Conditional Imitation Learning Driving Considering Camera and LiDAR Fusion

Hesham M. Eraqi[⋆†] **Mohamed N. Moustafa**[⋆] **Jens Honer**[†]
[⋆]Computer Science and Engineering department, The American University in Cairo, Egypt
[†]Driving Assistance department, Valeo Schalter und Sensoren GmbH, Germany
heraqi@aucegypt.edu, m.moustafa@aucegypt.edu, jens.honer@valeo.com

Abstract

Autonomous driving is a complex task that is difficult to cast into algorithms. Therefore, researchers turn to deep neural networks that map front-facing camera data stream to the associated driving commands. The learned driving policy can be conditioned to respond to navigational commands, thus the vehicle can take specific turns in intersections to reach a destination. Such visual input-based technique is demonstrated to drive efficiently when deployed on the same training environments. Nevertheless, performance dramatically decreases in new environments and is not consistent against varying weather conditions. In this work, a proposed model copes with such two challenges by fusing laser scanner input with the camera. On CARLA urban driving benchmark, our model improves autonomous driving success rate and average distance traveled towards destination on all driving tasks and environments combinations, while it's trained on automatically recorded traces. Generalization success rate improves by 52% and weather consistency improved by around four times.

1 Introduction

Road traffic crashes is a major world unsolved problem. The 2018 Global Status Report of the World Health Organization (WHO) reported an estimated 1.35 million yearly deaths due to road traffic crashes worldwide [27]. In addition, 10 million people sustain non-fatal injuries or become disabled as a result of road crashes yearly [27]. In addition, roadway crashes cause a huge property damage, and have substantial negative economic and social effects [23]. With approximately 90% of accidents being due to human errors [26], autonomous vehicles will play a vital role to save human lives and property damage. In addition, it promises far greater mobility for the elderly and people with disabilities. Energy consumption could be reduced in transportation by as much as 90% [8] with less traffic congestion and associated air pollution [3].

Despite the recent advances to achieve such promising vision, it is safe to believe that fully autonomous navigation in complex environments is still decades away [15], the problem is far from solved [13]. CARLA [7] is a widely used open-source simulator for autonomous car development focused on creating realistic virtual environment for automotive industry. Many contributors constantly improve it, which makes it a comprehensive tool for simulating real world scenarios. In [7], CARLA urban driving benchmark is introduced to benchmark the state-of-the-art Conditional Imitation Learning (CIL) approach [6] with modular pipeline and Reinforcement Learning-based approaches. The benchmark demonstrated that the CIL model is responsive to high-level navigational commands and drives efficiently when tested on the same training environments. However, performance rabidly decreases and does not generalize in new towns and, additionally, is not consistent against varying weather conditions. To cope with such two challenges, in this work, our proposed model extends the CIL model by fusing laser scanner input with the camera. On CARLA benchmark,

Machine Learning for Autonomous Driving Workshop at the 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

our model improves driving success rate in new towns by 52% and weather consistency by 3.9 times. It performs significantly better than the CIL model [6] in all the different combinations of tasks and environmental setups, though being trained on driving traces recorded automatically, unlike the CIL model that is trained on data collected by a human driver.

2 Related Work

The majority of literature approaches to autonomous driving are composed of multiple different models [15] [7], e.g., detection [12] (drivable free space [10], lanes' markings [18], or pedestrians [21]), tracking of driving scene objects [4], motion planning [14], collision avoidance [9], mapping [28], and more models [15]. Then the results from these components are then combined in a rule-based module that produces the vehicle driving actions [7]. Such mediated perception approach relies on scene understanding [12] on a level that might add redundant information and useless complexity to an already difficult task of autonomous driving; a small portion of the detected objects are relevant to driving decisions. It also requires robust solutions to open challenges in scene understanding and expensive data annotation [12]. Direct perception is another approach [5] that learns a mapping from input camera image to several meaningful affordance indicators of driving situation, then a rule-based controller translates them into driving actions. The indicators are chosen via heuristics and the controller design is as expensive as the case with the mediated perception rule-based module [29].

