * (L_{c,r} * \log F_{\theta,c,r}(\mathbf{x})) + ((1 - L_{c,r}) * \log (1 - F_{\theta,c,r}(\mathbf{x})))) \tag{1}$$

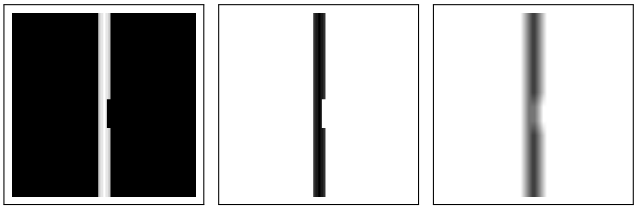
with $L_{c,r}$ being the label assigned in Section II-A-*Training Label Grid* and F being the output of the network.

C. Post-Processing for Navigation Use Case

For the navigation use case, the output of the FCN model requires further processing. To control the vehicle a trajectory has to be calculated. First, a path and its boundaries are calculated before a trajectory is optimized through that path. For this use case, we are conditioning the FCN model on low prior knowledge as it improves the certainty of the model on a single path.

1) Goal Selection: Independent of the model being conditioned and outputting only a single driving tube or multiple, in order to extract the paths, a goal on the grid has to be defined. Low prior knowledge in the form of offline planned desired routing is used similar to the model conditional. The desired route is projected onto the grid as single pixel width lines. The point where the desired route intersects with the border of the output grid is taken as the start point for a goal search. Around this point, with a search radius of the approximated uncertainty of the low knowledge data source, the point with the maximum probability of being part of a driving tube is found. This point is defined as the goal point.

2) Path Extraction and Smoothing: After a goal is determined a path from the position of the ego vehicle in the output grid to the goal is calculated. First, the output grid O (Section II-A-*Output Grid*) is converted into a cost grid $C = 1 - O$. As the vehicle has a spatial dimension larger than a single pixel the final cost grid for path extraction is created by a uniform convolution for each pixel on C with



(a) (b) (c)

Fig. 6: Generation of convoluted cost grid. (a) Artificialy created optimal output grid with narrow passage by obstacle. (b) Cost grid created by inverting (a). If a path would be planned, this path would be too close to the obstacle. (c) Convoluted cost grid. The convolution considers the surrounding of each pixel. Pixels close to the obstacle receiver higher cost.

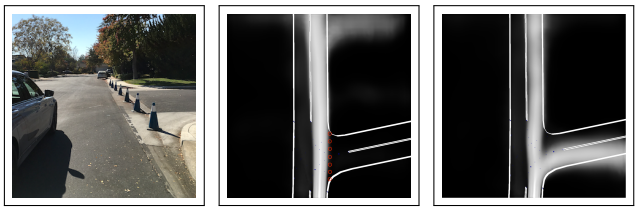
the convolution mask dimension equaling the length of the vehicle (Figure 6). Note that this is over-approximating as the vehicle is longer than it is wide. On this cost grid, a Dijkstra search is performed. If the total or the maximum cost of a single pixel along the found path is above a certain threshold a new goal must be selected (re-routing).

This path in grid coordinates is transformed into the ego vehicle coordinate system, in which the path is smoothed by fitting cubic splines to multiple waypoints while maximizing the closeness of the fit with the given points and also maximizing a smoothing criterion (the full fitting and smoothing algorithm is described in [17]). If alternatively a search algorithm which considers the holonomic constraints of the vehicle is used, smoothing would no longer be necessary. A similar approach used in [18] would be feasible where reachability maps taking holonomic constraints into account are generated offline and are then online used to plan a path through a cost grid.

3) Boundary Extraction: The smoothed path points are transformed back onto the output grid to calculate the boundaries of the driving tube for each point. The probability threshold for which a pixel in the output grid is still assumed to be part of the driving tube, is defined offline. From each interpolated point on the path, a binary search in two spatial directions orthogonal to the path is performed to find the pixel furthest away in the respective direction with a probability value still higher than the threshold. The output towards the trajectory planner is then the desired path consisting of waypoints, heading as well as boundaries left and right orthogonal towards the heading.

