DirectionNet: Road Main Direction Estimation for Autonomous Vehicles from LiDAR Point Clouds*

Wancheng Shen[1], Meiping Shi[1] and Tao Wu[1,†]

Abstract— **In recent years, "end-to-end" autonomous driving methods based on deep learning have received increasing attention. Most of these methods take raw sensor data as input and directly generate vehicle control commands. The autonomous driving system is so complex that overfitting is inevitable, which will seriously affect the robustness and safety of the entire system. Although deep learning has shown outstanding performance over traditional machine learning in many tasks including environment perception, using end-to-end technologies that lack explanatory in the autonomous driving system will reduce system's robustness and safety. Different from existing perception tasks, inspired by human's driving style, we extract the key points of global route planning's projection on roads, which is called the "road main direction". This "road main direction" is then used for the subsequent driving tasks, guiding the vehicle along the road. Instead of using image as input, which is very susceptible to weather and illumination, our model directly processes the LiDAR point clouds. The results show that our method can extract the main direction of the road in different scenarios to provide direction guidance for the unmanned vehicle without a prior high-definition map.**

I. INTRODUCTION

After decades of development, autonomous driving technology can be mainly divided into two categories, one is based on traditional methods and supplemented by deep learning, and the other is based on deep learning, which has received extensive attention from researchers in recent years. The second method, so called end-to-end method, predicting control commands directly from raw sensor data using neural networks [1]–[5].

Conventional pipeline of autonomous driving generally consists of modules such as mapping, localization, perception, route planning and motion planning. Each component of this pipeline is fully developed, and this pipeline is used by many self-driving companies, which has been proven to be stable and effective. However, this pipeline is highly dependent on precise localization information provided by high-definition (HD) maps, which are expensive to build and maintain. When the driving environment of the vehicle changes, but the prior HD map is not updated in time, the localization information will be deviated and affecting the subsequent driving tasks, even the safety of the entire system.

Different from traditional approaches, the end-to-end approach aims to avoid building the maps of the environment and the subsequent localization and planning, collapsing the problem with a single neural network [4]. This makes the autonomous driving system simple and efficient. Just feeding one image captured by the camera, the neural network will output the optimizing control commands, then the unmanned vehicle will move to the goal points without the map.

However, for the autonomous driving system, which is full of complexity and uncertainty, it is very difficult to use a single neural network to implement all these tasks perfectly. The mapping function that the neural network needs to learn will be very complex and requires huge training data to cover driving variations. The training process of the model will also become extremely difficult, and it will take a long time for the loss to converge. Inevitably, the model will overfit to a certain extent, which leads to low generalization, and the performance of the model will greately degrade if the vehicle enters a completely alien environment.

More importantly, using a single neural network to replace all the modules in the traditional pipeline will greatly reduce the security redundancy of the driving system, which is the critical factor in the automatic driving. In the traditional way, if the pre-order module, such as localization or perception deviates, the subsequent planning and control modules will try to make up for it to increase the security of the system. But in the end-to-end way, a small error can induce severe consequence. For example, when we reproduced the work called BADGR [6], we found that the model would erroneously consider shadows under tree canopies as impassable areas to avoid and drive off the normal road to rush towards the lawn, which would be dangerous if there were people on the lawn.

In summary, we believe that the role of deep learning is too small in traditional autonomous driving methods, but greatly exaggerated in the "end-to-end" methods. In this paper we try to find an intermediate way to transition from traditional methods to end-to-end methods to fully exploit the strengths of deep learning. For the tasks of obstacle avoidance, local planning and vehicle control, there are already well-established solutions based on traditional methods, deep learning should be used for problems that are difficult to model with traditional methods, such as perception tasks. Different from existing perception tasks, inspired by human's driving style, we extract the "main direction" of the road from sensor data. An overview of the proposed approach is shown in Fig. 1.

We analyze in-depth the process of human drivers driving to the destination under the guidance of a simple navigation map: driving along the road, avoiding obstacles, overtaking

*This work was not supported by any organization

[1]Wancheng Shen, Meiping Shi and Tao Wu are with the college of Intelligence Science and Technology, National University of Defense Technology, Changsha, China 14752276852@163.com; mp-shi@nudt.edu.cn; wutao@nudt.edu.cn

[†]Corresponding Author.

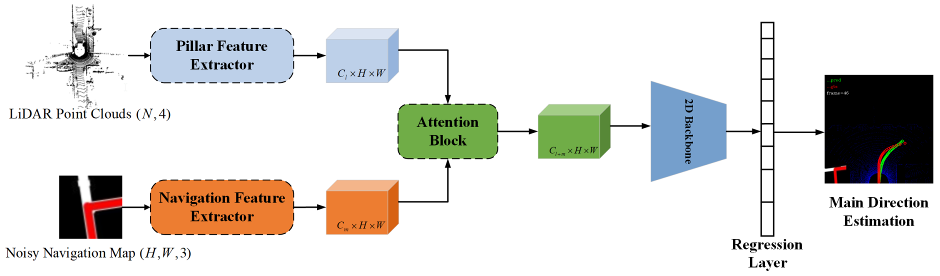

Pillar Feature Extractor
$C_l \times H \times W$
LiDAR Point Clouds $(N,4)$
Attention Block
$C_{l+m} \times H \times W$
2D Backbone
Navigation Feature Extractor
$C_m \times H \times W$
Noisy Navigation Map $(H,W,3)$
Regression Layer
...pred
...gts
frame=46
Main Direction Estimation

Fig. 1. **System overview:** DirectionNet takes LiDAR point clouds and local navigation map as input to estimate the main direction of the road. Using a Pillar Feature Extractor to extract the feature of the point clouds and then fused with the navigation map feature through the Attention Block. Finally a 2D backbone extracting high-level feature map followed by a full-connected layer to regress the length and y-coordinates representing the road main direction

or other driving actions, and then continue to drive along the road, so, the main direction of the road is a very important information for the driver. Instead of using image as input, which is very susceptible to weather and illumination, our model directly processes the LiDAR point clouds. The ”road main direction” is then used for the subsequent driving tasks, guiding the vehicle along the road.