As an alternative to the mediated perception approach, the end-to-end approach directly maps input sensory data to driving actions via deep machine learning regression. It aims to optimize all of the aforementioned sub-problems simultaneously and eventually lead to better performance and smaller systems. The first end-to-end work was done by Pomerleau [24] in 1989 that used a multilayer perceptron (MLP). Afterwards, the computational power dramatically increased thanks to massive parallelization in modern GPU's in combination with modern deep artificial neural network concepts like Convolutional Neural Networks (CNN) [20]. Those advances allowed for having more successful methods that use CNN for end-to-end learning of autonomous vehicles steering [1] [11] [15] [2]. The major drawback in this approach is that the vehicle cannot be guided to take a specific turn at an upcoming intersection. Conditional Imitation Learning (CIL) [7] [6] overcomes such limitation by training, on top of a perception CNN, multiple different command-conditional modules ("branches") predicting driving commands for each possible navigational command.

3 Proposed Model

3.1 Network Architecture

Our model extends the CIL model [6] by fusing a LiDAR (laser scanner, acronym of Light Detection And Ranging) sensor input with the camera. The strengths aspects for each sensor can compensate for the weaknesses of the other. The accurate LiDAR range information resolve the camera depth perception ambiguity, while camera's dense angular resolution compensates LiDAR sparsity. Also, LiDAR is less sensitive to ambient lighting and weather conditions [16]. Our proposed network is end-to-end trainable, given input sensory data, the vehicle driving commands are predicted, in addition to predicted vehicle speed. The predicted speed is used to avoid vehicle unnecessary stopping, we deliberately increase the predicted throttle if the vehicle is stopping while the predicted speed value is above a pre-determined threshold.

Figure 1 shows our proposed network architecture. The network takes the high-level navigational command C as an input, alongside image coming from a front-facing camera, LiDAR pointcloud, and a measurements vector. All inputs are processed independently. The currently observed RGB image coming from the camera is fed into 8 convolutional layers, and the associated LiDAR full scan pointcloud is encoded to a grayscale image using Polar Grid View (PGV) representation as descried later in subsection 3.2 then fed into 4 convolutional layers. The RGB image size is 200×88 pixels resolution, and the PGV grayscale image is 90×32 pixels resolution, where the second dimension represents the utilized LiDAR number of layers. We use a ReLU activation function [22] in all hidden layers, and a linear activation for the output layers. Figure 1 describes the number of neurons per layer, and for the convolutional layers describes kernel sizes and the used padding as well. Batch normalization is applied after all the convolutional layers, and we apply 50% dropout [25] after fully-connected hidden layers, and apply 20% dropout after convolutional layers.

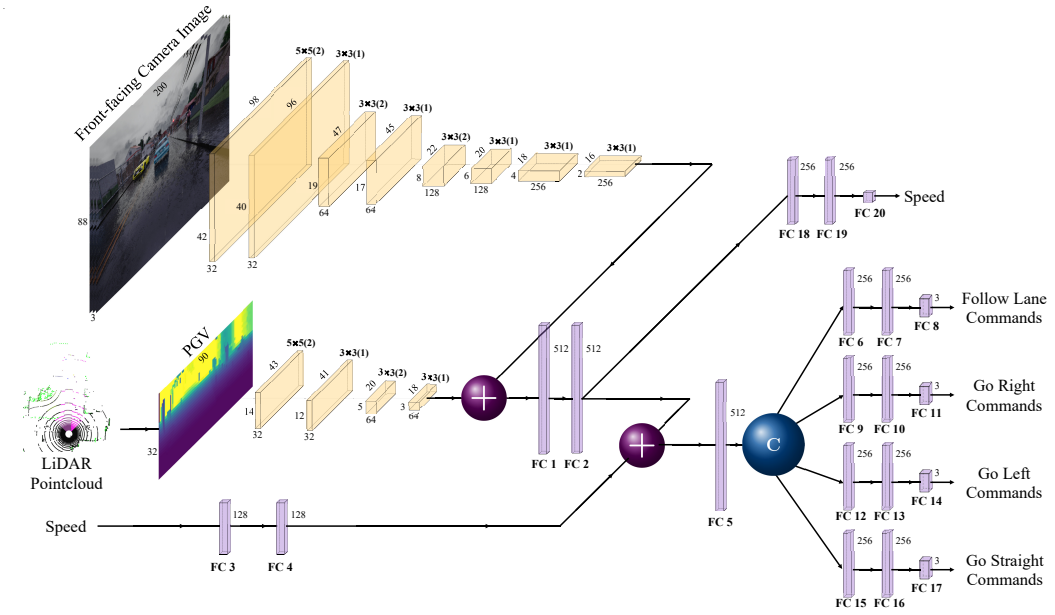

Figure 1: Proposed Network Architecture

The measurement vector is composed of vehicle speed. The speed, throttle, and brake values are scaled between 0 and 1, according to minimum and maximum possible values. The steering wheel angle is scaled between -1 and 1, with extreme values corresponding to full left and full right. The camera RGB images and the LiDAR PGV images are normalized to be in the range of $[0, 1]$. For each output branch corresponding to a navigational high-level command, driving actions a are three-dimensional vectors that include steering wheel angle s, throttle t, and braking b; $a = [s, t, b]$. Given ground-truth actions a_g and speeds v_g, and predicted actions a and speeds v, the loss function L is defined as follows:

$$\begin{aligned} L = \lambda_s \left\| s - s_g \right\|^2 + \lambda_t \left\| t - t_g \right\|^2 \\ + \lambda_b \left\| b - b_g \right\|^2 + \lambda_v \left\| v - v_g \right\|^2, \end{aligned} \tag{1}$$

where λ_s, λ_t, λ_b, and λ_v are empirically set to 0.5, 0.2, 0.15, and 0.15 respectively. The model is trained using Adam optimizer [19] with $\beta_1 = 0.7$, $\beta_2 = 0.85$, and initial learning rate of 0.0002, and it's multiplied by 0.5 every 10 epochs. We used mini-batches of 120 samples, where each min-batch has the same number of samples for each high-level navigational command C. Half of the images in every mini-batch are augmented as described later in subsection 3.3.

3.2 LiDAR Polar Grid View

As in figure 1, the currently observed LiDAR pointcloud is encoded to a grayscale image using Polar Grid View representation (PGV). Figure 2 shows a sample camera RGB image, the corresponding LiDAR pointcloud full scan top view projection, and the generated PGV which provides a 2D dense proximity spherical representation of the environment. Each LiDAR layer is associated with a PGV row, and each beam is associated with a single PGV column based on its horizontal angle. A PGV pixel holds the average depth values for all LiDAR beams that are associated with it.

We provide a linear-time algorithm for PGV generation in Algorithm 1. After each of steps 2 and 3 in the algorithm, a small constant angle can be added to make θ_{unique} and ϕ_{unique} represent segments centers instead of edges.

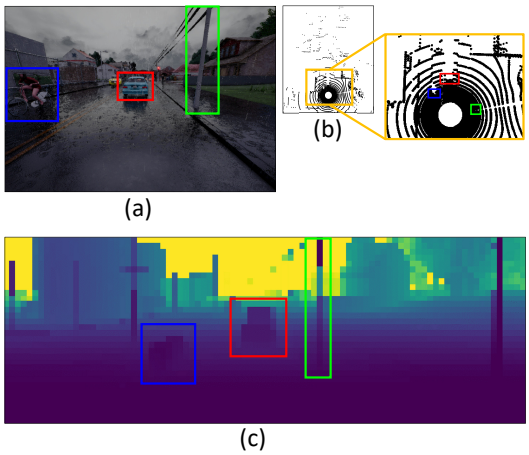

Figure 2: (a) Sample RGB camera image. (b) Corresponding LiDAR pointcloud top view projection. (c) Generated PGV from the LiDAR pointcloud. Three objects are matched in the figures: a vehicle, a bicyclist, and a light pole.

Algorithm 1: LiDAR Polar Grid View (PGV) linear-time algorithm

Input: Lists x , y, and z: LiDAR full scan Cartesian pointcloud in sensor local coordinate system
Parameters: N: LiDAR number of layers
u_val: value for unreflected LiDAR beams
FoV_U and FoV_L: LiDAR vertical field of view
θ_{res}: horizontal resolution of the produced PGV
Output: V: Grayscale image representing the PGV

$(\rho, \theta, \phi) =$ **cart2polar**(x, y, z) // Convert from Cartesian to Polar Coordinates
// Allocate empty map
allocate a list θ_{unique} in [0°, 180°) with step θ_{res} // [0°, 180°) for front-facing pointcloud
allocate a list ϕ_{unique} in $[FoV_U, FoV_L)$ with step $(FoV_L - FoV_U)/N$
allocate matrix V of values u_val and size **length**$(\phi_{unique}) \times$ **length**(θ_{unique})
foreach $V^{i,j} \in V$ **do**
    $target_\phi = |\phi - \phi_{unique}[i]| \leq (FoV_U - FoV_L)/2N$
    $target_\theta = |\theta - \theta_{unique}[j]| \leq \theta_{step}/2$
    $values_\rho = \rho[target_\phi$ **&** $target_\theta]$
    if ***length***$(values_\rho) > 0$ **then** $V^{i,j} =$ **mean**$(values_\rho)$