4) Trajectory Planning: A path-tracking controller using a Model Predictive Control (MPC) approach [19] is used to generate an optimized trajectory subject to vehicle dynamics and moving objects. In doing so, the controller optimizes a forecast of the system behavior. Generally, a defined weighted quadratic cost function is minimized with respect to all system constraints. The output from the FCN model can be incorporated into the MPC controller in two ways. Either the cost grid C from Section II-C.2 is directly used as a weighted minimization target for the tracking controller or the extracted boundaries from Section II-C.3 are added as additional constraints for the controller. An example

Fig. 7: Planned trajectory based on the output of the FCN path model. Part of the road is blocked by traffic cones (red circles). The trajectory is planned around them.



(a) (b) (c)

Fig. 8: FCN path model compared to high prior knowledge for blocked road using traffic cones. (a) Picture of the scene: Blue cones block the right turn. (b) Overlay of high prior knowledge (HD map) with output of driving tubes of the FCN path model. (Cones are manually inserted as red circles) (c) Same road but without cones. This time the FCN path model shows a right turn.

of a planned trajectory projected into the camera image is shown in Figure 7. By optimizing the trajectory with an MPC controller another problem of typical end-to-end systems which output direct actor commands is solved. Using MPC, additional safety and verification constraints can be incorporated into the optimization process [20].

D. Post-Processing for Knowledge Validation Use Case

Prior high accuracy knowledge can be validated to a certain extent using the output grid of the FCN path model (Figure 8). For this use case, the FCN model is not conditioned on any prior knowledge as the model shall generalize over all possible driving tubes for the vehicle. We are using common methods in machine learning to enforce generalization to multiple paths instead of overfitting to a single: Increase dropout, decrease model capacity, and early stopping. A simple way to validate the map is to compare the driving tubes outputted by the FCN path model with driving tubes inferred by an HD map. The absolute difference between the two grids gives a heat map of where the two models differ. Taking the average of these differences gives an indicator of the divergence of the two models. High deviations between prior knowledge and newly inferred knowledge from sensor data have to be evaluated to take appropriate actions which are out of the scope of this paper.

III. Results

Training the FCN model conditioned for the *Navigation Use Case* we reach a pixelwise softmax accuracy of above 90% for our specific scene-based data set. For the unconditioned FCN we found empirically that a well-generalizing model only has an accuracy of $79 \pm 5\%$ as the label does not include all possible paths. However, the authors acknowledge that these accuracies are hard to interpret in context as they are not calculated on a standardized data set. Before a trained FCN path model is integrated into the vehicle for real world closed loop driving, we applied intensive testing on three levels. First, a simple closed-loop simulation for a virtual vehicle is created. Then a re-simulation algorithm is developed and implemented to ensure the system is able to deal with real-world sensor noise. Finally, open loop analysis for the scenes is conducted before closed loop is engaged. For this proof of concept, two driving scenes are selected and two models are trained with data collected from these. The first scene is a neighborhood environment typical for the US. This includes sections with varying road width, mostly no lane markings, and parked vehicles as well as other obstacles on the street. This scene is selected as it is a representative unstructured environment for sensor based-navigation. The second scene for which a model is trained is a highway environment. Here specifically the situation of lane keeping is chosen in order to compare our approach to standard driver assistance systems. Both scenes were successfully navigated using the respectively trained FCN path model. Even in environments where no lane markings are present the navigation function is maintained as the model orientates itself on the other input features it receives. For our testing closed-loop was only engaged on proving grounds and most of the evaluation is being done within our simulation pipeline. While the results to this point are mostly qualitative, we work towards a quantitative comparison against other approaches on standardized test sets. However, we showed that the model is able to maintain a driving function relying on input features solely from redundant sources and no raw images.

IV. Conclusion

In this work, a new approach is being presented to generate an online path model from sensor data which can be conditioned on prior knowledge. We applied and further developed state of the art algorithms from semantic segmentation and path planning. The model can be trained in an end-to-end fashion using data from recorded traces without having the disadvantages of direct model to actor output or raw/single sensor dependencies. The benefit of this approach is shown by implementing two separate use cases. One of the use cases included building a functional autonomous driving architecture where all necessary steps from data collection over model implementation and training to (re-)simulation are performed before testing the system in closed loop in the real world. While the navigation use case was originally designed to take over navigation from global navigation for a short limited time period until high prior knowledge is available again, the system is able to drive multiple miles solely on the online system. The second use case showed the first steps towards the feasibility of such a model to detect inaccuracies in map data.

