UAB
Universitat Autònoma de Barcelona

Advancing Vision-based End-to-End Autonomous Driving

Yi Xiao

Universitat Autònoma de Barcelona

Advancing Vision-based End-to-End Autonomous Driving

A dissertation submitted by **Yi Xiao** to the Universitat Autònoma de Barcelona in fulfilment of the degree of **Doctor of Philosophy** in the Departament de Ciències de la Computació.

Bellaterra, May 10, 2023

Director	**Dr. Antonio Manuel López** Dpt. Ciències de la Computació and Computer Vision Center (CVC) Universitat Autònoma de Barcelona
Thesis committee	**Dr. Aura Hernandez Sabaté** Dpt. Ciències de la Computació and Computer Vision Center (CVC) Universitat Autònoma de Barcelona
	Dr. Thorsten Bagdonat Group Innovation, Future Digital Platform, AI & Data Science Volkswagen AG
	Dr. Germán Ros Sánchez Emergent AI Intel Labs

Centre de Visió per Computador

This document was typeset by the author using LaTeX 2_ε.

ISBN: 978-84-124793-2-4

Printed by Ediciones Gráficas Rey, S.L.

Acknowledgements

Throughout my entire Ph.D., I have been very lucky to receive a great deal of support and assistance. My first and deepest thanks go to my advisor, Prof. Antonio Manuel López, who has been very patient with supervision and mentorship for me over the last five years. Today, I am still amazed by his dedication to academics and his passion for life. I admire him for his rigorous scholarship and great insights into academics, and he can always point out some details that I tend to miss and ignore. Apart from that, I am also inspired by his positive attitude towards life. He always encourages me to cherish and enjoy my time in hobbies, no matter how hard the research work. He used to share with me his paintings, and the bitter and sweet he experienced in the process of learning to paint. It reminds me of a quote from the science fiction *The Three-Body Problem* that once he recommended to me: “Weakness and ignorance are not barriers to survival, but arrogance is." I feel that despite being a very accomplished professor in his work, he always stays humble, eager for new knowledge, and never stops learning, which is one of the most valuable and important things I learned from him.

I would also particularly thank to Dr. Felipe Codevilla, for generously sharing his precious experience, opinions, and knowledge along my academic journey. Whenever I felt lost or got stuck in my study, he always discussed with me and encouraged me to explore and go one step further. I am inspired by his confidence in dealing with problems and his positive attitude toward work. I am so grateful to have had this wonderful time working with him.

I have had the pleasure of discussing my research with many ADAS teammates who always share to me their interesting ideas, including Diego, Gabriel, Akhil, Alexandre, Rubén, Carlos, Shivam, Enrico, Dr. Jose Luis, Dr. Idoia, and so on. My special thanks go to Sanket from Document Analysis Group, who never hesitate to share his thoughts, knowledge, and experience with me.

Through my journey in Barcelona, I am very lucky to have my dearest Chinese friends, who make me feel at home even though I am far away from my family: Lu Yu, Xialei Liu, Yaxing Wang, Xiaoqing Shi, Danna Xue, Chenshen Wu, Yi Zhang, Fei

Yang, Yulu Gao, Yixiong Yang, Shiqi Yang, Kai Wang, Qiuyue Yang, Jiarui Li, Ziwen Wang, Keyao Li, Yachan Guo, Yuyang Liu, Qi Dong, Chuanming Tang, Qingshan Chen, Tao Lian, Huihuang Chen, Zhijie Fang, Anhong, Lei Kang, Lichao Zhang, Yunwei Sheng, Xiaowan Chen, Tingting Qiu, Lang Xiong, Dailu Guan, Baoyi Zhang, Bin Liu, Yu Lian, Cong Wang, Kaicheng Zhang, Shida Xu, Yuting Lu, Ruobin Wu, Siyuan Ren and others. Thanks to all of them for having precious memory with me, especially in this hardest Covid-19 quarantine time for the past few years.

Last but not least, I would like to deeply thank my husband Ziyang Hong, my parents, my parents-in-law, my sister, my brother-in-law, and my little nephew. They have been so supportive all the time and brought a lot of joy to my life. I treasure all their love and accompany. I would also like to give myself a hug, for being brave and persistent through the journey. I want to end up with a quote that had always been kept in my heart: “The struggle finally will make everything all the more worth it." Although my whole Ph.D. was not always full of joy, looking back, I believe it was all the bitterness and sweetness that made it complete and fulfilling.

Abstract

In autonomous driving, artificial intelligence (AI) processes the traffic environment to drive the vehicle to a desired destination. Currently, there are different paradigms that address the development of AI-enabled drivers. On the one hand, we find modular pipelines, which divide the driving task into sub-tasks such as perception, maneuver planning, and control. On the other hand, we find end-to-end driving approaches that attempt to learn the direct mapping of raw data from input sensors to vehicle control signals. The latter are relatively less studied but are gaining popularity as they are less demanding in terms of data labeling. Therefore, in this thesis, our goal is to investigate end-to-end autonomous driving.

We propose to evaluate three approaches to tackle the challenge of end-to-end autonomous driving. First, we focus on the input, considering adding depth information as complementary to RGB data, in order to mimic the human being's ability to estimate the distance to obstacles. Notice that, in the real world, these depth maps can be obtained either from a LiDAR sensor, or a trained monocular depth estimation module, where human labeling is not needed. Then, based on the intuition that the latent space of end-to-end driving models encodes relevant information for driving, we use it as prior knowledge for training an affordance-based driving model. In this case, the trained affordance-based model can achieve good performance while requiring less human-labeled data, and it can provide interpretability regarding driving actions. Finally, we present a new pure vision-based end-to-end driving model termed CIL++, which is trained by imitation learning. CIL++ leverages modern best practices, such as a large horizontal field of view and a self-attention mechanism, which are contributing to the agent's understanding of the driving scene and bringing a better imitation of human drivers. Using training data without any human labeling, our model yields almost expert performance in the CARLA NoCrash benchmark and could rival SOTA models that require large amounts of human-labeled data.

Key words: *deep learning, autonomous driving, end-to-end, imitation learning, multimodality, representation learning*

Resumen

En conducción autónoma, una inteligencia artificial (IA) procesa el entorno para conducir el vehículo al destino deseado. En la actualidad, existen diferentes paradigmas que abordan el desarrollo de conductores dotados de IA. Por un lado, encontramos pipelines modulares, que dividen la tarea de conducción en subtareas como la percepción y la planificación y control de maniobras. Por otro lado, encontramos enfoques de conducción extremo-a-extremo que intentan aprender un mapeo directo de los datos en crudo de los sensores de entrada a las señales que controlan la maniobra del vehículo. Estos últimos enfoques están relativamente menos estudiados, pero están ganando popularidad ya que son menos exigentes en términos de etiquetado manual de datos. Por lo tanto, en esta tesis, nuestro objetivo es investigar la conducción autónoma basada en modelos de extremo-a-extremo.

Estudiamos tres aspectos. En primer lugar, nos centramos en los datos sensoriales de entrada. Consideramos agregar información de profundidad como complemento a la información de apariencia (imagen RGB), para así tener en cuenta la capacidad del ser humano de estimar la distancia a los obstáculos. En el mundo real, estos mapas de profundidad se pueden obtener de un sensor LiDAR o de un modelo de estimación de profundidad monocular, de formar que, en ningún caso, se necesita etiquetado manual de datos. En segundo lugar, basándonos en la hipótesis de que el espacio latente de los modelos extremo-a-extremo codifica información relevante para la conducción, usamos ese espacio latente como conocimiento previo para entrenar un modelo de conducción basado en *affordances*. Este modelo puede conducir correctamente, su entrenamiento requiere menos datos etiquetados manualmente que los pipelines modulares, y mejora la interpretabilidad de las maniobras ejecutadas. En tercer lugar, presentamos un nuevo modelo de conducción de extremo a extremo basado en visión, denominado CIL++, que se entrena mediante imitación. CIL ++ usa un campo de visión horizontal y un mecanismo de auto-atención, que le ayudan a comprender mejor la escena e imitar mejor a los conductores humanos. Así, usando datos de entrenamiento sin etiquetado manual, CIL++ conduce casi al nivel de un experto, como demuestra en las pruebas *CARLA NoCrash*, rivalizando con modelos del estado del arte que sí requieren grandes cantidades de datos etiquetados manualmente para su entrenamiento.

Palabras clave: *aprendizaje profundo, conducción autónoma, modelos extremo-a-extremo, aprendizaje por imitación, multimodalidad, representación del conocimiento*

Resum

En conducció autònoma, una intel·ligència artificial (IA) processa l'entorn per conduir el vehicle a la destinació desitjada. Actualment, hi ha diferents paradigmes que aborden el desenvolupament de conductors dotats d'IA. D'una banda, trobem sistemes modulars, que divideixen la tasca de conducció en sub-tasques com ara la percepció i la planificació i control de maniobres. D'altra banda, trobem enfocaments de conducció extrema-a-extrem que intenten aprendre un mapeig directe de les dades en cru dels sensors d'entrada als senyals que controlen la maniobra del vehicle. Aquests darrers enfocaments estan relativament menys estudiats, però estan guanyant popularitat ja que són menys exigents en termes d'etiquetatge manual de dades. Per tant, en aquesta tesi, el nostre objectiu és investigar la conducció autònoma basada en models d'extrem-a-extrem.

Estudiem tres aspectes. En primer lloc, ens centrem en les dades sensorials d'entrada. Considerem afegir informació de profunditat com a complement a la informació d'aparença (imatge RGB), per tenir en compte així la capacitat de l'ésser humà d'estimar la distància als obstacles. Al món real, aquests mapes de profunditat es poden obtenir d'un sensor LiDAR o d'un model d'estimació de profunditat monocular, de formar que, en cap cas, no cal etiquetatge manual de dades. En segon lloc, basant-nos en la hipòtesi que l'espai latent dels models extrem-a-extrem codifica informació rellevant per a la conducció, fem servir aquest espai latent com a coneixement previ per entrenar un model de conducció basat en *affordances*. Aquest model pot conduir correctament, el seu entrenament requereix menys dades etiquetades manualment que els sistemes modulars i millora la interpretabilitat de les maniobres executades. En tercer lloc, presentem un nou model de conducció extrem-a-extrem basat en visió, anomenat CIL++, que s'entrena mitjançant imitació. CIL ++ utilitza un camp de visió horitzontal i un mecanisme d'auto-atenció, que l'ajuden a comprendre millor l'escena i imitar millor els conductors humans. Així, usant dades d'entrenament sense etiquetatge manual, CIL++ condueix gairebé al nivell d'un expert, com demostra a les proves *CARLA NoCrash*, rivalitzant amb models de l'estat de l'art que sí que requereixen grans quantitats de dades etiquetades manualment per al seu entrenament.

Paraules clau: *aprenentatge profund, conducció autònoma, models extrem-a-extrem, aprenentatge per imitació, multi-modalitat, representació del coneixement*

Contents

List of Figures

List of Tables

1 Introduction

There is an increasing consensus about the need to progressively move towards new models of mobility as a service, aiming at (1) reducing traffic accidents, congestion, and pollution; (2) improving the mobility of temporary or permanent disabled people, as well as the elderly; (3) achieving time efficiency when delivering goods and transporting persons. Autonomous vehicles (AVs) are the core of such new paradigms of mobility.

According to the Society of Automotive Engineers (SAE International), AVs are mainly categorized into the following six levels:

- *Level 0 - No Automation*: driving is fully performed by humans;
- *Level 1 - Driver Assistance*: the vehicles have one automated system that helps with steering or braking, but the vehicles are driven by humans;
- *Level 2 - Partial Automation*: the vehicles have advanced driving assistance systems (ADAS) that can take steering, acceleration, and braking in certain situations, but humans still are the drivers and should take over control if the vehicles encounter situations that the automated systems cannot handle;
- *Level 3 - Conditional Automation*: the vehicles can perform most of the driving, but human override is still required. Mainly this is designed for tasks that do not require a lot of complex maneuvering, *e.g.*, long highway driving;
- *Level 4 - High Automation*: the vehicles self-drive under specific circumstances. A human override is an option, but the vehicles are programmed to stop themselves if the automatic driving systems fail;
- *Level 5 - Full Automation*: the vehicles fully perform the driving under all conditions. No human attention or interaction is required, and the steering wheel, brake/accelerator pedals are not even needed.