The rest of this paper is organized as follows. In Section II, we introduce some related works. Section III provides an overview of the proposed approach. In Section IV, experimental results are presented. Finally, the conclusions are summarized in Section V.

II. RELETED WORK

The conventional pipeline of autonomous driving typically consists of modules such as mapping and localization [7]–[9], perception [10]–[14], route planning and motion planning [15]–[17], control [18], [19]. This map-based method of autonomous driving relies heavily on offline-created HD maps using either LiDAR and/or vision. But it is expensive to build and maintain these HD maps. In order to reduce the operating cost of the system and improve the efficiency, there have been many learning-based methods to address these limitations by learning from data. In this section, we give a brief review of several popular learning-based systems, each with a different system design.

A. *Reinforcement Learning*

Reinforcement learning is a kind of experiential learning, it allows the agent to continuously explore and develop the environment based the reward. Reinforcement learning is often used in visual navigation tasks, it learns a navigation policy by maximizing the reward function for a set of state-actions pairs.

Oh et al. [20] proposed a novel Value Prediction Network (VPN), which combines model-free and model-based RL methods into a single network, so it has some advantages in some scenes. Different from most model-based methods, VPN predicts future values rather than future observations.

Similar with this work, for the task of avoiding obstacles in the indoor environment, Khan et al. [21] proposed the Generalized Computation Graphs (GCGs) and deployed it on a real-word small car. Manderson et al. [22] expanded on this by adding the ability of predicting terrain roughness in off-road unstructured environments.

Even further, BADGR [6] took visual navigation as a self-supervised multi-task reinforcement learning problem, simultaneously predicting terrain roughness, whether a collision will occur , and whether the target-point will be reached.

B. *Imitation Learning*

Although reinforcement learning has shown great performance in visual navigation tasks, it has risks in the process of collecting training data. Reinforcement learning requires the agent to explore the environment and make various attempts, but some attempts, such as collision, is very dangerous for the data acquisition equipment. Unlike reinforcement learning, imitation learning learns from demonstration, generally the decision-making data from human experts. Training data is very easy and safe to collect for imitation learning.

As early as 1989, Pomerleau et al. [23] implemented behavioral cloning using a simple 3-layer forward network for the task of road following. Its training data are observation and action pairs from a human expert. However, behavioral cloning is not robust to environment changes. Some researchers try to solve this problem with data augmentation or domain matching methods [24]–[27], but none have solved the problem fundamentally.

Stphane Ross et al. proposed the DAgger (Dataset Aggregation) algorithm [28] to address this problem. This is an online learning-based algorithm that continuously interacts with the environment and generates new data based on behavioral cloning strategies. Then re-train the model with the augmented data iteratively.

C. *Direct Perception Approach*

Another learning-based method attempts to find an intermediate representation of the environment that can provide useful information for vehicle control, which is most similar to the method proposed in this paper.

This concept was first proposed by Chen et al. in DeepDriving [29], which predicts some useful affordances such as body posture, distance from the vehicle in front, from the roadside, etc. from the input image.These affordances are then used to calculate the specific driving commands such as acceleration, braking, steering, etc. Al-Qizwini et al. [30]

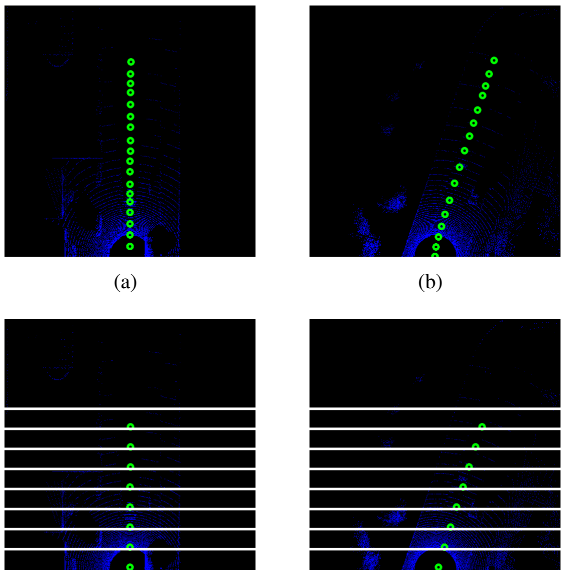


Fig. 2. Several examples of the road main direction. The top two pictures are of the time-sampling method, and the bottom two pictures are of the spatial-sampling method.

and Sauer et al. [31] improved the DeepDriving by analyzing various CNNs for the mapping from images to commands and adding the ability to navigate at the crossroads, stop at traffic lights. Apart from these works, traversable region recognition also belongs to this category, DeepGoal [32] used a weakly-supervised model named cGAN-LSTM to learn driving intention which is then fused with reliable obstacle perception for motion planning.