3.3 Training Dataset

The original CIL model is trained on a dataset collected by a human driver using CARLA simulator. The driver uses a signal to record his intent when approaching intersections [6], such a signal is used as the ground-truth navigational high-level command. In contrast, our model is trained on data that is automatically recorded using two different methods. The first data collection method relies on CARLA simulator autopilot feature. In each data collection episode, the episode time duration, weather set, traffic and pedestrians density, vehicle starting position are randomly chosen. The ego-vehicle purposelessly follows lane and takes random turning decisions in intersections and avoid obstacles until the episode duration ends. After each episode, the navigational high-level command is generated by looking-ahead in future frames to determine the turn the vehicle decided to take as described in Algorithm 2. In the algorithm description, x and y are lists of ego-vehicle x and y world coordinates ordered by timestamp, and $intersections$ is a list of town intersections. The threshold t is the minimum absolute change in vehicle x and y coordinates when taking right or left turns in an intersection, we empirically set it to 15 meters. The $inside$ function returns $True$ only if the passed x and y world coordinates to the function lie within the passed intersection s; i.e.; within a circle having a predetermined radius value (empirically set to 30 meters) from the intersection center point. The algorithm has linear-time complexity in relation to the number of data samples. Horizontal arrow

symbols in the algorithm description indicate appending to a list, and indexing lists with s, m, and e retrieves list start, middle, and end items respectively. In the second data collection method, we used a route planner and PID controllers. Each episode a random pair of ego-vehicle source and destination are chosen, then the modular pipeline system introduced in [7] is used to avoid obstacles and follow waypoints by making use of simulator privileged information.

Algorithm 2: High-level command generation algorithm

Input: Lists x , y, $intersections$
Parameters: Threshold t indicating minimum absolute change in vehicle coordinates when taking a turn
Output: List $cmds$ of high level command per timestamp

allocate empty list $cmds$ & $i = 0$
while $i <$ ***count***(x) **do**
  allocate 2 empty lists x_in and y_in
  // Check if within intersection
  for s ***in*** $intersections$ **do**
    while ***inside***$(s, x[i], y[i])$ **do**
      $(x\_in, y\_in) \leftarrow (x[i], y[i])$ & $i = i + 1$
      if $i \geq$ ***count***(x) **then break**
    if x_in list is not empty **then break**
  // Assign high-level commands
  if x_in list is not empty **then**
    if $(|x\_in[s] - x\_in[e]| > t)$ **&** $(|y\_in[s] - y\_in[e]| > t)$ **then**
      if $(x\_in[e]-x\_in[s])*(y\_in[m]-y\_in[s])-(y\_in[e]-y\_in[s])*(x\_in[m]-x\_in[s])) \geq 0$ **then**
        $cmd = "GoLeft"$
      else $cmd = "GoRight"$
    else $cmd = "GoStraight"$
    for x_in times **do** $cmds \leftarrow cmd$
  else
    $cmds \leftarrow "FollowLane"$ & $i = i + 1$

As in [6], temporally-correlated noise is injected into the steering during training. The noise simulates gradual drift away from the desired trajectory, then the vehicle is let to recover from these perturbations to provide the network with examples of recovery from unexpected disturbances. During model training, online data augmentation is applied on half of the mini-batch images before feeding them to the network. To augment an image, it is passed through a pipeline of a sequential series of augmentation methods. Each augmentation method in the pipeline has a predefined probability of occurrence which defines the percentage of augmented images having that method existing in their augmentation pipeline. In addition, each augmentation method has stochastic parameters to let each image be augmented differently. As an example, when adding Gaussian noise, for each image to be augmented, the Gaussian noise variance is sampled from a parameterized uniform probability distribution. Two different types of data augmentation methods are adopted. The first type is for photometric transformations: changing brightness, lighting conditions, and applying additive white Gaussian noise and Gaussian blurring [17]. The second type is for geometric transformations: horizontal flipping. In the case of horizontal flipping, the sign of the ground-truth steering wheel angle is flipped as well.