As of 2023, AVs operating at Level 3 and above remain a marginal portion of the market.

Developing AVs is a tremendous challenge involving multi-disciplinary topics, *e.g.*, cyber-security, vehicle-to-X communication, environment perception, localization, route planning, and maneuver control; all governed by ethical and legal aspects. Focusing on the scientific-technical topics, one of the essential challenges nowadays is how to exploit artificial intelligence (AI), in the form of machine/deep learning, to efficiently process the vast amount of data for building up systems that enable autonomous driving. Although some AVs were unveiled before the 1980s, it was AI which made possible AVs with higher automation in the last three decades.

The first attempt at AI for autonomous driving can be traced back to the early 1980s, when Ernst Dickmanns and his team conducted self-driving experiments on a Mercedes-Benz van named VaMoRs. VaMoRs was equipped with sensors and a software system for translating sensory data into appropriate driving controls for the steering wheel, throttle, and brake pedals. By 1987, it was capable of self-driving at a speed of up to 96 kilometers per hour. In 1989, a model named ALVINN [131] from Carnegie Mellon University (CMU) pioneered the use of neural networks to predict steering for the task of the road following. In 1995, CMU NAVLAB completed a cross-country journey for 3,100 miles, in which the 98% of the route was driven autonomously. These models can be categorized as *Level 1* since they only have one automated system that helps with steering, while throttle and brake are still under the control of human drivers for safety reasons. Another milestone of AI models for AVs, named Stanley [171], came out in the second Defense Advanced Research Projects Agency (DARPA) Autonomous Vehicle Challenge in 2005. Stanley relied on a software pipeline with many individual modules powered by machine learning techniques for intermediate tasks, *e.g.*, finding the path, detecting obstacles, and staying on the course while avoiding them. Stanley can be seen as an AV of *Level 2* because it determined steering, throttle, and brake values for driving, but it can only handle limited types of obstacles in static environments without interaction with moving traffic. These pioneering works show the dawn of AI in autonomous driving. Since that, AI has been the biggest force of breakthroughs in the creation of AVs, drawing more interest and attentions on AVs of *Level 3* or above.

1.1 Modular Pipeline and End-to-End

Accordingly, the research community is gradually exploring different approaches for developing AI drivers. These approaches can be broadly summarized in two paradigms, *Modular Pipeline* and *End-to-End*.

Modular Pipeline. The modular pipeline, which is also known as the perception-planning-action pipeline, refers to a system that consists of clearly defined modules

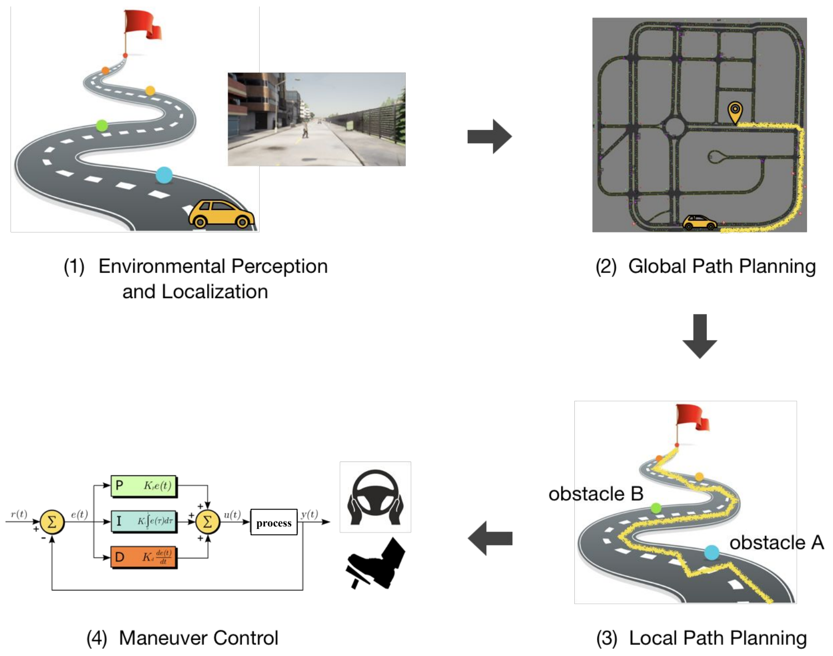


Figure 1.1: An autonomous driving system following the modular pipeline consists of four modules: environmental perception and localization, global path planning, local path planning, and control.

with particular responsibilities. More specifically, the system can be hierarchically decomposed into four components [68], shown in Figure 1.1: (1) environmental perception [57, 85] and localization: firstly, an AV should apprehend the environment and localize itself; (2) global path planning: then, according to the starting location, the road network should provide the best-planned route guiding the AV to the destination; (3) local path planning [6, 137, 177]: next, given the global planned route, a set of continuous waypoints are generated taking into account the dynamic obstacles and constraints to guide the future maneuver; (4) maneuver control [125, 153]: finally, a control system (*e.g.*, a PID controller) is designed for the execution of the motion.

Just developing the environmental perception is already an especially complex research topic, since we can find a plethora of associated sub-tasks, *e.g.*, object detection (2D image-based [110, 139, 140], 3D image-based [25, 117], 3D LiDAR-

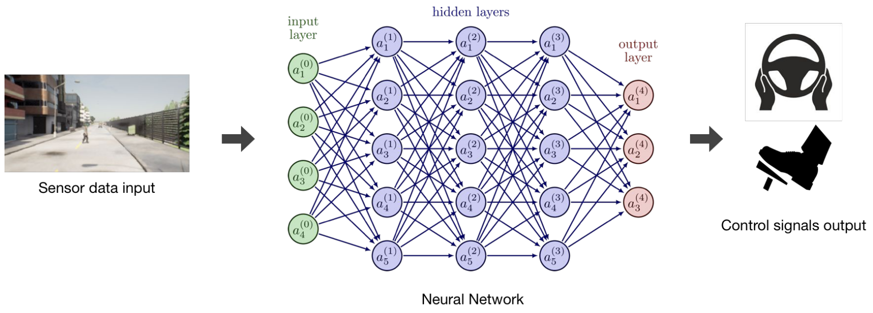


Figure 1.2: An end-to-end autonomous driving model takes images as input and uses a neural network to directly output maneuver control signals for driving.

based [98, 135, 161, 190, 191, 201], and multimodal [10, 32, 51, 66, 96, 101, 129, 182]); object tracking (image-based [18, 36, 115, 132, 156, 180, 184], multimodal [47]); traffic sign recognition [203]; semantic segmentation (by class/category [111, 122, 195] and object instances [13, 107, 174], usually based on images of the visual spectrum but also on multimodal data [71, 152]); monocular depth estimation [58, 59, 64, 70]; stixel-world representation (based on stereo images [12, 42, 78, 151], monocular [16, 23], and multimodal [130]); as well as SLAM and place recognition [22, 33, 102, 136, 142, 158, 170, 193, 198, 202]. Apart from perception, there is also a plethora of different approaches for motion planning with AI [6, 137, 163, 177].

All these sub-tasks are combined for building up the whole modular system requiring great tuning and adaptation. On the one hand, this traditional divide-and-conquer engineering principle holds its benefits of interpretability, *i.e.*, being easier to debug when unexpected failures occur in the system [168]. However, it usually requires data from different modalities, which genuinely comes with a price of a huge amount of cost and human effort on data labeling.

End-to-End. As an alternative to the modular pipeline, the end-to-end paradigm has become another mainstream in the field of autonomous driving [20, 28, 38, 40, 80, 100, 131, 188, 199]. This paradigm proposes to directly map the sensory inputs to the maneuver control signals through a neural network, instead of using intermediate modules to deal with specific tasks of perception, path planning, and vehicle control. We show this paradigm in Figure 1.2. The outputs of the model will be the steering, acceleration, and brake values that can be directly performed in actual driving. These holistic sensorimotor driving models can be closer to human nature, since humans learn to perceive the environment and take actions simultaneously.

One obvious advantage of end-to-end models is that they are simpler and more straightforward in training. Besides, the appealing advantage of this paradigm is that the supervision required for training these models consists of the vehicle's state variables, which can be automatically collected from fleets of human-driven vehicles without any need for human labeling. Due to the difficulty of interpreting the relationship between image content and inferred driving actions, the reliability of these models is controversial [44], which has become one of the main concerns. In this thesis, we focus on this *end-to-end* paradigm.

1.2 Reinforcement and Imitation Learning

For an apprentice agent (*i.e.*, an end-to-end model), its goal is to learn a policy to accomplish a specific task. This policy refers to *"what actions it should take, given its current state and the state of the environment"*. For instance, in end-to-end driving, the agent tries to learn a driving policy, *i.e.*, given the observation from the environment and vehicle's state, how to control the steering wheel, throttle, and brake pedals. In recent years, two strategies have been extensively studied for apprentice agents to learn policies: *Reinforcement Learning* [29, 86, 128, 173, 199] and *Imitation Learning* [28, 39, 80, 83, 126, 185]. Following reinforcement learning, apprentice agents try to explore the policy in a self-discovery manner, while in imitation learning, the agents try to mimic the behavior of expert agents who provide demonstrations.

Reinforcement Learning (RL). It refers to an apprentice agent learning to perform a task by making decisions and receiving feedback from its environment in the form of a reward. Basically, most RL problems can be mathematically modeled as a Markov Decision Process (MDP) [15], which is a mathematical framework describing a discrete-time stochastic control process. An MDP involves four elements:

- a set of agent states S;
- a set of actions A;
- a transition function providing the probability of reaching state s_{t+1} from state s_t as a consequence of action a_t;
- a function providing a reward for the agent in case of reaching state s_{t+1} from state s_t as a consequence of action a_t.

In general, the transition and reward functions are often considered as modeling the environment, since they govern its dynamics. Since these two functions are unknown, the apprentice agent needs to interact with the environment and estimate the policy by trial and error. By maximizing the long-term cumulative rewards, the agent iteratively fine-tunes the estimated policy to ultimately obtain an optimal one.

Many agents trained with RL have already shown their potential of surpassing human performance in several fields [29], *e.g.*, robot manipulation [87, 121], game playing [97, 116] and Go [159, 160]. However, in the field of autonomous driving, most of the existing RL agents can only handle very simple scenarios or sub-tasks [29]. We can find some works that focus on using RL to learn a policy for lane following [90, 146], and also some works that aim at handling decision-making on traffic light [30], as well as some works that put an effort on the control part [43, 106]. RL models that are trained for end-to-end autonomous driving have not been extensively studied so far.

One advantage of RL method is that the training of the agents does not rely on expert data. Hence, they can bypass the pitfalls of dataset bias/distribution shift [40] and causal confusion [44]. These RL-trained agents might explore more innovative policies since even some dangerous cases that are out of the expert data distribution can occur during training. Thus, RL allows apprentice agents to explore new rules based on some concrete problems. In addition, it can be useful when the task to be solved requires much more interaction with the environment. However, RL comes with its challenges. Given the sparsity of the reward (*e.g.*, we only receive a reward when the game is won or lost), RL models usually require a much larger amount of episodes for training, thus being more time-consuming. For instance, AlphaGoZero [160] has gone through five million games of Go to beat the human world champion. In addition, it is risky to train agents by trial-and-error, specifically in more complex tasks such as robot manipulation and autonomous driving. The high cost of maintenance and the lack of reproducibility are also the major limitations of RL models. Moreover, sometimes it can be very hard, or even impossible to manually design a reward function for certain environments.

Imitation Learning (IL). Unlike RL, IL does not need to specify an explicit reward function to learn a policy. Instead, a set of demonstrations (*i.e.*, observation-action pairs) are provided by an expert agent, then the apprentice agent implicitly learns the policy by mapping the observation to the action. Some previous works in different domains, such as piloting an aircraft [148], robot manipulation [138, 165] and autonomous driving [39,61,80,126] have shown that IL is a useful and promising method that deserves further research.

IL does not face the issues of sparse rewards or manually defining an explicit reward function to satisfy the desired behavior which can be extremely complicated in some scenarios. Certainly, the drawbacks of IL can not be ignored. As a data-driven method, IL needs to address the dataset bias problem [40]. Specifically, during training when the expert demonstrations are provided, the models try to mimic the behavior of the expert agent only in universal samples, which makes it hard to generalize well in rare cases for which demonstrations are not provided.