III. METHODS

The conventional pipeline of autonomous driving is rely on offline-created HD maps heavily which are expensive to build and maintain. In order to reduce the operating cost of the system, this paper proposes a method to guide the vehicle with low cost noisy navigation maps rather than HD maps. The core of our proposed method is the extraction of the road main direction, we first explain the meaning and representation of the road main direction in detail, then introduce the network architecture including the point clouds processing and the attention mechanism using to fuse LiDAR point clouds feature and local navigation map feature.

A. Road Main Direction Representation

No matter in the urban or in the off-road environment, the road has a "main direction". The main direction of the road refers to the trend of the road, it guides human drivers along the road.

It is an inherent attribute of the road and will not be affected by the dynamic vehicles driving on the road or other road participants. Several examples of the road main direction can be found in Fig. 2.

After projecting the results of global route planning to the road in the local LiDAR coordinate system, we use a series of key points $S=\{p_1,p_2,\cdots,p_k\}, p_i=(x_i,y_i)$, where x and y indicate "forward" and "left" respectively in the LiDAR coordinate system, to represent the road main direction, which can be divided into two methods, time-sampling and spatial-sampling.

As shown in the Fig. 2(a) and 2(b), we can project the historical trajectory points of the vehicle into the LiDAR coordinate system and then use the set of trajectory points to represent the road main direction.

This simple representation method is based on time-sampling. The time interval between every two points is equal. If the frequency of the odometer collecting the global coordinates is 50 Hz, the time interval between every two trajectory points is 20ms. The distance between every two trajectory points will be relatively close if the speed of the data-collecting vehicle is slow, otherwise will be relatively far. This leads to uneven data distribution, which is not conducive to model training.

Referring to the representation of lane line [33], another method to model road main direction is based on spatial-sampling, which is the method used in this article. Drawing several equally spaced horizontal lines under the bird's eye view of the point clouds, the road main direction is then represented by a set of intersections of the horizontal lines and the historical trajectory, as depicted in Fig. 2(c) and 2(d). This method can generate uniformly distributed data compared to time-sampling method. The number of points can represent the length l of the road main direction in appearance. Let the farthest observation distance be H meters, spacing between these horizontal lines be h meters, then the maximum length of the road main direction $M=\lfloor H/h \rfloor$. Because h is fixed, these intersections can be uniquely determined by their y-coordinates, namely$\{y_0,y_1,\cdots,y_{l-1}\}$.

For each horizontal line $h_s(x=x_s)$, finding the first trajectory point $p_i=(x_i,y_i)$ which x-coordinate is greater than x_s, then the index of the last trajectory point which x-coordinate is less than x_s is $i-1$.Then we use the linear interpolation to obtain the corresponding y-coordinate y_s of x_s through (1):

$$y_s=y_{i-1}-\frac{y_{i-1}-y_i}{x_{i-1}-x_i}(x_{i-1}-x_s), \tag{1}$$

Therefore, the road main direction can be represented as $D=\{l,y_0,y_1,\cdots,y_{l-1},\cdots,y_M\}$, if $l<M$, the rest y-coordinates is set to zeros.

B. Point Clouds Feature Extraction

Our DirectionNet takes raw point clouds and noisy local navigation maps as input and output estimated main direction of the road, network architecture is shown in Fig. 1.

There is no completely fixed paradigm for deep learning to process point clouds data. Several methods project point clouds into a perspective view to obtain BEV images or Range images [34], [35] and apply image-based feature extraction approaches. Some methods process point clouds directly [36]. Considering the problem to be solved in this paper, extracting features from the bird's eye view [12] is more in accordance with the physical meaning of the main direction of road.

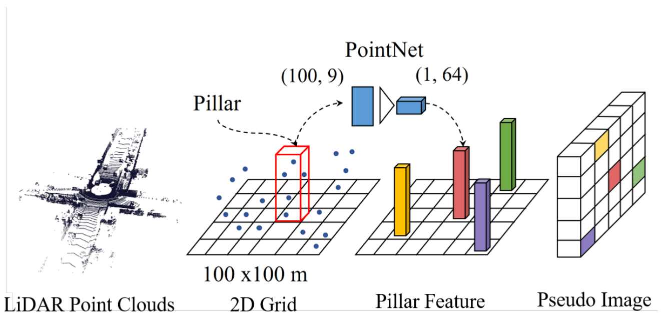


Fig. 3. **Pillar Feature Extractor:** We first discretize the point clouds into a 2D grid to generate a set of pillars, then use PointNet to extract pillar-wise features from these pillars.Finally, these pillar-wise features are scattered back to a 2D pseudo image.

DirectionNet takes two main steps to process the input LiDAR point clouds, just as shown in Fig. 3: (1) Rasterizing point clouds to 2D grids in Cartesian coordinates from the bird's eye view; (2) Converting the point clouds to the sparse pseudo-image through a Pillar feature net.