4 Experimental Results

We adopt the experimental setup of the CARLA urban driving benchmark [7] to evaluate the proposed model. The benchmark is conducted in two different towns, one town is used for training data collection, while the other one is kept unseen during training. The parameters for the PID controllers we used during training data recording are tuned in the training town. Four autonomous driving tasks are included; "straight", "Single Turn", and "Navigation", where the route towards the destination in each task has no turns, single turn, more than one turn respectively, and finally in the "Dynamic

Table 1: CARLA urban driving benchmark [7] is composed of 48 experiment sets to evaluate a model; 24 sets for the training town (used during model training) and the same for the testing town. Each set represents a combination of a driving task, a town, and a weather condition, as shown in the table, and is composed of 25 test scenarios. "S", "O", "N", and "DN" stand for "straight", "one (single) turn", "navigation", and "dynamic navigation" tasks respectively.

Experiment ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Task	S	O	N	DN	S	O	N	DN	S	O	N	DN	S	O	N	DN	S	O	N	DN	S	O	N	DN
Weather Condition	Clear Afternoon (Train)				Wet Noon (Train)				Wet Cloudy Noon (Test)				Hard Rain Noon (Train)				Clear Sunset (Train)				Soft Rain Sunset (Test)			

Navigation" task there are random moving vehicles and pedestrians. For each combination of a task, a town, and a weather set, testing is conducted over 25 different test scenarios having predefined start and destination locations, this gives a total of 1200 test scenarios for each model under test. Table 1 describes the benchmark experiments for one town. In each test scenario, the objective is to reach a given destination goal location before a predetermined deadline. The deadline is set to the time needed to reach the goal along the optimal route at a low speed of 10 km/h.

4.1 Success Rate and Distance to Destination

Table 2 benchmarks the proposed model before and after LiDAR fusion with the state-of-the-art CIL model [6] on the CARLA urban driving benchmark [7]. The table reports the autonomous driving success rate on different tasks and test conditions, and the average percentage of distance to goal travelled is available between parentheses. The latter metric provides valuable insight that cannot be inferred from success rate. It is not included in the original benchmark, thus we include the results we record from deploying the CIL pre-trained model in [6].

Table 2: Autonomous Driving success rate average percentage, the average percentage of distance to goal travelled is between parentheses.

Task	Model	Percentages of average success rate and distance to goal			
		Training town		New town	
		Training weathers	New weathers	Training weathers	New weathers
Straight	Camera, [7] results	95 (-)	98 (-)	97 (-)	80 (-)
	Camera, [7] pre-trained	99 (97.2)	**100 (100)**	89 (90.4)	92 (92.7)
	Camera (our data)	**100 (100)**	**100 (100)**	99 (95.7)	**100 (100)**
	Camera + LiDAR	**100 (100)**	**100 (100)**	**100 (100)**	**100 (100)**
Single Turn	Camera, [7] results	89 (-)	90 (-)	59 (-)	48 (-)
	Camera, [7] pre-trained	88 (82.7)	94 (85.5)	56 (54.8)	74 (60.6)
	Camera (our data)	97 (97.3)	98 (97.7)	57 (56.1)	72 (67.2)
	Camera + LiDAR	**100 (100)**	**100 (100)**	**92 (90.0)**	**92 (91.5)**
Navigation	Camera, [7] results	86 (-)	84 (-)	40 (-)	44 (-)
	Camera, [7] pre-trained	78 (88.6)	84 (89.3)	35 (9.7)	58 (45.4)
	Camera (our data)	87 (91.1)	88 (92.5)	33 (16.9)	34 (16.9)
	Camera + LiDAR	**92 (92.7)**	**92 (92.7)**	**68 (77.0)**	**68 (76.9)**
Dynamic Navigation	Camera, [7] results	83 (-)	82 (-)	38 (-)	42 (-)
	Camera, [7] pre-trained	80 (88.3)	74 (81.5)	28 (17.4)	54 (35.1)
	Camera (our data)	84 (91.0)	82 (87.3)	26 (9.5)	30 (29.4)
	Camera + LiDAR	**86 (93.0)**	**86 (92.9)**	**53 (37.5)**	**64 (59.9)**

Table 2 results confirms that our model performs significantly better than the CIL model [6] in all the different combinations of tasks and environmental setups. The results demonstrate that our model improves autonomous driving success rate by 52% when deployed on new towns (town 2) and

weather conditions unseen during training. The learned driving policy consistency against varying weather conditions improved by around 3.9 times.

5 Conclusion

In this work, we extended the state-of-the-art CIL model by fusing laser scanner input with camera, which led to improving autonomous driving performance in terms of generalization to environments that are unseen during training and making performance consistent against varying weather conditions. Our model utilizes a dataset of driving traces recorded automatically, unlike CIL model that is trained on data that is manually collected by a human driver. On the CARLA urban driving benchmark, our model is demonstrated to performs significantly better than state-of-the-art models in all the different combinations of tasks and environmental setups. It significantly improves generalization, in terms of autonomous driving success rate, by 52% and improves consistency against varying weathers by four times. The average distance to goal travelled on all autonomous driving tasks and driving conditions and the collisions rates are significantly improved.