Collectively, there are two typical forms of IL that are highly mentioned [172]: *Behavior Cloning* (BC) [147] and *Inverse Reinforcement Learning* (IRL) [120]. The models trained via BC aim at directly learning the policy via the supervision of expert demonstrations, without inferring an explicit reward function. Unlike BC, IRL first leverages demonstrations to infer a reward function which is then used in RL to learn the policy for generating the demonstrations. As its name suggests, IRL is understood as the inverse of RL. This is because the goal of an RL model is to find the optimal policy according to the rewards received from a reward function, while in IRL, the optimal policy is provided in the form of demonstrations, and the goal for the model is to recover the reward function. In other words, IRL can be seen as a learning method in which a set of demonstrations from the expert are provided for the apprentice agent to learn how to mimic the expert's behavior. This is why IRL is categorized as a form of IL.

After all, a part of IRL still involves RL, which means it is still expensive in terms of computational time and safety because of the needed interaction between the agent and the environment. Although some work [40, 143] have pointed out that BC models can be unstable in performance due to the so-called covariate shift problem [164], BC models hold obvious benefices such as simplicity (*e.g.*, the training requires only demonstration data) and efficiency (*e.g.*, no interaction between the agent and the environment is needed), which makes this approach to be more popular and studied for autonomous driving [39, 40, 145, 185, 186].

In recent years, Codevilla *et al.* [40] proposed to train an end-to-end autonomous driving model by what they called conditional imitation learning (CIL). Specifically, the model is trained by demonstrations based on expert driving, but, in addition, the model is able to accept high-level commands such as *turn left in the next intersection*, in order to be navigated toward the desired destination. In this thesis, CIL is taken as the training approach for our proposals.

1.3 Supervision

Regarding the type of supervision required to train the deep models enabling autonomous driving, we can find three cases: *Pixel-based*, *Image-based*, and *Signal-based*. Figure 1.3 shows their differences.

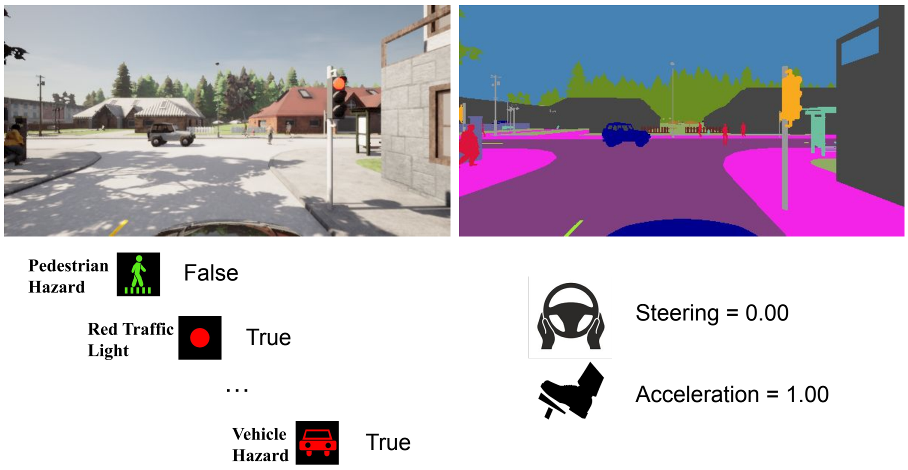


Figure 1.3: Three different levels of ground truth data used as supervision to train deep models for autonomous driving. Top left: RGB image in a driving scene. Top right: pixel-wise semantic supervision usually provided by human labeling. Bottom left: Image-level driving affordances, also provided by human labeling. Bottom right: supervision from vehicle signals, which can be automatically collected onboard, thus not requiring offline human labeling.

Pixel-based Supervision. It refers to attaching supervision to each pixel/voxel of the sensor raw data (images, point clouds, *etc.*) for training deep models, such as semantic/instance segmentation [8, 54, 69, 108, 111, 169, 179, 194]. In autonomous driving, examples are lane segmentation [5,119], pedestrian segmentation [109,175] and dynamic objects tracking [103]. Sometimes this supervision is used just as an auxiliary task for training models, for instance, for training sensorimotor models [80, 199]. Although in practice, usually, the best performing models arise from such pixel-based supervised training, this approach genuinely comes with a cost, especially when we want to deploy the system in the physical world. In particular, the data labeling is costly to collect and their accuracy is difficult to guarantee, which has a great impact on training performance.

Image-based Supervision. Since densely human-labeled data is difficult to obtain, some work [88, 91] focus on approaches that require sparser/weaker human labeling in the form of 2D bounding boxes, image tags, global constraints or scribbles *etc.* This appealing idea has also inspired some work in autonomous driving. For instance, in [26, 150, 186], the authors propose image-based driving affordances as labels to supervise a deep model in a direct perception approach, where an additional controller is then used for maneuvering an autonomous vehicle. Collectively, such affordances can be understood as a relatively small set of interpretable variables describing events that are relevant for an agent acting in an environment [62], which can bring better interpretability for a deep driving model. Compared to pixel-based annotated data, to a certain extent, such image-based annotated data indeed can ease the curse of human labeling.

Signal-based Supervision. In addition to the sensors used to access the environment (*e.g.*, cameras, LiDARs, *etc*), other onboard sensors (*e.g.*, those providing steering angle, acceleration/brake, waypoints, *etc*) can be used to supervise the training of end-to-end models, which we term as signal-based supervision. In autonomous driving [39, 80, 185, 199], these signal data monitor the interaction of the driving expert and the AV. One typical example is CIL model, where the input data are RGB images, speeds, and high-level commands, and the data used as supervision are control signals directly obtained from onboard sensors. It greatly reduces the cost of human labeling, alleviating the problem of data hunger for training deep models.

1.4 Waypoint-based and Action-based End-to-End

Regarding end-to-end autonomous driving models, the basic approach is to map the raw images from RGB cameras to a low-level space. This output space can be either a set of navigation waypoints or signals used to maneuver the vehicle. We term these two kinds of methods as *Waypoint-based End-to-End* and *Action-based End-to-End*.

Waypoint-based End-to-End. Waypoint-based end-to-end models are trained to output a local trajectory indicating what waypoints in the next few time steps the agent is supposed to follow. After that, with these predicted waypoints, additional controllers are designed for further providing the proper control signals (*i.e.*, steering, throttle, and brake) in driving. Several works [14, 27, 28, 35, 118, 133, 155] have shown that this is a feasible method in autonomous driving. For instance, Chen *et al.* [28] proposed to divide the training of an autonomous driving model into two steps: first, a privileged agent who has access to bird-eye-view(BeV) information of the environment was trained to predict some sets of waypoints according to different commands. These corresponding waypoints are given to a low-level controller to output the control signals. Then, the sensorimotor agent received RGB images from a forward-facing camera and produced waypoints in the reference frame of the RGB camera, then, these waypoints are projected into the vehicle's coordinate frame and passed to a low-level controller.

On the one hand, this approach holds the benefit of taking into account short-term time-series information (*i.e.*, the trajectory prediction usually consists of waypoints in a few future time steps), which provides the flexibility of adjustment. For instance, an error correction system for future steps can be properly designed to prevent the agent from going out of track. Also, such a system can be easier to interpret since it is divided into two modules: perception&planing and control. Moreover, this system can be used together with multi-agent trajectory prediction, in order to consider the interaction between dynamic objects. However, on the other hand, this approach requires tuning additional controllers to follow the predicted trajectory. Moreover, generating waypoints on BeV coordinates for supervising a model, involves 3D knowledge of the scene.

Action-based End-to-End. Action-based end-to-end models output control signals which can act on the vehicle either directly or after some signal-stabilizer filtering (*e.g.*, using a PID) [17, 39, 40, 80, 105, 126, 185, 199], thus, following a more pure end-to-end style. Action-based end-to-end models are usually simpler, and more straightforward in data collection, training and actual driving, and they do not require parameters tuning for extra controllers. However, these models lack diagnosability when they deviate from the desired trajectory and lack predictability since they cannot provide a sequence of output for the short-term future.

Given their pros and cons, in end-to-end autonomous driving, there is no clear conclusion on which of these two approaches is better. In [181], both of them are combined. The actions applied in driving are a weighted combination of two output branches of the model: one for the control signals, and the other for the waypoints that are further processed by a controller, in order to achieve complementary advantages. In this thesis, considering the fact that the *Action-based End-to-End* is closer to pure end-to-end driving, we focus on this approach.

1.5 Offline and Online Evaluation

Nowadays, we can find many public datasets in the field of autonomous driving, *e.g.*, Cityscapes [41], KITTI [60], ApolloScape [81], nuScenes [24] and Waymo [53, 167], involving sensor data such as RGB images, LiDAR and Radar point clouds. For some perception tasks, the evaluation of the respective models can be satisfied using these static datasets. For instance, for a model trained for pedestrian detection, we can use the ground truth and the bounding box predictions on the static data to compute the Average Precision (AP) and mean Average Precision (mAP), in order to know how well the model is performing; for a model trained for road semantic segmentation, we can use the pixel-wise ground truth and the segmentation predictions on the static data to compute their overlap as a metric. In general, these evaluation metrics are correlated to the performance of the model. This kind of evaluation allows us to test the effectiveness of the model with static information, thus it can be named *Offline Evaluation*.

However, to develop an end-to-end autonomous driving model, using static datasets to assess models is not sufficient. This is because driving actions at time step t might lead to different observations at time step $t+n$, which most probably differ from those in the static datasets. In this case, defining evaluation metrics using static datasets can not correlate well with actual driving performance [38]. Considering this, there are many works [27, 28, 39, 80, 155, 181, 185] rely on driving simulators to assess the performance of end-to-end models, which is named as *Online Evaluation*. In fact, even though the performance of individual modules from modular pipelines can be assessed by offline evaluation, online evaluation is still required to assess the driving performance of the full modular system.

As in many recent works on end-to-end driving [39, 80, 104, 105, 118, 141, 150, 178, 185, 199], all experiments in this thesis are conducted on the CARLA simulator platform [50]. CARLA is an open-source simulator that provides training and testing environments and facilities for developing autonomous driving systems. It includes multiple towns with single- and multi-lane roads and supports various traffic conditions (*e.g.*, we can set a different amount of dynamic objects) and scenarios (*e.g.*, we can define corner cases) under different lighting and weathers. Moreover, we can generate a large range of available sensor data, which can be efficiently used for initial experiments of the models that might be further developed in the physical world. With the development of the CARLA simulator, three benchmarks (the CARLA original, NoCrash, and Leaderboard) have been presented for researchers to compare their autonomous driving systems through *Online Evaluation*.

1.6 Goal and Outline

The overall objective of this thesis is to contribute to the research on pure vision-based end-to-end autonomous driving leveraging the CIL model. This model was trained in an end-to-end approach, using RGB images, speed, and high-level commands as inputs, and control signals (*i.e.*, steering, acceleration/brake) collecting during expert driving as supervision. In this way, the training data does not require any human labeling, and the trained driving model is able to output control signals that can directly maneuver the vehicle, without explicitly human-defined controllers. We leverage the simplicity of this end-to-end driving model, aiming at improving its performance. In particular, our contributions are:

- Chapter 2: we address the question *Can an end-to-end driving model be improved by using multimodal sensor data over just relying on a single modality?* We assume color images (RGB) and depth (D) as single modalities, and RGBD as multimodal data. Specifically, we explore RGBD from the perspective of early, mid, and late fusions of the RGB and D modalities. The presented results show that multimodal RGBD end-to-end driving models outperform their single-modal counterparts. Moreover, early fusion shows better performance than mid and late fusion schemes. In addition, multisensory RGBD (*i.e.*, based on camera and LiDAR) outperforms monocular RGBD; however, we conclude that it is worth pursuing the special case of single-sensor multimodal end-to-end models.