1) Point clouds Rasterization: We first rasterize the point clouds into an evenly spaced grid on the $x-y$ plane, this is the same as cutting a series of pillars from the original 3D point clouds. We denote a point in the point clouds by $q=(x,y,z,r)$, where x,y,z are 3D cartesian coordinates of q and r is the intensity. Same as in PointPillars [12] and GndNet [37], in addition to the 3D coordinates of the point, we apply feature enhancement on every point in each pillar by adding the offset relative to the centroid $c=(c_x,c_y,c_z)$ of all point clouds in the pillar: $x_c=(x-c_x), y_c=(y-c_y), z_c=(z-c_z)$, and the offset to the center of the pillar x_p,y_p. The point q is enhanced from $D=4$ dimension to $D=9$ dimension.

Due to the sparsity of the point clouds, the pillars will be mostly empty, and the non-empty pillars usually have only a few points in. The number of non-empty pillars in each point clouds is different, and the number of points of each non-empty pillar is also different. For the batch training, we set the number of all the non-empty pillars from raw point clouds to I and the number of points in each pillar to N, then we can get a fixed (D,I,N) size dense tensor after the rasterization. The data will be randomly sampled to the set size if the point clouds(pillars) contain too many pillars(points) or be padded by zero if the data is too little.

2) Pillar Feature Extraction and Pseudo Image: Next, we use a simplified version of PointNet which is the same as PointPillars [12] and GndNet [37] to extract the point-wise feature of each non-empty pillar. The simplified PointNet consisting of a linear layer, BatchNorm [38] and ReLU generates a (C_l,I,N) sized tensor, where the subscript l indicates LiDAR. Then, a (C_l,I) sized pillar-wise feature tensor is output through a max-pooling layer over the channels.

Last step, these pillar-wise features are projected to the original 2D grid map according to the grid coordinates of each pillar to obtain a (C_l,H,W) sized pseudo-image, where H and W indicate the height and width of the 2D grid map.

C. Attention Mechanism

Because the vehicle will only travel in a fixed direction on the road instead of going in different directions at the same time, the method proposed in this paper only predicts one "main direction" of the road. In most cases, this method is valid and accurate. However, in the cases of intersections, only using LiDAR point clouds as input, the predicted road main direction will be ambiguous because there is no navigational information about the direction of intersections. So we introduce the coarse precision navigation map to provide necessary navigational information to remove ambiguity at intersections.

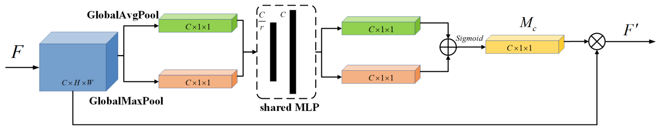
We use one residual block of ResNet18 as the navigation feature extractor to process the input navigation map. The navigation feature extractor generates a (C_m,H,W) sized feature map, where the subscript m indicates navigation map. Then we concatenate the pseudo-image of size (C_l,H,W) generated from LiDAR point clouds and the (C_m,H,W) sized navigation feature map to get the fusion feature map of size (C_f,H,W), where $f=l+m$.

Using this simple feature fusion method, the model may over-rely on the information of the navigation map and ignore the environmental information provided by the LiDAR point clouds. However, these noisy navigation maps are unreliable in many cases and the navigational information provided by these maps is not perfectly aligned with the main direction of the road. For the above considerations, we introduce the channel attention mechanism to learn the weights between navigation feature map and point clouds feature map, let neural network capture the correlations between navigation maps and LiDAR point clouds during fusion.

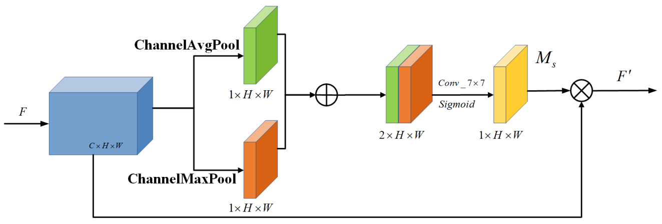
As shown in Fig. 4(a), we first use both global-max-pooling and global-average-pooling simultaneously on the concatenated feature map $\mathrm{F}\in\mathbb{R}^{C\times H\times W}$ to aggregate two different spatial feature vectors: $\mathrm{F}^{\mathrm{c}}_{\mathrm{avg}}$ and $\mathrm{F}^{\mathrm{c}}_{\mathrm{max}}$. Both feature vectors are then fed into a weights-shared MLP with one hidden layer. In order to capture the relationship between different channels, the number of units in the hidden layer is set to C/r, where r is the reduction rate. We then use element-wise addition to merge these two output feature vectors of the MLP. The channel attention map $\mathrm{M_c}\in\mathbb{R}^{C\times 1\times 1}$ is generated through a sigmoid activation function.This attention map represents the weights of different channels of the input feature map. Finally, using a channel-wise multiplication between $\mathrm{F_c}$ and $\mathrm{M_c}$ to get the channel attention weighted feature map $\mathrm{F}'\in\mathbb{R}^{C\times H\times W}$.

In addition to the interference of intersections, dynamic vehicles on the road will also affect the prediction results of the model. So we introduce the spatial attention mechanism to let the model focus on the areas of interest such as road curbs or repeated objects on both sides of the road.