References

[1] Mariusz Bojarski et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[2] Mariusz Bojarski et al. Explaining how a deep neural network trained with end-to-end learning steers a car. *arXiv preprint arXiv:1704.07911*, 2017.

[3] Austin Brown, Brittany Repac, and Jeff Gonder. Autonomous vehicles have a wide range of possible energy impacts. Technical report, NREL, University of Maryland, 2013.

[4] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8748–8757, 2019.

[5] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2722–2730, 2015.

[6] Felipe Codevilla, Matthias Miiller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE, 2018.

[7] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. *arXiv preprint arXiv:1711.03938*, 2017.

[8] Hesham M Eraqi, Yehya Abouelnaga, Mohamed H Saad, and Mohamed N Moustafa. Driver distraction identification with an ensemble of convolutional neural networks. *Journal of Advanced Transportation*, 2019, 2019.

[9] Hesham M. Eraqi, Youssef Emad Eldin, and Mohamed N. Moustafa. Reactive collision avoidance using evolutionary neural networks. In *Proceedings of the 8th International Joint Conference on Computational Intelligence - Volume 1: ECTA, (IJCCI 2016)*, pages 251–257. INSTICC, SciTePress, 2016.

[10] Hesham M. Eraqi, Jens Honer, and Sebastian Zuther. Static free space detection with laser scanner using occupancy grid maps authors. In *IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, Yokohama, Japan, October 2017.

[11] Hesham M. Eraqi, Mohamed N. Moustafa, and Jens Honer. End-to-end deep learning for steering autonomous vehicles considering temporal dependencies. In *Machine Learning for Intelligent Transportation Systems Workshop in the 31st Conference on Neural Information Processing Systems (NIPS)*, Montreal, Canada, December 2017.

[12] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[13] German Ros, Vladlen Koltun, Felipe Codevilla, and Antonio M. Lopez. Carla autonomous driving challenge 2019 results, 2019.

[14] Brian Ichter, James Harrison, and Marco Pavone. Learning sampling distributions for robot motion planning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7087–7094. IEEE, 2018.

[15] Joel Janai, Fatma Güney, Aseem Behl, and Andreas Geiger. Computer vision for autonomous vehicles: Problems, datasets and state-of-the-art. *arXiv preprint arXiv:1704.05519*, 2017.

[16] Maria Jokela, Matti Kutila, and Pasi Pyykönen. Testing and validation of automotive point-cloud sensors in adverse weather conditions. *Applied Sciences*, 9(11):2341, 2019.

[17] Alexander B. Jung et al. imgaug, 2020. Online; accessed 01-Feb-2020.

[18] Jaehoon Jung et al. Efficient and robust lane marking extraction from mobile lidar point clouds. *ISPRS journal of photogrammetry and remote sensing*, 147:1–18, 2019.

[19] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[21] Chengyang Li, Dan Song, Ruofeng Tong, and Min Tang. Illumination-aware faster r-cnn for robust multispectral pedestrian detection. *Pattern Recognition*, 85:161–171, 2019.

[22] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

[23] Luis Martín, Leticia Baena, Laura Garach, Griselda López, and Juan de Oña. Using data mining techniques to road safety improvement in spanish roads. *Procedia-Social and Behavioral Sciences*, 160:607–614, 2014.

[24] Dean A Pomerleau. Alvinn, an autonomous land vehicle in a neural network. Technical report, Carnegie Mellon University, Computer Science Department, 1989.

[25] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[26] DL Strickland. How autonomous vehicles will shape the future of surface transportation. *Testimony of The Honorable David L. Strickland Administrator National Highway Traffic Safety Administration House Committee on Transportation and Infrastructure Subcommittee on Highways and Transit*, 2013.

[27] WHO. Global status report on road safety 2018. *World Health Organization*, 2018.

[28] Sascha Wirges, Tom Fischer, Christoph Stiller, and Jesus Balado Frias. Object detection and classification in occupancy grid maps using deep convolutional networks. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3530–3535. IEEE, 2018.

[29] Huazhe Xu, Yang Gao, Fisher Yu, and Trevor Darrell. End-to-end learning of driving models from large-scale video datasets. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2174–2182, 2017.