- Chapter 3: we address the question *Can an action-based end-to-end driving model encode useful driving-related information in its latent space?* We believe this encoded latent space may be an effective pre-training strategy for learning a direct perception model. Specifically, we first train several kinds of action-based end-to-end models as representation learning encoders. In the second stage, these learned encoders are used to train deep models that focus on predicting driving affordances, from which an additional controller was designed to maneuver the AV. The presented results show that this strategy enables a significant reduction in the number of human-labeled data required to train an interpretable driving affordance model, thus, keeping a major advantage of modular pipelines. Further, our approach improves over other recent pre-training proposals such as contrastive methods and even over ImageNet (supervised) pre-training. We also show that expert driving data (*i.e.*, coming from human drivers) is an important source for learning a good representation. To the best of our knowledge, this work is the first to show that expert demonstrations can act as an effective action-based representation learning technique.

- Chapter 4: we address the question *Can an end-to-end model be improved by considering conditions closer to human driving?* We propose a model termed CIL++, which is improved on the basis of CIL, providing a new strong pure vision-based end-to-end driving baseline. First, we drastically increase the image view by using a horizontal field of view (HFOV) similar to human drivers, which runs on 180° to 220°. Further, we propose to use a visual transformer [49] as a mid-level attention mechanism for associating feature map patches across different views. These changes allow CIL++ to perform at the expert level on the CARLA NoCrash benchmark. To the best of our knowledge, CIL++ is the first pure vision-based end-to-end driving model capable of obtaining competitive results on complex CARLA towns compared to SOTA driving models that require large amounts of human-labeled data.

Finally, in Chapter 5, we draw the global conclusions arising from the whole Ph.D. work and prospect future work.

2 Multimodal End-to-End Autonomous Driving

2.1 Introduction

Autonomous vehicles (AVs) are core for future mobility. Along with many machine learning techniques that have been proposed in the past few decades, developing artificial intelligence (AI) for driving AVs is seen as a promising topic. Two main paradigms are under research, namely, *modular pipelines* and *end-to-end driving*.

The modular paradigm attaches to the traditional divide-and-conquer engineering principle since AI drivers rely on modules with identifiable responsibilities; for instance, to provide environmental perception [57, 85], as well as route planning and maneuver control [125, 153]. As for the perception, it involves many specific complex sub-tasks, including but not limited to semantic segmentation [71, 111, 122, 152, 174, 195], monocular depth estimation [58, 59, 64, 70], object detection [10, 101, 129, 135, 140, 191], tracking [18, 36, 47, 180, 184], SLAM and place recognition [22, 33, 136, 158, 170, 193, 202] and traffic sign recognition [203], *etc.*

The end-to-end driving paradigm focuses on learning holistic models that are able to directly map raw sensor data into control signals for maneuvering AVs [20, 100, 131, 188], *i.e.*, without forcing explicit sub-tasks related to perception or planning. In other words, it advocates learning to perceive and act simultaneously, as humans do. Such sensorimotor models are obtained through a data-driven supervised learning process, which is characteristic of modern AI. End-to-end driving models can accept high-level navigation commands [39, 82, 105, 178], or be restricted to specific navigation sub-tasks such as lane keeping [34, 52, 83] and longitudinal control [61].

Driving paradigms highly rely on convolutional neural networks (CNNs). In this context, one of the main advantages of the modular paradigm is the ability to explain the decisions of AI drivers in terms of its modules, which is more difficult for pure end-to-end driving models [21, 93, 104]. However, developing some of the critical modules of the modular paradigm requires hundreds of thousands of supervised data samples [79, 166], *e.g.*, raw sensor data with ground truth (GT). Mostly the GT is provided manually (*e.g.*, human labeling of object bounding boxes [60], pixel-level delineation of semantic classes [41]), thus being a major bottleneck for this paradigm. Conversely, end-to-end approaches enable CNN-based models to learn driving from raw sensor data (*i.e.*, without human-labeled GT) and associated supervision in terms of vehicle's variables (*e.g.*, steering angle, speed, geo-localization, and orientation [113, 149, 188]); note that such supervision does not require human intervention in terms of explicitly labeling the content of the raw sensor data. Moreover, end-to-end models are demonstrating an *unreasonable* effectiveness in practice [20, 34, 39, 61], which makes this research worthy of further exploration.

Although AVs will be multisensory platforms, equipping and maintaining on-board synchronized heterogeneous sensors are quite expensive nowadays. As a consequence, most end-to-end driving models rely only on vision [20, 26, 34, 39, 61, 83, 100, 118, 150, 188, 192], *i.e.*, which are visuomotor models. This idea is inherently reasonable since human drivers mainly rely on vision. However, multimodality has shown better performance in key perception sub-tasks such as object detection [10, 32, 65, 66, 96, 101, 129, 135, 182], tracking [47], and semantic segmentation [71, 152]. Thus, exploring multimodality for end-to-end driving is worthwhile.

Accordingly, in this chapter, we address the question *Can an end-to-end driving model be improved by using multimodal sensor data over just relying on a single modality?* In particular, we assume color images (RGB) and depth (D) as single modalities, and RGBD as multimodal data. Due to its capability of accepting high-level commands, this study is based on the CNN architecture known as *conditional imitation learning* (CIL) [39]. We explore RGBD from the perspective of early, mid, and late fusion of the RGB and D modalities. Moreover, as in many recent works on end-to-end driving [39, 104, 105, 118, 141, 150, 178], our experiments rely on the CARLA simulator [50].

The presented results show that multimodal RGBD end-to-end driving models outperform their single-modal counterparts. Moreover, the early fusion scheme shows better performance than the mid and late fusion schemes. In addition, multisensory RGBD (*i.e.*, RGB and D are from camera and LiDAR) outperforms monocular RGBD (*i.e.*, D is generated from a monocular depth estimation model); however, we conclude that it is worth pursuing this special case of single-sensor multimodal end-to-end models.

We present the work as follows: Section 2.2 reviews the related literature. Section 2.3 presents the used CIL architecture from the perspective of early, mid, and late fusion schemes. Section 2.4 summarizes the experimental setting and the obtained results. Finally, Section 2.5 draws the main conclusions and future work.

2.2 Related Work

This section focuses on two main related topics: *multimodal perception* and *end-to-end driving models learned by imitation.*

2.2.1 Multimodality

Object detection is one of the perception tasks for which multimodality has received the most attention. Enzweiler *et al.* [51] developed a pedestrian detector using hand-crafted features and shallow classifiers combined as a mixture of experts (MoE), where multimodality relies on image luminance and stereo depth. Gonzalez *et al.* [66] detected vehicles, pedestrians, and cyclists (vulnerable road users (VRUs)), using a multimodal MoE based on space-time calibrated RGB and LiDAR depth. Chen *et al.* [32] used calibrated RGB and LiDAR depth as input for a CNN-based detector of vehicles and VRUs, which is a current trend [10,32,96,129,182]. Some of these works are inspired by Faster R-CNN [140], since they consist of a first stage for proposing regions potentially containing objects of interest, and a second stage performing the classification of those regions to provide final object detection; *i.e.*, following a mid-level (deep) fusion scheme where CNN layers of features from the different modalities are fused in both stages [32,96,182]. Other alternatives are early fusion at raw data level [129], the late fusion of independent detectors [10,129], or just using different modalities at separated steps of the detection pipeline [135]. Other approaches focus on multispectral appearance, as in Li *et al.* [101], where different fusion schemes for RGB and Far Infrared (FIR) calibrated images are compared.

All these studies and recent surveys [9,55] show that detection accuracy increases with multimodality. Therefore, more perception tasks have been addressed under the multimodal approach. Dimitrievski *et al.* [47] proposed a pedestrian tracker that fuses camera and LiDAR detections to solve the data association step of their tracking-by-detection approach. Schneider *et al.* [152] proposed a CNN architecture for semantic segmentation that performs a mid-level fusion of RGB and stereo depth, leading to a more accurate segmentation of small objects. Ha *et al.* [71] proposed a mid-level RGB and FIR fusion approach in a CNN architecture for semantic segmentation. Piewak *et al.* [130] used a mid-level fusion of LiDAR and camera data to produce a Stixel representation of the driving scene, showing improved accuracy in terms of geometry and semantics of the resulting representation.

In this chapter, rather than focusing on individual perception tasks such as object detection, tracking, or semantic segmentation, we challenge multimodality in the context of end-to-end driving, exploring early, mid, and late fusion schemes.

2.2.2 End-to-End Driving

Three decades ago, Pomerleau *et al.* presented ALVINN [131], a shallow sensorimotor neural network that was able to perform end-to-end road following without obstacles in scene. ALVINN controlled a CMU's van, NAVLAB, along a 400m straight path at ~ 2 Km/h and under good weather conditions. Although the addressed scenario is extremely simple compared to driving in real traffic, it was already necessary to simulate data for training the sensorimotor model. In fact, in their work, camera images (30×32 pixels, blue channel) were already combined with laser range finder data (8×32 depth cells) via an early fusion scheme. LeCun *et al.* [100] trained an end-to-end 6-layer CNN for off-road obstacle avoidance using image pairs (from a stereo rig) as input. Such CNN was able to control a 50cm-length four-wheel truck, DAVE, for avoiding obstacles at a speed of ~ 7 Km/h. During data collection for training, the truck was remotely controlled by a human operator, thus, the CNN was trained according to imitation learning (teleoperation-based demonstration [7]). More recently, Bojarski *et al.* [20] developed a vision-based end-to-end driving CNN which was able to control the steering wheel of a real car in different traffic conditions. In this case, since throttle and brake are not controlled, neither lane and road changing are considered, nor stop-and-go maneuvers.

These pioneering works inspired new proposals based on imitation learning for CNNs. Eraqi *et al.* [52] applied vision-based end-to-end control of the steering angle (neither throttle nor break), focusing on including temporal reasoning by means of long short-term memory recurrent neural networks (LSTMs). Training and testing were done in the Comma.ai dataset [149]. George *et al.* [61] applied similar ideas for controlling the speed of the car. Xu *et al.* [188] presented the BDD dataset and focused on vision-based prediction of the steering angle using a fully convolutional network (FCN) and an LSTM, forcing semantic segmentation as an auxiliary training task. Innocenti *et al.* [83] performed vision-based end-to-end steering angle prediction for lane keeping on private datasets, and Chen *et al.* [34] in the Comma.ai dataset.

Affordances have been proposed as intermediate tasks between environmental perception and prediction of the vehicle control parameters [26, 150]. Affordances do not need to solve perception sub-tasks such as explicit object detection, *etc*; but they form a compact set of factors that influence driving according to prior human knowledge. Chen *et al.* [26] evaluated them on the TORCS simulator [183], thus in car racing conditions (no pedestrians, no intersections, *etc.*) under clean and dry weather; while Sauer *et al.* [150] used the CARLA simulator, which supports regular traffic conditions under different lighting and weather [50]. Muller *et al.* [118] developed a vision-based CNN with an intermediate road segmentation task for learning to perform vehicle maneuvers in a semantic space; the driving policy consists of predicting waypoints within the segmented road and applying a low-level PID controller afterward. Training and testing are done in CARLA, but neither other vehicles nor pedestrians are included. Using LiDAR data, Rhinehart *et al.* [141] combined imitation learning and model-based reinforcement learning to predict expert-like vehicle trajectories, relying on CARLA but without dynamic traffic participants.

These end-to-end driving models do not accept high-level navigation instructions such as *turn left at the next intersection* (without providing explicit distance information), which can come from a global planner or as voice commands from passengers of AVs. Hubschneider *et al.* [82] proposed to feed a turn indicator in the vision-based CNN driving model by concatenating it with features of a mid-level fully connected layer of the CNN. Codevilla *et al.* [39] proposed a more concrete method, in which a vision-based CNN consisting of two blocks: 1) an initial block common to all navigation instructions for processing input and extract features; 2) a second block branched from the initial block according to a subset of navigation instructions (at next intersection turn-left/turn-right/go-straight, or just keep lane). In the first block, the state of the vehicle is also incorporated as part of the mid-level feature of the CNN; in particular, the current speed is encoded since it highly relates to the output of steering angle, throttle, and break (Yang *et al.* [192] have reported the usefulness of speed feedback in end-to-end driving). The overall approach is termed Conditional Imitation Learning (CIL). Experiments were performed in CARLA for different traffic situations (including other vehicles and pedestrians), lighting, and weather conditions. There are many works leveraged from CIL, such as Muller *et al.* and Sauer *et al.* that we mentioned above. Apart from them, Liang *et al.* [105] used CIL as an imitation learning stage before refining the resulting model by applying reinforcement learning. Wang *et al.* [178] used CIL incorporating ego-vehicle heading information at the same CNN-layer level as speed. All these works focus on vision-based end-to-end driving. In our work, we explore multimodal end-to-end driving based on RGB and depth, which can be complementary to most of the cited papers. Without losing generality, we chose CIL as the core CNN architecture due to its effectiveness and increasing use.