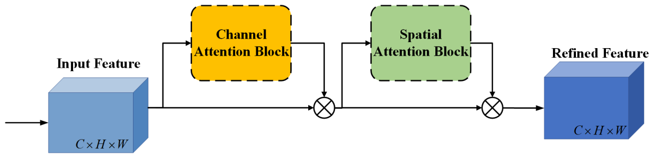
As shown in Fig. 4(b), for the input feature map $\mathrm{F}\in\mathbb{R}^{C\times H\times W}$, we get two $(1,H,W)$ sized tensor from channel-average-pooling and channel-max-pooling and then concatenate them along the channel axis to highlight focus regions. After pooling, we apply a 2D convolution layer with sigmoid activation function on the $(2,H,W)$ sized concatenated tensor to generate the spatial attention map $\mathrm{M_s}\in\mathbb{R}^{1\times H\times W}$. Finally, using multiplication between $\mathrm{F_{in}}$ and $\mathrm{M_s}$ to get the spatial



(a) Channel Attention Block



(b) Spatial Attention Block



(c) Connect CAB and SAB in series

Fig. 4. **The overview of the attention mechanism. (a):** The channel attention uses both global-average-pooling and global-max-pooling following with a shared MLP to merge information between channels; **(b):** The spatial attention utilizes both channel-average-pooling and channel-max-pooling following with a convolution layer to obtain spatial information; **(c):** Finally, connect channel attention block and spatial attention block in series.

attention weighted feature map $\mathrm{F}' \in \mathbb{R}^{C\times H\times W}$.

Similar with CBAM [39], we connect channel attention block and spatial attention block in series (as depicted in Fig. 4(c)) to focus on 'which' is imporant between navigation maps and LiDAR point clouds, 'where' is important on the road.

Once getting the refined fused feature through the attention block, we can use a 2D backbone such as ResNet [40] to extract the high-level feature map and a full-connected layer to regress the length and y-coordinates representing the road main direction.

D. Loss Function

We train the DirectionNet using an L_2 loss between the predicted road main direction D_t^p and the ground truth road main direction D_t^{gt} at the t-th frame, then the loss function is given by:

$$\mathscr{L} = ||D_t^p - D_t^{gt}||_2, \tag{2}$$

Note that if the ground truth length of the road main direction l_t^{gt} is less than the maximum length M of the road main direction, the rest loss is set to zeros.

IV. EXPERIMENTS

A. Data Collection

It's very easy to obtain the training data for our DirectionNet and don't need manual data annotation. Only a vehicle with LiDAR is required, odometry information can be provided by wheel odometer, Lidar SLAM technology or GPS. In this part, we use the CARLA [41] simulator to generate the dataset. We use Town04 to Town06 as training and hold out Town03 for validation, all of these towns are US style. We generate local navigation map by drawing all routes white on black background and drawing red on the traversed roads to simulate the results of global planning.

B. Data Processing and Augmentation

For each point cloud, the points within 2m from the LiDAR are removed due to the occlusion of data-collecting vehicle. We apply a number of data augmentations during training to expose the model to novel cases that are not collected and enhance the generalization performance of the model.

Firstly, we randomly rotate the point clouds and the road main direction points by a random yaw angle $\theta \in \pm 10^\circ$ to simulate the situation of the front of the car facing different directions on the road. This enables the model to pull the vehicle back to the main direction of the road. Then randomly horizontal flipping the point clouds and the corresponding road main direction points to simulate different turning directions out of the intersection. Besides, we add Gaussian noise on the navigation map to enhance the robustness of the system.

C. Implementation Details

We rasterize the point cloud into a $(50m \times 50m)$ sized 2D grid map, the cell resolution is $(0.5m \times 0.5m)$, so $H = W = 100$. The x, y, z axes are in the range of $[(0m, 50m), (-25m, 25m), (-4m, 4m)]$ and the points outside the range will be removed. The maximum number of points per pillar (N) is 100 and the maximum number of pillars per point cloud (I) is 10000.

The farthest observation distance H is set to 20 meters, spacing between these horizontal lines h is set to 0.5 meter.

The model is implemented in TensorFlow [42] and the training data is converted into TFRecord to accelerate the training.

D. Evaluation Metrics

For algorithm evaluation, we design two evaluation metrics, stability and accuracy. Stability is used to evaluate the reliability of the method, which is critical in autonomous driving systems. Only if there are enough prediction points, the prediction result can reflect the direction of the road. One prediction is considered valid if the prediction length $l > \delta_l$. Accuracy is used to evaluate how well the valid predicted direction aligns with the true direction. The definitions of these measures are as follows:

$$Stability = \frac{N_{valid}}{N_{total}}, \quad Accuracy_i = \frac{C_i}{S_i}, \tag{3}$$

where N_{valid} is the number of the valid predictions, N_{total} is the number of all predictions, C_i indicates the number of correctly predicted trajectory points in the prediction result of the i-th frame, and S_i represents the minimum of the predicted length and the ground-truth length.

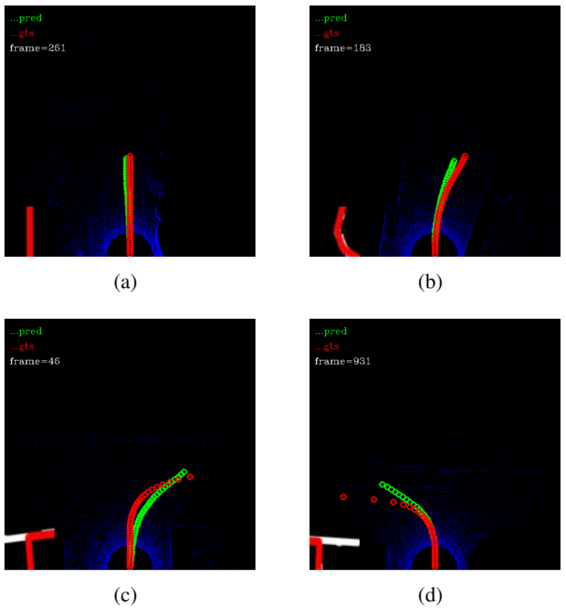


Fig. 5. Several results on the CARLA. The top two pictures are at straight roads, and the bottom two pictures are at intersections.Green points indicate the predicted road main direction and red points indicate the ground truth.