Further Work: For comparability and better analytic evaluation the FCN model should be adapted using standardized data sets such as KITTI [6]. Also, an empirical analysis of the contribution of each input towards the FCN performance would be interesting. As described, the FCN models are trained scene specific. As a further consideration, the diversity of scene data will be analyzed and used to train models which maintain their overall performance over multiple scenes. To better assess this performance, an analytical evaluation of the results and outputs against a ground truth should be performed. Such a ground truth can be produced from an HD map and high precision dGPS localization. While the map validation process was presented conceptually, the performance of this approach requires further evaluation.

Acknowledgment

The authors thank the BMW Group for the support of this work.

References

[1] M. Aeberhard, S. Rauch, *et al.*, "Experience, Results and Lessons Learned from Automated Driving on Germany's Highways," *IEEE Intelligent Transportation Systems Magazine*, 2015.

[2] J. Ziegler, P. Bender, *et al.*, "Making bertha drive-an autonomous journey on a historic route," *IEEE Intelligent Transportation Systems Magazine*, 2014.

[3] S. Bittel, V. Kaiser, *et al.*, "Pixel-wise Segmentation of Street with Neural Networks," 2015.

[4] V. Badrinarayanan, A. Handa, *et al.*, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Robust Semantic Pixel-Wise Labelling," 2015.

[5] L. Caltagirone, S. Scheidegger, *et al.*, "Fast LIDAR-based road detection using fully convolutional neural networks," *IEEE Intelligent Vehicles Symposium, Proceedings*, 2017.

[6] J. Fritsch, T. Kuehnl, *et al.*, "A New Performance Measure and Evaluation Benchmark for Road Detection Algorithms," in *International Conference on Intelligent Transportation Systems (ITSC)*, 2013.

[7] M. Bojarski, D. Del Testa, *et al.*, "End to End Learning for Self-Driving Cars," 2016.

[8] D. Barnes, W. Maddern, *et al.*, "Find Your Own Way : Weakly-Supervised Segmentation of Path Proposals for Urban Autonomy,"

[9] T. Luettel, M. Himmelsbach, *et al.*, "Autonomous Ground Vehicles - Concepts and a Path to the Future," *Proceedings of the IEEE*, 2012.

[10] S. Ulbrich, T. Menzel, *et al.*, "Defining and Substantiating the Terms Scene, Situation, and Scenario for Automated Driving," *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 2015.

[11] G. Tanzmeister, M. Friedl, *et al.*, "Road course estimation in unknown, structured environments," in *IEEE Intelligent Vehicles Symposium*, 2013.

[12] F. Ghallabi, F. Nashashibi, *et al.*, "LIDAR-Based Lane Marking Detection For Vehicle Positioning in an HD Map," 2018.

[13] C. Fernández, M. Gavilán, *et al.*, "Free space and speed humps detection using lidar and vision for urban autonomous navigation," in *Intelligent Vehicles Symposium (IV), 2012 IEEE*, IEEE, 2012.

[14] O. Ronneberger, P. Fischer, *et al.*, "U-Net: Convolutional Networks for Biomedical Image Segmentation," 2015.

[15] W. Shi, F. Jiang, *et al.*, "Single image super-resolution with dilated convolution based multi-scale information learning inception module," in *2017 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2017.

[16] C. Szegedy, W. Liu, *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.

[17] P. Dierckx, "Algorithms for smoothing data with periodic and parametric splines," *Computer Graphics and Image Processing*, 1982.

[18] D. Ferguson and M. Likhachev, "Efficiently Using Cost Maps For Planning Complex Maneuvers Efficiently Using Cost Maps For Planning Complex Maneuvers," in *International Conference on Robotics and Automation Workshop on Planning with Cost Maps*, 2008.

[19] B. Gutjahr, L. Groll, *et al.*, "Lateral Vehicle Trajectory Optimization Using Constrained Linear Time-Varying MPC," *IEEE Transactions on Intelligent Transportation Systems*, 2016.

[20] K. Leung, E. Schmerling, *et al.*, "On Infusing Reachability-Based Safety Assurance within Probabilistic Planning Frameworks for Human-Robot Vehicle Interactions," 2018.