Focusing on multimodality, Sobh *et al.* [162] used CARLA to propose a CIL-based driving approach modified to process camera and LiDAR data. In this case, the information fusion is done by a mid-level approach; in particular, before fusion, RGB images are used to generate a semantic segmentation which corresponds to one of the information streams reaching the fusion layers, and there are two more independent streams based on LiDAR, one encoding a bird view and the other a polar grid mapping. Khan *et al.* [92] used CARLA to propose an end-to-end driving CNN based on RGB and depth images, which predicts only the steering angle, assuming that neither other vehicles nor pedestrians are present. In the first

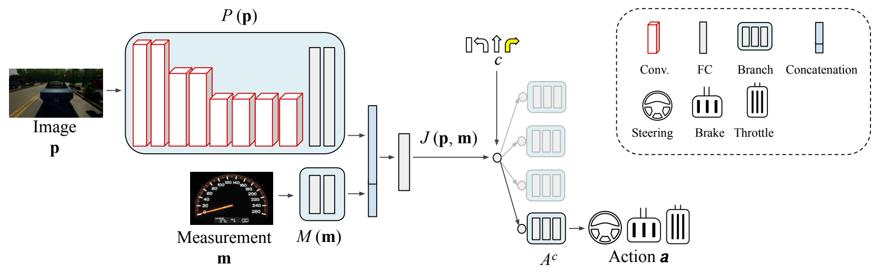


Figure 2.1: CIL architecture: vehicle maneuvers (actions) in the form of the triplet < steering angle, throttle, brake>, depend on a high-level route navigation command (branch selector) running on *turn-left, turn-right, go-straight, continue* , as well as observations in the form of perception data (*e.g.*, an RBG image) and vehicle state measurements (*e.g.*, speed).

step, the CNN is trained by only depth information (taken as the Z-buffer produced by UE4, the game engine behind CARLA). This CNN has an initial block of layers (CNN encoder) that outputs depth-based features, which are later used to predict the steering angle with a second block of fully connected layers. In the second step, this angle-prediction block is discarded and replaced by a new fully connected one. This new block relies on the fusion of the depth-based features and a semantic segmentation produced by a new CNN block that processes the RGB image paired with the depth image. During training, semantic segmentation is conditioned to depth-based features due to the fusion block and back-propagation. This approach can be considered a type of mid-level fusion.

In contrast to these multimodal end-to-end driving approaches, we assess early, mid, and late fusion schemes without forcing intermediate representations which are not trivial to obtain (*e.g.*, semantic segmentation is an open problem in itself). Moreover, we run CARLA benchmark [50], which includes dynamic obstacles (vehicles and pedestrians) and generalization conditions (unseen town and weather). We show that CIL with the early fusion scheme produces state-of-the-art results.

2.3 Multimodal Fusion

We first detail CIL [39], and then show how we adapt this model to leverage multimodal perception data.

2.3.1 Base CIL Architecture

Figure 2.1 shows the CNN implementing CIL. The observations (the input of CIL) are twofold, perception data, $\mathbf{p}$, and vehicle's state measurements, $\mathbf{m}$. The action, $\mathbf{a}$ (the output of CIL), consists of vehicle controls for maneuvering. CIL includes a CNN block to extract perception features, $P(\mathbf{p})$, and a block of fully connected layers to extract measurement features $M(\mathbf{m})$. A joint layer of features is formed by concatenating $P(\mathbf{p})$ and $M(\mathbf{m})$, which is further processed by a new fully connected layer to obtain the joint features $J(<P(\mathbf{p}), M(\mathbf{m})>)$, which is simplified as $J(\mathbf{p}, \mathbf{m})$. So far, the processing of observations by the neural network is common to any driving maneuver/action. However, many times, the autonomous vehicle reaches ambiguous situations which require the incorporation of informed decisions. For instance, when reaching a cross intersection, without incorporating a route navigation command (*e.g.*, from a global trajectory plan), the vehicle can only take a random decision about turning or going straight. Thus, the end-to-end driving CNN must incorporate high-level commands, c, such as *'in the next intersection turn left'*, *'turn right'*, or *'go straight'*. Moreover, $\mathbf{a}$ will take very different values depending on c. Thus, provided c takes discrete values, having specialized neural network layers for each maneuver can be more accurate. All this is achieved in the CIL proposal by incorporating fully connected maneuver/action branches, A^c, selected by c (both during CNN training and the actual driving test).

We follow the CIL architecture proposed in [39]. Therefore, $\mathbf{p}$ is an RGB image of 200×88 pixels and 8 bits at each color channel, $\mathbf{m}$ is a real value with the current speed of the vehicle, and $\mathbf{a}$ consists of three real-valued signals which set the next maneuver in terms of steering angle, throttle, and brake. The idea is to perform vision-based end-to-end autonomous driving, as well as taking into account the vehicle speed to apply higher/lower throttle and brake for the same perceived traffic situation. In [39], the focus is on handling intersections, thus the considered c values are {*turn-left, turn-right, go-straight, continue*}, where the last refers to just keep driving in the current lane and the others inform about what to do when reaching the next intersection. Accordingly, there are four branches A^c. If we term by F the end-to-end driver, then we have $F(\mathbf{p}, \mathbf{m}, c) = A^c(J(\mathbf{p}, \mathbf{m}))$. We detail the parameters of CIL in Table 2.1, which has the same architecture as the multimodal model we term early fusion in this work.

Module	Input Dimension	Output Channels	Num. of Kernels	Stride	Dropout
Perception	$200 \times 88 \times M$[*]	32	5	2	0.0
	$98 \times 48 \times 32$	32	3	1	0.0
	$96 \times 46 \times 32$	64	3	2	0.0
	$47 \times 22 \times 64$	64	3	1	0.0
	$45 \times 20 \times 64$	128	3	2	0.0
	$22 \times 9 \times 128$	128	3	1	0.0
	$20 \times 7 \times 128$	256	3	2	0.0
	$9 \times 3 \times 256$	256	3	1	0.0
	$7 \times 1 \times 256$	512	-	-	0.0
	512	512	-	-	0.0
Measurement (speed)	1	128	-	-	0.0
	128	128	-	-	0.0
Join	512+128	512	-	-	0.3
Action Branch	512	256	-	-	0.5
	256	256	-	-	0.5
	256	3	-	-	0.0
Speed Branch	512	256	-	-	0.5
	256	256	-	-	0.5
	256	1	-	-	0.0

Table 2.1: The parameters of original CIL network. Notice that the multimodal model with the early fusion scheme has the same architecture as CIL, the only difference being the input dimension.

2.3.2 Fusion Schemes

Figure 2.2 illustrates how we fuse RGB and depth information following early, mid, and late fusion schemes. In addition to the parameters of the original CIL network that we provided in Table 2.1 (which is the same as the multimodal model following the early fusion scheme), we also detail the parameters of the other two schemes, mid fusion and late fusion in Table 2.2 and Table 2.3, respectively.

[*]Depending on single- or multimodal input, dimension M could be either 3 (RGB only), 1 (Depth only) or 4 (RGBD)

Module	Input Dimension	Output Channels	Num. of Kernels	Stride	Dropout
Perception (RGB)	200 × 88 × 3	32	5	2	0.0
	98 × 48 × 32	32	3	1	0.0
	96 × 46 × 32	64	3	2	0.0
	47 × 22 × 64	64	3	1	0.0
	45 × 20 × 64	128	3	2	0.0
	22 × 9 × 128	128	3	1	0.0
	20 × 7 × 128	256	3	2	0.0
	9 × 3 × 256	256	3	1	0.0
	7 × 1 × 256	512	-	-	0.0
	512	512	-	-	0.0
Perception (Depth)	200 × 88 × 1	32	5	2	0.0
	98 × 48 × 32	32	3	1	0.0
	96 × 46 × 32	64	3	2	0.0
	47 × 22 × 64	64	3	1	0.0
	45 × 20 × 64	128	3	2	0.0
	22 × 9 × 128	128	3	1	0.0
	20 × 7 × 128	256	3	2	0.0
	9 × 3 × 256	256	3	1	0.0
	7 × 1 × 256	512	-	-	0.0
	512	512	-	-	0.0
Measurement	1	128	-	-	0.0
	128	128	-	-	0.0
Join	512+512+128	512	-	-	0.3
Action Branch	512	256	-	-	0.5
	256	256	-	-	0.5
	256	3	-	-	0.0
Speed Branch	512	256	-	-	0.5
	256	256	-	-	0.5
	256	1	-	-	0.0

Table 2.2: The parameters of multimodal CIL network following mid fusion scheme.

Early Fusion. With respect to the original CIL, we only change the number of channels of $\mathbf{p}$ from three (RGB) to four (RGBD). The CIL network only changes the first convolutional layer of $P(\mathbf{p})$ to accommodate for the extra input channel, the rest of the network is equal to the original.

Mid Fusion. We replicate twice the perception processing $P(\mathbf{p})$. One of the $P(\mathbf{p})$ blocks processes only RGB images, and the other one only has depth maps. Then, we build the joint feature vector $<P(\text{RGB}), P(\text{D}), M(\mathbf{m})>$ which is further processed to obtain $J(\text{RGB}, \text{D}, \mathbf{m})$. From this point, the branched part of CIL is the same as in the original architecture.

Late Fusion. We replicate twice the full CIL architecture. Thus, RGB and depth channels are processed separately, but the speed measurement is shared as input. Hence, we run $A^c(J(\text{RGB}, \mathbf{m}))$ and $A^c(J(\text{D}, \mathbf{m}))$, and their outputs are concatenated and further processed by a module of fully connected layers, the output of which conveys the final action values. Note that this is a kind of mixture-of-experts approach, where the two experts are jointly trained.

As is common practice in the literature, we assume a pixel-level correspondence of all channels and normalize all of them to be in the same magnitude range (we normalize depth values to match the range of color channels, *i.e.*, from 0 to 255).

2.3.3 Loss Function

Given a predicted action $\mathbf{a}$, its ground truth $\mathbf{a}^{gt}$, and a vector of weights $\mathbf{w}$, we use the L1 loss $\ell_{act}(\mathbf{a}, \mathbf{a}^{gt}, \mathbf{w}) = \sum_i^n |w_i(a_i - a_i^{gt})|$, with $n = 3$ (steering angle, throttle, brake). Note that when computing $\mathbf{a}$, only one A^c branch is active at a time. In particular, the one selected by the particular command c that is associated with the current input data $(\mathbf{p}, \mathbf{m})$. We make this fact explicit by changing the notation to $\ell_{act}(\mathbf{a}, \mathbf{a}^{gt}, \mathbf{w}; c)$.

In addition, as in other computer vision problems addressed by deep learning [48, 89], we empirically found that using multi-task learning helps to obtain more accurate CIL networks. In particular, we add an additional branch of three fully connected layers to predict current vehicle speed from the perception data features $P(\mathbf{p})$. This prediction relies on an L1 loss $\ell_{sp}(s, s^{gt}) = |s - s^{gt}|$, where s is the predicted speed and s^{gt} is the ground truth speed which, in this case, is already available since it corresponds to the measurement used as input. Speed prediction is only used during training.

Thus, all networks, *i.e.*, both single- and multimodal, are trained according to the same total loss:

$$\ell(\mathbf{a}, \mathbf{a}^{gt}, \mathbf{w}; c; s, s^{gt}) = \beta \ell_{act}(\mathbf{a}, \mathbf{a}^{gt}, \mathbf{w}; c) + (1 - \beta)\ell_{sp}(s, s^{gt}), \tag{2.1}$$

where β is used to balance the relevance of ℓ_{act} and ℓ_{sp} losses.