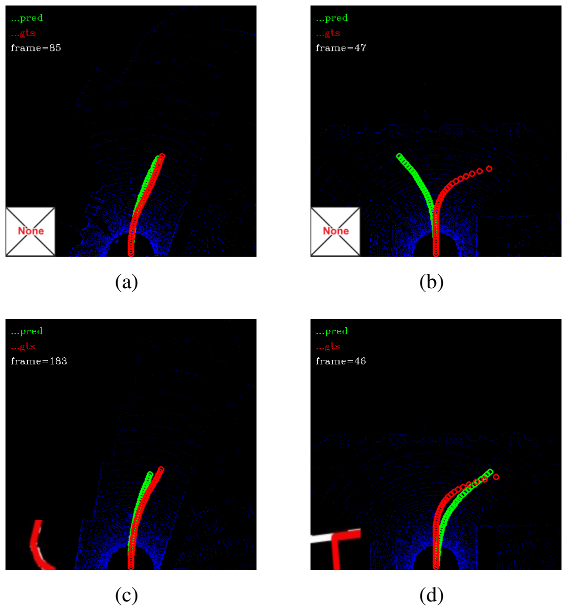


Fig. 6. The results of controlled experiments w/ and w/o navigation maps. The top two pictures are w/o navigation maps, and the bottom are w/ navigation maps.

If the distance between the ground-truth y-coordinate offset and the predicted offset is less than a given threshold (e.g. 0.5 meter), the predicted offset is assumed to be correct.

E. Results

1) Performance on CARLA: Our DirectionNet can get an average accuracy of 86.6% and 100% stability on the validation set. Note that, we use the autopilot script provided by CARLA to collect data for training and testing, this script will control the vehicle to do some actions such as obstacles avoiding and lanes chaning. Because of this, the collected historical trajectory points will not perfectly align with the real main direction of the road, so the average accuracy on the validation set will be lower than the actual result.

Real-time is very important for the autonomous driving systems. Our model can process each frame of data within 60ms, which can fully meet the real-time requirement. The specific performance is shown in Fig. 5. From which we can see that no matter driving on the stright/curve road or at the intersections, our DirectionNet can predict the main direction of the road correctly and stably. With the LiDAR point clouds and local navigation map, the predicted road main direction can guide the vehicle drive to the the destination along the road.

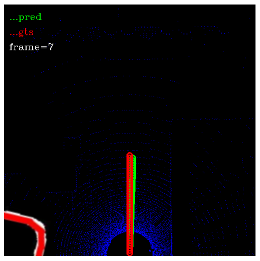
2) Ablation Experiments: To verify the influence of the navigation map and attention mechanism on our proposed model, we performed several ablation experiments. Firstly, we make two groups of controlled experiments to figure out what role the navigation map plays in our model.

In one controlled experiment, we set the input navigation maps to zeros to observe the prediction results only using the LiDAR point clouds, the result is shown in Fig. 6. Without the navigation mapsour DirectionNet can still correctly predict the road main direction on the stright/curve road. However, just as mentioned in Section III-C, the predicted road main direction will be ambiguous at intersections since there is no information of which direction to go. As expected, the navigation maps provide the model with necessary navigational information to remove ambiguity at intersections.

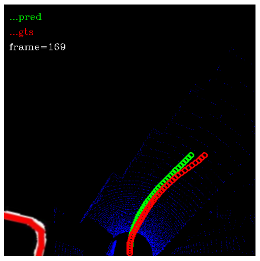
In another controlled experiment, we fix the input navigation map as "turning-left" to check if the navigation maps are over trusted. Just as shown in Fig. 7(a) and 7(b), when driving on the stright road, although the navigation map gives a wrong turning-left signal, the predicted road main direction is still stright. But at the intersection, in Fig. 7(c) and 7(d), the prediction direction will inevitably turn left. From this, we can conclude that our DirectionNet does not rely too much on the navigation map, but will combine the environmental information provided in the LiDAR point cloud and the navigational information to give the final prediction.

Secondly, we make one group of controlled experiments to figure out how the spatial attention mechanism affects model performance, the result is shown in Fig. 8. We can clearly see that the predicted direction will be severely affected by the obstacles on the road without spatial attention mechanism and this phenomenon is effectively improved after adding the spatial attention mechanism.