†Depending on RGB or Depth input, dimension N could be either 3 (RGB) or 1 (Depth)

Module	Input Dimension	Output Channels	Num. of Kernels	Stride	Dropout
Perception (RGB/Depth)	200 × 88 × N[†]	32	5	2	0.0
	98 × 48 × 32	32	3	1	0.0
	96 × 46 × 32	64	3	2	0.0
	47 × 22 × 64	64	3	1	0.0
	45 × 20 × 64	128	3	2	0.0
	22 × 9 × 128	128	3	1	0.0
	20 × 7 × 128	256	3	2	0.0
	9 × 3 × 256	256	3	1	0.0
	7 × 1 × 256	512	-	-	0.0
	512	512	-	-	0.0
	512	512	-	-	0.0
Measurement (RGB/Depth)	1	128	-	-	0.0
	128	128	-	-	0.0
Join (RGB/Depth)	512+128	512	-	-	0.3
Action Branch (RGB/Depth)	512	256	-	-	0.5
	256	256	-	-	0.5
	256	3	-	-	0.0
Join (Streams)	3+3	256	-	-	0.0
Final Action	256	128	-	-	0.0
	128	128	-	-	0.0
	128	3	-	-	0.0
Speed Branch (RGB/Depth)	512	256	-	-	0.5
	256	256	-	-	0.5
	256	1	-	-	0.0
Join (Speeds)	1+1	256	-	-	0.0
Final Speed	256	128	-	-	0.0
	128	128	-	-	0.0
	128	1	-	-	0.0

Table 2.3: The parameters of multimodal CIL network following late fusion scheme.

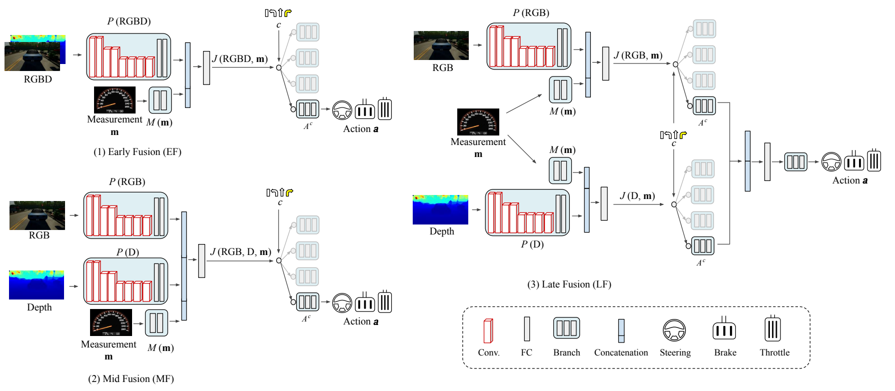


Figure 2.2: Network Architectures - we explore RGBD from the perspective of early, mid, and late fusion of the RGB and Depth (D) modalities. (1) Early Fusion: the raw RGB and D channels are concatenated as one input for the CIL architecture; (2) Mid Fusion: intermediate CIL feature layers from RGB and D streams are fused; (3) Late Fusion: the output (maneuver controls) of the RGB and D CIL streams are fused to output the final values after further neural processing.

2.4 Experiments

We start by summarizing the environment we use for our experiments, *i.e.*, CARLA simulator in Section 2.4.1. Next, we describe the dataset we use for training our AI drivers in Section 2.4.2, the training protocol that we follow in Section 2.4.3, and the driving benchmark available in CARLA in Section 2.4.4. Finally, we present and discuss the obtained results in Section 2.4.5.

2.4.1 Environment

In order to conduct our experiments, we rely on the open-source driving simulator CARLA [50]. There are several reasons: 1) many previous works on end-to-end driving rely on CARLA [39, 104, 105, 118, 141, 150, 178], thus it facilitates the comparison between our results and the previous literature; 2) developing AVs requires a lot of supports from humans and material resources. In particular, for some dangerous scenarios, directly collecting data or conducting experiments by human drivers in the physical world are very risky, which encourages research that relies on simulators for preliminary assessments; 3) it is demonstrated in [38] that current offline

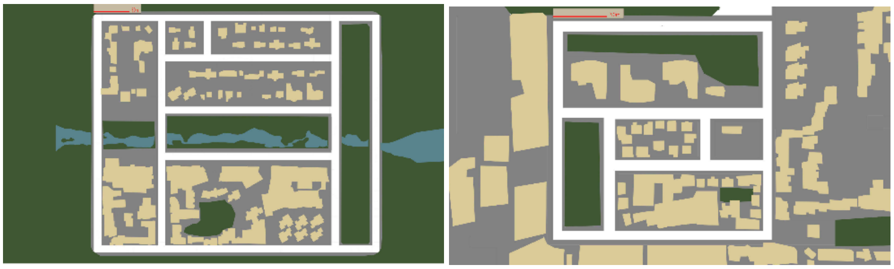


Figure 2.3: Bird-eye View road maps of Town 1 (left) and Town 2 (right).

	Training (dataset)	Validation (episodes)	Testing (episodes)
Wet cloudy noon	▨	▨	Towns 1 & 2
Soft rainy sunset	▨	Towns 1 & 2	
Clear noon	Town 1		
Clear after rain		▨	
Clear sunset		▨	
Heavy rain noon		▨	

Table 2.4: Training, validation, and testing settings. Training is based on a pre-recorded dataset. Validation and testing are based on actual driving episodes. Grey means 'not used'.

evaluation metrics (*i.e.*, based on static datasets) for assessing end-to-end driving models do not correlate well with actual driving, which is also observed in [17]. With the above considerations, it is necessary to perform driving models in an onboard driving regime, which is possible in a realistic simulator such as CARLA.

Briefly, CARLA (version 0.8.6) contains two towns, called Town 1 and Town 2 (Figure 2.3), which are based on two-directional roads (single-lane) with turns and intersections, buildings, vegetation, urban furniture, traffic signs, traffic lights, and dynamic objects such as vehicles and pedestrians. Town 1 deploys 2.9 km of road and 12 intersections, while Town 2 contains 1.4 km of road and 8 intersections. The different towns are set to be traveled under six different weather conditions (Figure 2.4): 'clear noon', 'clear after rain', 'heavy rain noon', and 'clear sunset', 'wet cloudy noon' and 'soft rainy sunset'.

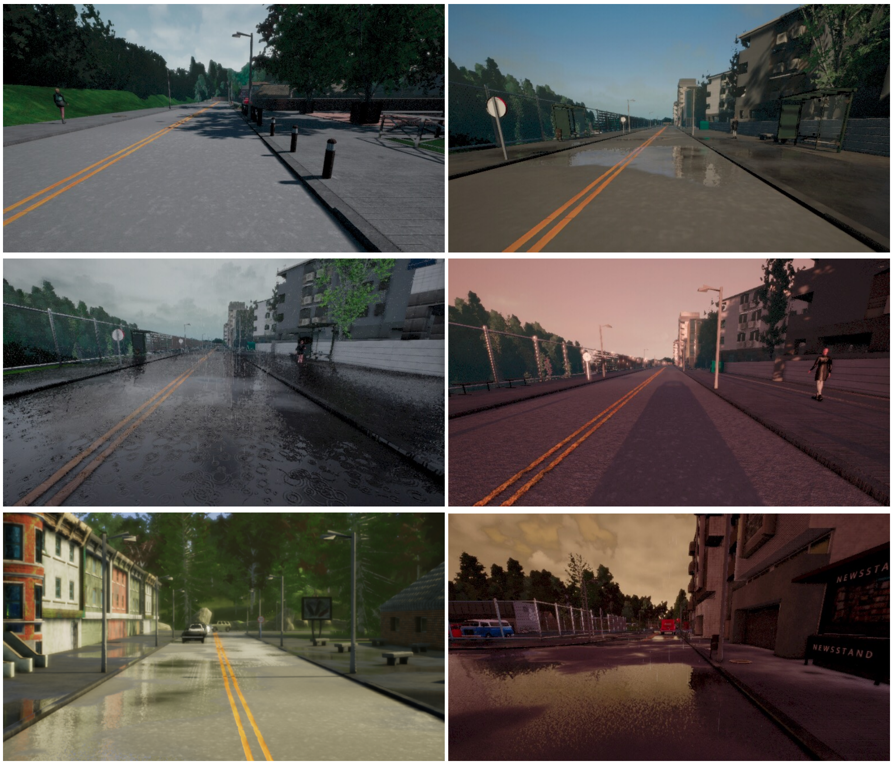


Figure 2.4: Top, from Town 1: clear noon (left) and clear after rain (right). Middle, from Town 1: heavy rain noon (left) and clear sunset (right). Bottom, from Town 2: wet cloudy noon (left) and soft rainy sunset (right).

2.4.2 Training Dataset

In order to train our CNNs, we use the same dataset as in [38] which corresponds to 25 hours of driving in Town 1, balancing weather conditions (Table 2.4). Briefly, this dataset was collected by a hard-coded autopilot with access to all the privileged information of CARLA required for driving like an expert. The autopilot kept a constant speed of 35 km/h when driving straight and reduced the speed when making turns. Images were recorded at 20fps from three cameras: a central forward-facing one and two lateral cameras facing 30° left and right. The central camera is the only one used for the actual driving test, while the images coming from the lateral cameras are used during data collection and only at training time to simulate episodes of recovering from driving errors as can be done with real cars [20] (the protocol for injecting noise follows [39]). Overall, the dataset contains ∼ 2.5 millions of RGB images of 800×600 pixels resolution, with associated ground truth (see Figure 2.5) consisting of corresponding images of dense depth and pixel-wise semantic classes (semantic segmentation), as well as meta-information consists of the high-level commands provided by the navigation system (continue in the lane, at next intersection go straight/turn left/turn right), and the state measurements of the vehicle such as speed, steering angle, throttle, and brake. In this work, we use perfect semantic segmentation to develop an upper-bound driver. Since we focus on end-to-end driving, the twelve semantic classes of CARLA are mapped to five which we consider sufficient to develop such an upper-bound. In particular, we keep the original *road-surface*, *vehicle*, and *pedestrian*, while *lane-marking* and *sidewalk* are mapped as *lane-limits* (Town 1 and Town 2 only include two-directional (single-lane) roads, separated by double solid lines), and the remaining seven classes are mapped as *other*.

Focusing on depth information, as is common in the literature, we assume that RGB images have associated dense depth information; for instance, Premebida *et al.* [134] obtained depth information from LiDAR point clouds. In CARLA, the depth ground truth is extremely accurate since it comes directly from the Z-buffer used during simulation rendering. In particular, depth values run from 0 to 1,000 meters and are codified with 24 bits, which means that depth precision is of ∼ 1/20 mm. This distance range coverage and depth precision are far beyond what even *active* sensors can provide. Therefore, we post-process depth data to make it more realistic. In particular, we take as a realistic sensor reference the Velodyne information of the KITTI dataset [60]. First, we trim depth values to consider only those within the 1 to 100 meters interval, *i.e.*, pixels of the depth image with values outside this range are considered as not having depth information. Second, we re-quantify the depth values to have an accuracy of ∼ 4 cm. Third, we perform inpainting to fill in the pixels with no information. Finally, we apply a median filter to avoid having perfect

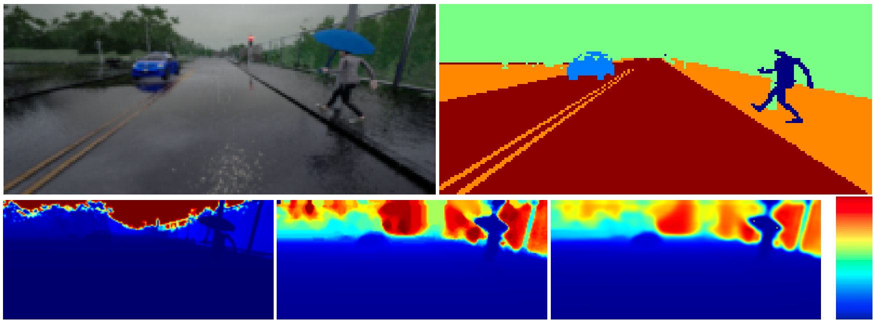


Figure 2.5: Top: original RGB image and semantic segmentation ground truth (for the five considered classes); Bottom from left to right: CARLA depth ground truth, post-processed to be closer to the capabilities of an *active* depth Sensor, and monocular depth estimation from a model trained using such a depth.

depth boundaries between objects. The new depth images are used both during training and testing. Figure 2.5 shows an example of a depth image from CARLA and its corresponding post-processed version.

2.4.3 Training Protocol

All CIL models in this work rely on the same training protocol, partially following [39]. In all CIL models, original sensor channels (R/G/B/D) are trimmed to remove the sky and very close areas (top and bottom part of the channels), and down-scaled to finally obtain channels of 200×88 pixel resolution. In our initial experiments, we found that traditional photometric and geometric recipes for data augmentation were not providing better driving models, thus, we do not use them. Dropout is not used in convolutional layers, while it is used in some fully connected layers as detailed in Table 2.1, Table 2.2 and Table 2.3.

During the training, we use the Adam optimizer with 120 training samples per iteration (minibatch), an initial learning rate of 0.0002, which decreased to half each 50K iterations. Minibatches are balanced in terms of per A^c branch samples. We set $\mathbf{w} = (0.5, 0.45, 0.05)$ to weight the control signals (action) in the loss function. Action and speed losses are balanced by $\beta = 0.95$. For selecting the best intermediate model of a training run, we do 500K iterations monitoring a validation performance measurement, V_P, each 100K iterations (thus, five times). The intermediate model with the highest V_P is selected as the resulting model of the training run. Since CIL models are trained from scratch, variability is expected in their performance. Thus, for each type of model, we perform five training runs, finally selecting the model with the highest V_P.

Using Table 2.4 as a reference, we define V_P to balance training-validation differences in terms of town and weather conditions. In particular, we use $V_P = 0.25V_w + 0.25V_t + 0.50V_{wt}$; where V_w is the success rate when validating in Town 1 and 'soft rainy sunset' weather (not included in training data), V_t is a success rate when validating in Town 2 (not included in training data) and 'clear noon' weather (included in training data), and V_{wt} stands for success rate when validating in Town 2 and 'soft rainy sunset' (neither town nor weather is part of the training data). Therefore, note that V_P is a weighted success rate based on 75 episodes.

2.4.4 Driving Benchmark

CARLA was deployed as a benchmark with infrastructures for assessing the performance of AI drivers [50], which has been widely used in the related literature. For comparison, we test our models on the same benchmark. Four *driving tasks* of increasing difficulty are defined:

- *straight*: the destination point is straight ahead from the starting point, with no dynamic objects;
- *one turn*: the destination is one turn away from the starting point, with no dynamic objects;
- *navigation*: the route of the episodes consists of more turns, with no dynamic objects;
- *navigation with dynamic obstacles*: the route of the episodes consists of more turns, as well as dynamic objects.

For each driving task, an AI driver is assessed over a total of E_T driving episodes. Each episode has different starting and destination points with an associated topological route. An episode is considered as successful if the AI driver completes the route within a time budget. Collisions do not lead to the termination of an episode unless the AV runs in time-out as a consequence. If we term as E_S the total number of successfully completed episodes by the assessed AI driver, then its *success rate* is defined as $100 \times (E_S/E_T)$. E_T is determined by the selected town and weather conditions.

Table 2.4 shows how the benchmark organizes towns and weather conditions for training, validation, and testing. Irrespective of the town and weather, validation and testing are always based on episodes, but not in pre-recorded datasets, while training requires pre-recording a dataset. Validation is performed to select a driving

	RGB	Active				Estimation	
		D	EF	MF	LF	D	EF
1	**48**	**74**	**91**	61	60	51	42
2	36	67	71	71	63	49	44
3	46	73	75	58	**67**	46	**51**
4	40	68	71	**74**	60	**59**	46
5	36	68	77	52	62	51	49

Table 2.5: V_P for five training runs for RGB only, Depth (D) only, and RGBD combined by early (EF), mid (MF), or late (LF) fusion. Depth: from an active sensor or estimated from RGB images.

model among those trained as different trials from the same training dataset, while testing is performed by actually running the benchmark for the selected models.

Regarding town and weather conditions, the benchmark establishes four main town-weather blocks under which the four driving tasks need to be tested, assuming 25 episodes for each considered weather. Therefore, for each block, the E_T value is different as we can deduce from Table 2.4. In particular, these are the town-weather blocks defined in the benchmark with their respective E_T value:

- *Training conditions.*: driving (*i.e.*, running the episodes) in the same conditions as the training set (Town 1, four weather conditions), thus, $E_T = 100$;
- *New town*: driving under the four weather conditions of the training set but in Town 2, $E_T = 100$;
- *New weather*: driving in Town 1 but under the two weather conditions not seen at training time, $E_T = 50$;
- *New town & weather*: driving in conditions not included in the training set (Town 2, two weather conditions), $E_T = 50$.

2.4.5 Experimental Results

We start the analysis of the experimental results from Table 2.5, which is produced during training and selection of the best-trained CIL models. We focus first on RGB data, as well as depth based on the post-processed CARLA depth ground truth, termed here as *active* depth (Section 2.4.2) since its accuracy and covered depth range is characteristic of active sensors (*e.g.*, LiDAR). We see that the best (among five training runs) validation performance V_P is 48% when using RGB data only.

			Active				Estimated	
Task	SS	RGB	D	EF	MF	LF	D	EF
				Training Conditions				
Straight	98.00 ± 1.73	96.33 ± 1.53	98.67 ± 1.53	98.33 ± 0.58	92.33 ± 2.08	**99.00** ± 0.00	92.33 ± 1.15	97.33 ± 1.15
One turn	100.00 ± 0.00	95.00 ± 0.00	92.00 ± 0.00	**99.00** ± 0.00	91.67 ± 2.08	90.33 ± 0.58	84.67 ± 1.15	96.33 ± 1.53
Navigation	96.00 ± 0.00	89.00 ± 2.00	89.33 ± 2.08	92.67 ± 1.15	90.67 ± 1.15	93.67 ± 0.58	75.33 ± 1.15	**94.33** ± 0.58
Nav.Dynamic	92.00 ± 1.00	84.00 ± 2.00	82.67 ± 0.58	89.33 ± 0.58	78.33 ± 2.89	89.00 ± 2.65	71.00 ± 1.00	**89.67** ± 1.15
				New Weather				
Straight	100.00 ± 0.00	84.00 ± 0.00	**99.33** ± 1.15	96.00 ± 2.00	94.67 ± 3.06	96.00 ± 0.00	92.00 ± 2.00	84.67 ± 1.15
One turn	100.00 ± 0.00	76.67 ± 4.16	**94.67** ± 2.31	**94.67** ± 2.31	94.00 ± 2.00	92.00 ± 2.00	93.33 ± 2.31	80.67 ± 1.15
Navigation	95.33 ± 1.15	72.67 ± 2.31	89.33 ± 1.15	91.33 ± 2.31	90.67 ± 3.06	**96.00** ± 0.00	73.33 ± 2.31	80.67 ± 5.03
Nav.Dynamic	92.67 ± 1.15	68.67 ± 4.62	90.00 ± 2.00	86.00 ± 4.00	80.67 ± 3.06	**92.67** ± 3.06	76.67 ± 4.16	77.33 ± 6.11
				New Town				
Straight	100.00 ± 0.00	84.00 ± 2.00	94.33 ± 0.58	**96.33** ± 0.58	87.00 ± 1.00	77.00 ± 0.00	78.33 ± 1.53	71.67 ± 2.08
One turn	96.67 ± 0.58	68.00 ± 1.00	74.33 ± 2.52	**79.00** ± 1.73	78.00 ± 2.65	58.67 ± 2.08	46.33 ± 1.15	47.00 ± 1.00
Navigation	96.00 ± 0.00	59.67 ± 3.06	85.33 ± 1.15	**90.00** ± 2.00	80.67 ± 0.58	52.33 ± 0.58	45.67 ± 3.06	46.67 ± 3.06
Nav.Dynamic	99.33 ± 0.58	54.33 ± 3.79	70.33 ± 1.15	**84.33** ± 2.52	73.67 ± 2.52	55.67 ± 2.31	44.33 ± 2.52	46.67 ± 4.04
				New Town & Weather				
Straight	100.00 ± 0.00	84.67 ± 1.15	**97.33** ± 1.15	**97.33** ± 2.31	88.67 ± 1.15	**97.33** ± 1.15	78.00 ± 0.00	89.33 ± 1.15
One turn	96.00 ± 0.00	66.67 ± 4.62	72.67 ± 1.15	**82.67** ± 2.31	69.33 ± 3.06	67.33 ± 2.31	62.67 ± 1.15	64.00 ± 3.46
Navigation	96.00 ± 0.00	57.33 ± 6.11	84.00 ± 3.46	**92.67** ± 3.06	78.67 ± 3.06	72.67 ± 1.15	55.33 ± 6.11	60.67 ± 2.31
Nav.Dynamic	98.00 ± 2.00	46.67 ± 6.43	69.33 ± 2.31	**94.00** ± 0.00	73.33 ± 3.06	73.33 ± 2.31	54.00 ± 4.00	49.33 ± 3.06

Table 2.6: Mean and standard deviation of success rates on the original CARLA Benchmark, by running it three times. CIL based on perfect semantic segmentation (SS) acts as an upper bound. Excluding SS, for models tested under the same environment and traffic conditions, we show in bold the higher means and underline similar success rates considering standard deviations as well.

Thus this corresponding CIL model is used as an RGB-based driver in the following experiments. Analogously, for the case of using only active depth (D), the best CIL reports performance of 74%. The best performances for early fusion (EF), mid fusion (MF), and late fusion (LF) are 91%, 74%, and 67%, respectively. Analogously, the corresponding CIL models are taken as drivers for the following experiments.

Table 2.6 reports the performance of the selected models according to the original CARLA benchmark. We have included a model trained on perfect semantic segmentation (SS) according to the five classes considered for autonomous driving (see Figure 2.5). This model is considered the upper-bound model. Indeed, its performance is most of the time ≥ 96, reaching 100 several times. This also confirms that provided there is a proper input, the CIL model is able to drive properly in CARLA conditions. We can see that only active depth is already powerful information for end-to-end driving, clearly outperforming RGB in non-training conditions. However, in most cases, RGBD outperforms the use of only RGB or only D. The clearest case is for new town and weather with dynamic objects, *i.e.*, for the most challenging conditions, where using only RGB as input reaches a success rate of

Task	MP	RL	CAL	CIRL	MT	Active EF	MP	RL	CAL	CIRL	MT	Active EF
			Training Conditions						New Town			
Straight	98	89	**100**	98	98	98.33 ± 0.58	92	74	93	**100**	**100**	96.33 ± 0.58
One turn	82	34	97	97	87	**99.00** ± 0.00	61	12	**82**	71	81	79.00 ± 1.73
Navigation	80	14	92	**93**	81	92.67 ± 1.15	24	3	70	53	72	**90.00** ± 2.00
Nav.dynamic	77	7	83	82	81	**89.33** ± 0.58	24	2	64	41	53	**84.33** ± 2.52
			New Weather						New Town & Weather			
Straight	**100**	86	**100**	**100**	**100**	96.00 ± 2.00	50	68	94	**98**	96	97.33 ± 2.31
One turn	95	16	**96**	94	88	94.67 ± 2.31	50	20	72	82	82	**82.67** ± 2.31
Navigation	**94**	2	90	86	88	91.33 ± 2.31	47	6	68	68	78	**92.67** ± 3.06
Nav.dynamic	**89**	2	82	80	80	86.00 ± 4.00	44	4	64	62	62	**94.00** ± 0.00

Table 2.7: Success rate comparison with previous methods (see main text).

Km per	Event	RGB	Active D	Active EF
Infraction	Sidewalk	0.86 ± 0.10	35.80 ± 1.30	16.76 ± 5.54
	Opposite lane	0.73 ± 0.04	1.65 ± 0.24	3.29 ± 1.96
Driven Km	(Perfect driving: 17.30 Km)	13.62 ± 0.67	35.80 ± 1.30	20.22 ± 0.54

Table 2.8: Infractions on dynamic navigation in new town & weather.

46.67 ± 6.43, and using only D as input reaches 69.33 ± 2.31, but the multimodal one following early fusion scheme achieves the success rate of 94.00 ± 0.00. For a new town (irrespective of the weather conditions) early fusion clearly outperforms mid and late fusion. In any case, it is clear that multimodality improves CIL performance with respect to a single modality, which is the main question we want to answer in this work.