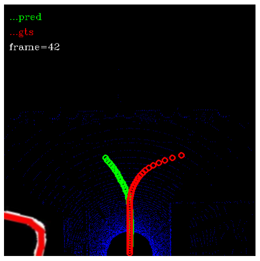
3) Generalization Experiments: Although the method proposed in this paper has shown good performance in the simulation environment, it is a big challenge to deploy the method on real-word vehicles with a variety of installed LiDARs. To test the generalization performance of the model, we first test the model trained with CARLA simulated data on the KITTI [43], [44] odometry dataset which is collected in real-world. The average accuracy dropped from 86.6% to 83.3%. Because the LiDAR used in both of CARLA



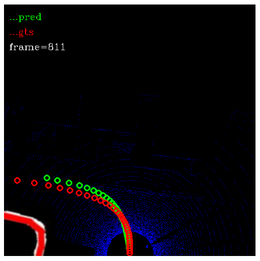
(a)



(b)

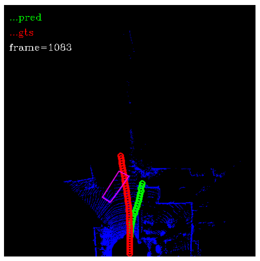


(c)

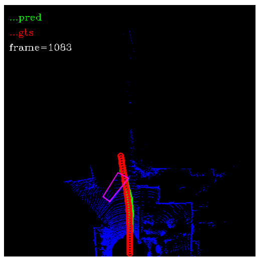


(d)

Fig. 7. The results of controlled experiments of fixed "turning-left" input navigation map. The top two pictures are at straight roads, and the bottom two pictures are at intersections.



(a) W/O Spatial Attention



(b) W/ Spatial Attention

Fig. 8. The results of controlled experiments w/ and w/o spatial attention mechanism.In the pink box is the obstacle in front of the road.

and KITTI is 64-channel mechanical LiDAR, the drop of performance is not noticeable.

But when we test the model on a 128-channel mechanical LiDAR, the performance has dropped so dramatically that the predicted road direction no longer reflect the direction of the road at all. The results can be found in Fig. 9. Therefore, enhancing the generalization ability of our proposed method on LiDARs with different scanning patterns will be the focus of the future study.

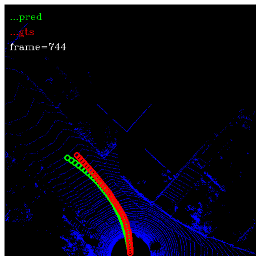
There is a growing trend for the autonomous driving companies to use more and more cheap solid-state LiDARs instead of expensive traditional mechanical scanning LiDARs. Therefore, we also test the generalization performance of the model on the solid-state LiDAR, the results can be seen in Fig. 10. The performance also dropped dramatically.

In conclusion, our model can still work normally on the LiDARs with similar scanning patterns, but cannot work with LiDARs with different scanning patterns.

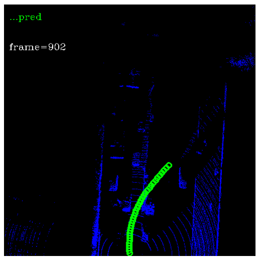
V. CONCLUSIONS

In this paper, we find an intermediate way to transit from traditional autonomous driving methods to end-to-end autonomous driving methods to fully exploit the strengths of deep learning. Inspired by human's driving style, the proposed DirectionNet can extract the road main direction of the road directly from the LiDAR point clouds, which can be used with subsequent motion planning and control to guide the vehicle drive along the road.

Up to now, the generalization ability of our method is not suitable for LiDARs with different scanning patterns on the market. Though collecting more training data from different LiDAR and training a new model from scratch could alleviate this problem, it unfortunately might be infeasible due to various LiDARs and real-world scenarios. Thus, our future work will tend to utilize domain adaptation technology [45] to enhance the generalization ability of our method under different LiDARs and different geographic environments.

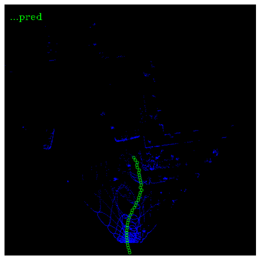


(a) KITTI result

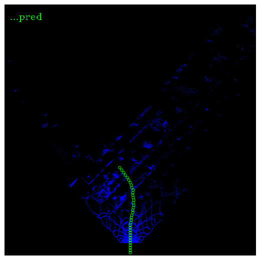


(b) 128-channel LiDAR

Fig. 9. The results of generalization performance test (We don't have ground truth of the 128-channel LiDAR).



(a)



(b)

Fig. 10. The results of generalization performance test on the solid-state LiDAR.

REFERENCES

[1] F. Codevilla, M. Mller, A. Lpez, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 4693–4700.

[2] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 3530–3538.

[3] S. Hecker, D. Dai, and L. V. Gool, "End-to-end learning of driving models with surround-view cameras and route planners," *Springer, Cham*, 2018.

[4] A. Amini, I. Gilitschenski, J. Phillips, J. Moseyko, and D. Rus, "Learning robust control policies for end-to-end autonomous driving from data-driven simulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1–1, 2020.

[5] A. Amini, G. Rosman, S. Karaman, and D. Rus, "Variational end-to-end navigation and localization," *arXiv*, 2019.

[6] G. Kahn, P. Abbeel, and S. Levine, "Badgr: An autonomous self-supervised learning-based navigation system," *IEEE Robotics and Automation Letters*, vol. PP, no. 99, pp. 1–1, 2021.

[7] A. Stentz, D. Fox, M. Montemerlo, and M. Montemerlo, “Fastslam: A factored solution to the simultaneous localization and mapping problem with unknown data association,” in *In Proceedings of the AAAI National Conference on Artificial Intelligence*. AAAI, 2003, pp. 593–598.

[8] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, “Monoslam: real-time single camera slam,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[9] J. J. Leonard and H. F. Durrant-Whyte, “Simultaneous map building and localization for an autonomous mobile robot,” in *Intelligent Robots and Systems ’91. ’Intelligence for Mechanical Systems, Proceedings IROS ’91. IEEE/RSJ International Workshop on*, 1991.

[10] Q. Luo, H. Ma, L. Tang, Y. Wang, and R. Xiong, “3d-ssd: Learning hierarchical features from rgb-d images for amodal 3d object detection - sciencedirect,” *Neurocomputing*, vol. 378, pp. 364–374, 2020.

[11] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2018.

[12] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, “Pointpillars: Fast encoders for object detection from point clouds,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 12 689–12 697.

[13] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” *Springer, Cham*, 2016.

[14] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” *arXiv e-prints*, 2018.

[15] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. J. Teller, “Anytime motion planning using the rrt*,” in *2011 IEEE International Conference on Robotics and Automation*, 2011.

[16] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[17] S. M. LaValle and J. J. Kuffner, “Rapidly-exploring random trees: Progress and prospects,” 2000.

[18] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat, “Predictive active steering control for autonomous vehicle systems,” *IEEE Transactions on Control Systems Technology*, vol. 15, no. 3, pp. 566–580, 2007.

[19] W. Schwarting, J. Alonso-Mora, L. Paull, S. Karaman, and D. Rus, “Safe nonlinear trajectory generation for parallel autonomy with a dynamic vehicle model,” *IEEE Transactions on Intelligent Transportation Systems*, pp. 2994–3008, 2017.

[20] J. Oh, S. Singh, and H. Lee, “Value prediction network,” in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/ffbd6cbb019a1413183c8d08f2929307-Paper.pdf

[21] G. Kahn, A. Villaflor, B. Ding, P. Abbeel, and S. Levine, “Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 5129–5136.

[22] T. Manderson, S. Wapnick, D. Meger, and G. Dudek, “Learning to drive off road on smooth terrain in unstructured environments using an on-board camera and sparse aerial images,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 1263–1269.

[23] D. Pomerleau, “Alvinn : An autonomous land vehicle in a neural network,” *Advances in Neural Information Processing Systems*, vol. 1, 1989.

[24] M. Laskey, J. Lee, R. Fox, A. Dragan, and K. Goldberg, “Dart: Noise injection for robust imitation learning,” in *1s Conference on Robot Learning*, 2017.

[25] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, and J. Zhang, “End to end learning for self-driving cars,” 2016.

[26] J. Ho and S. Ermon, “Generative adversarial imitation learning,” in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016. [Online]. Available: https://proceedings.neurips.cc/paper/2016/file/cc7e2b878868cbae992d1fb743995d8f-Paper.pdf

[27] A. Giusti, J. Guzzi, D. Ciresan, F. L. He, J. P. Rodriguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, and G. D. Caro, “A machine learning approach to visual perception of forest trails for mobile robots,” *IEEE Robotics and Automation Letters*, pp. 1–1, 2017.

[28] S. Ross, G. J. Gordon, and J. A. Bagnell, “A reduction of imitation learning and structured prediction to no-regret online learning,” *Aistats*, vol. abs/1011.0686, 2011.

[29] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, “Deepdriving: Learning affordance for direct perception in autonomous driving,” *IEEE*, pp. 2722–2730, 2015.

[30] M. Al-Qizwini, I. Barjasteh, H. Al-Qassab, and H. Radha, “Deep learning algorithm for autonomous driving using googlenet,” in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017.

[31] A. Sauer, N. Savinov, and A. Geiger, “Conditional affordance learning for driving in urban environments,” in *Proceedings of The 2nd Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Billard, A. Dragan, J. Peters, and J. Morimoto, Eds., vol. 87. PMLR, 29–31 Oct 2018, pp. 237–252. [Online]. Available: https://proceedings.mlr.press/v87/sauer18a.html

[32] A. Hm, W. A. Yue, X. A. Rong, B. Sk, and T. A. Li, “Deepgoal: Learning to drive with driving intention from human control demonstration - sciencedirect,” *Robotics and Autonomous Systems*, vol. 127.

[33] X. Li, J. Li, X. Hu, and J. Yang, “Line-cnn: End-to-end traffic line detection with line proposal unit,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 1, pp. 248–258, 2020.

[34] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, “Multi-view 3d object detection network for autonomous driving,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[35] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, “Rangenet ++: Fast and accurate lidar semantic segmentation,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[36] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, “Pointnet: Deep learning on point sets for 3d classification and segmentation,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 77–85.

[37] A. Paigwar, . Erkent, D. Sierra-Gonzalez, and C. Laugier, “Gndnet: Fast ground plane estimation and point cloud segmentation for autonomous vehicles,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 2150–2156.

[38] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML’15. JMLR.org, 2015, p. 448456.

[39] S. Woo, J. Park, J. Y. Lee, and I. S. Kweon, “Cbam: Convolutional block attention module,” *Springer, Cham*, 2018.

[40] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[41] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “CARLA: An open urban driving simulator,” in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[42] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[43] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite,” in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3354–3361.

[44] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The KITTI dataset,” *International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231 – 1237, Sept. 2013.

[45] L. T. Triess, M. Dreissig, C. B. Rist, and J. Marius Zllner, “A survey on deep domain adaptation for lidar perception,” in *2021 IEEE Intelligent Vehicles Symposium Workshops (IV Workshops)*, 2021, pp. 350–357.