In order to further analyze the goodness of multimodality, we compare it to previous single-modality methods (see Section 2.2). Not all the corresponding papers provide details about the training methodology or training datasets; thus, this comparison is solely based on the reported performances on the original CARLA benchmark and can only be taken as an additional reference about the goodness of multimodality. Consider that early fusion is the smaller CNN architecture in terms of parameters, thus, we take this model for comparison. Table 2.7 shows the results. MP and RL stand for modular perception and reinforcement learning, respectively. The reported results are reproduced from [50]. CAL stands for conditional affordance learning and the results are reproduced from [150]. CIRL stands for controllable imitative reinforcement learning and the results are reproduced from [105]. Finally, MT stands for multi-task learning, and the results are reproduced from [104]. We see how, in the presence of dynamic traffic participants, the

RGBD early fusion (with active depth) is the model with a higher success rate on the original CARLA benchmark. On the other hand, such an early fusion approach can be combined with CAL or CIRL methods, since they are totally compatible. We think that this comparison with previous works reinforces the idea that multimodality can help end-to-end driving.

Once it is clear that multimodality is beneficial for end-to-end driving, in this context we can raise the question of *Whether monocular depth estimation [64, 70, 112, 187] can be as effective as depth coming from active sensors?* In the former case, it corresponds to a multisensory multimodal approach, while the latter case corresponds to a single-sensor multimodal approach in which both RGB and depth come from the same camera sensor (depth is estimated from RGB). In order to carry out a proof-of-concept, we use a monocular depth estimation model [70] (which was a SOTA model at the moment of its publication) fine-tuned on the CARLA training dataset. More specifically, the dataset used for training the multimodal CIL models is also used to fine-tune our monocular depth estimation model, *i.e.*, using the post-processed depth channels and corresponding RGB images. Figure 2.5 shows an example of monocular depth estimation.

Analogously to the experiments shown so far, we train a CIL model based on the estimated depth as well as on the corresponding multimodal (RGBD) fusion. In order to reduce the burden of experiments, we use early fusion since it achieves the best performance for the active depth case. The training performances for model selection can be seen in Table 2.5. We use the CIL models of V_P 59% and 51%, respectively. In validation terms, such performances are already clearly worse than the analogous based on active depth. Table 2.6 shows the results of the original CARLA benchmark. Indeed, these are worse than using active depth, however, when remaining in the training conditions, monocular-based EF outperforms depth and RGB alone, and in fact, shows similar performance as active depth. This is not the case when we change from training conditions to the others, since monocular depth estimation itself does not perform equally well in this case, and so happens to EF. However, we think that this single-sensor multimodal setting is worth pursuing. Moreover, although it is out of the scope of this work, we think that performing end-to-end driving may be a good protocol for evaluating depth estimation models beyond the static metrics currently used, which are agnostic to the task in which depth estimation is going to be used. Note that even for evaluating the driving performance of end-to-end driving models, it has been shown that relying only on static evaluations may be misleading [17, 38].

Finally, for the RGB, Active D, and EF models, we assess additional infractions for the new town and weather with dynamic objects. Table 2.8 shows the driven Km per infraction of each model. Note that not all such infractions imply an accident stopping the AV. For instance, the AV can run into an opposite lane a bit without crashing with other vehicles. As a reference, we also show the amount of driven Km on which these measurements are based. All models are supposed to complete the same testing routes (*i.e.*, same total Km), termed as *perfect driving* in Table 2.8. However, if a model fails to follow the right path at an intersection the route would be recomputed, thus, it needs to drive more Km to reach the destination. On the contrary, if it fails to complete the routes, the driven Km will be lower. We see that RGB and Active EF models are not far, but Active D fails very much at taking the right path at intersections. RGB performs the worst in all metrics. The Active D model does not run over the sidewalk and uses the curbside as a cue, which also helps with lane keeping except at intersections. Active EF shows a good equilibrium between RGB and Active depth single-modality models.

2.5 Conclusion

In this chapter, we compare single- and multimodal perception data for end-to-end driving. As for multimodal perception data, we focus on RGB and depth, since they are usually available in autonomous vehicles through the presence of cameras and active sensors such as LiDAR. As for the end-to-end driving model, we use branched conditional imitation learning (CIL). Relying on a well-established simulation environment, CARLA, we assess the driving performance of single-modal (RGB, depth) CIL models, as well as multimodal CIL models according to early, mid, and late fusion schemes. In all cases, the depth information available in CARLA is post-processed to obtain a more realistic range of distances and depth accuracy. This depth is also used to train a depth estimation model so that the experiments cover multimodality not only based on a multisensory setting (RGB and active depth) but also based on a single-sensor setting (RGB and estimated depth). Overall, the experiments clearly lead us to conclude that multimodality (RGBD) is indeed a beneficial approach for end-to-end driving. In the future, we plan to follow this line of work, considering other sources of multi-modality usually available in modern vehicles, such as GNSS information, which even though usually noisy, eventually can complement direct scene sensing.

3 Action-based Representation Learning for Autonomous Driving

3.1 Introduction

The development of autonomous vehicles (AVs) is a significant multidisciplinary challenge. Currently, the main paradigm being pursued in developing AVs follows a traditional divide-&-conquer engineering strategy. In particular, modular pipelines are proposed with key modules for perception, route planning, and maneuver control, among others [197]. In turn, these modules may be composed to deal with different tasks, *e.g.*, perception encompasses object detection and tracking, semantic class/instance segmentation, *etc.* [85]. These tasks rely on models trained from data using modern deep learning techniques [68]. Following such a data-driven approach is not a problem in itself since it is possible to collect petabytes of onboard data (raw sensor data, vehicle state variables, *etc*). Continuously, not only from fleets of AVs under development but also from sensorized human-driven vehicles under naturalistic driving. However, in practice, the best-performing models arise from supervised deep learning, and this means that the raw data must be augmented with ground truth, which is collected through time-consuming and costly human labeling (*e.g.*, bounding boxes, object silhouettes, *etc*).

The data labeling bottleneck associated with these approaches has caused the idea of end-to-end driving [100,131] to receive renewed interest [3,20,39,40,77,188]. In this paradigm a deep model is trained to directly control an AV from input raw sensor data (mainly images), *i.e.*, without a clear separation between perception and maneuver planning, and without explicit intermediate perceptual tasks to be solved. In this pure data-centered approach, the supervision required to train deep end-to-end driving models does not come from human labeling; instead, the vehicle's state variables, which can be automatically collected from fleets of human-driven vehicles, are used as supervision (*e.g.*, speed, steering angle, acceleration, braking). These models are mainly trained by behavior cloning (BC) of human driving experiences. However, despite the undeniable good performance shown by end-to-end driving models, their reliability is controversial, due in particular to the difficulty of interpreting the relationship between inferred driving actions and image content [44], as well as training instabilities [40].

A different paradigm, conceptually midway between pure modular and end-to-end driving, is the so-called direct perception approach [26, 150], which focuses on learning deep models to predict driving affordances, from which an additional controller can maneuver the AV. In general, such affordances can be understood as a relatively small set of interpretable variables describing events that are relevant for an agent acting in an environment [62]. Driving affordances bring interpretability while only requiring weak supervision, in particular, human labeling just at the image level (*i.e.*, not pixel-wise).

In this chapter , we show that action-based methods, that focus on predicting the control actions, such as end-to-end driving trained with BC, can be an effective pre-training strategy for learning a direct perception model (Figure 3.2). This strategy enables a significant reduction in the number of labeled images required to train such a model. Overall, this means that we can leverage the data collected by fleets of human-driven vehicles for training interpretable driving models, thus, keeping a major advantage of modular pipelines while reducing data supervision (*i.e.*, human labeling). Further, our approach improves over other recent pre-training proposals such as contrastive methods [4] and even over ImageNet (supervised) pre-training. We also show that learning from expert data in our approach leads to better representations compared to training inverse dynamics models using the approach in [1]. This shows that expert driving data (*i.e.*, coming from human drivers) is an important source for representation learning. As is a common practice nowadays, we run our experiments in the CARLA simulator [50]. To the best of our knowledge, we are the first to show that expert demonstrations can act as an effective action-based representation learning technique. This constitutes the primary contribution of this work.

We present the work as follows. First, we review the related work in Section 3.2. Then, we present our main method of learning representation using end-to-end models with action-based supervision in Section 3.3. In Section 3.4, we summarize the experimental setting and the obtained results. Finally, we draw the main conclusions and future work in Section 3.5.

3.2 Related Work

Since human-based data labeling is a general problem for all kinds of new data-intensive applications, not only for autonomous driving, learning representations for deep models with the support of weak supervision and self-supervision are open challenges that attract great interest.

In the autonomous driving context, the use of driving affordances [26, 150] allows for weak supervision since only human labeling at the image level is required. Based on these interpretable affordances a controller is tuned to drive. Since [150] focuses on urban driving using the CARLA simulator, inspired by this work, we have defined four affordances to consider the explicit detection of hazards involving pedestrians and vehicles, respecting traffic lights, and considering the heading of the vehicle within the current lane. Defining the best set of affordances to drive is not the focus of this work , but we have chosen a reasonable set.

In order to solve visual tasks, we can find self-supervision based on auxiliary and relatively simple (pretext) tasks such as learning colorization [99], rotations [56, 63, 189], shuffling cues [114], or solving a jigsaw puzzle of image parts [123]; it has been shown that self-supervision can match traditional ImageNet (supervised) pre-training provided one works with large enough CNNs [95], although it has been argued that these proxy tasks are not sufficiently hard so as to fully exploit large unsupervised datasets [67]. Another branch of self-supervised learning is based on contrastive methods [31, 72, 124], which learn representations by comparing data pairs. While in these methods, supervision is based on different ways of transforming or comparing the input data itself, in this work, supervision comes in the form of expert driver actions. In fact, we include in our study a recent contrastive method, ST-DIM, designed in the context of playing Atari games [4], adapted to actions required for driving. We will see how action-based representation learning outperforms ST-DIM as a representation learning strategy to infer affordances.

In fact, in a perceive-&-act context, dynamics learning [37, 73] and inverse dynamics [1, 127, 157] can be used as action-based supervision strategy. Broadly speaking, being able to predict the next states of an agent or the action between state transitions, yields useful representations. This action-centric approach to supervision is in line with our work. Thus, our study includes experiments with different inverse and forward dynamics supervision strategies. We show the importance of those strategies in the autonomous driving context. However, different than previous work, we empirically demonstrate that expert actions yield a better representation learning than random actions used in [1].

Finally, it is also worth mentioning teacher-student strategies [28, 200] which allow one to train an end-to-end driving student model from a teacher model. In this case, even if the student is end-to-end, the data labeling bottleneck arises during the supervised training of the teacher, which requires bounding boxes and/or semantic segmentation. Since the student is still an end-to-end driving model, the issue of interpretability once this model is deployed in the AV still would remain open.

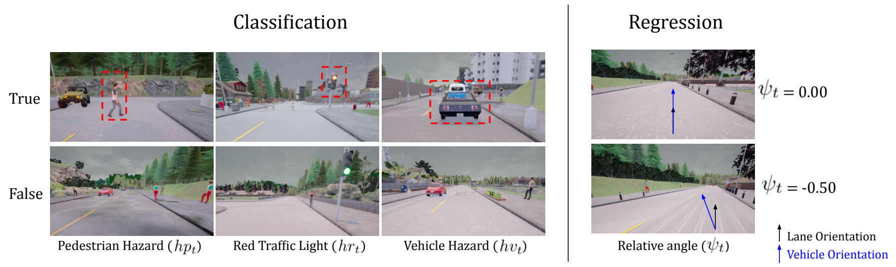
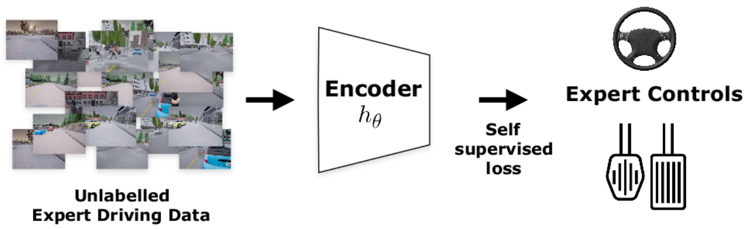
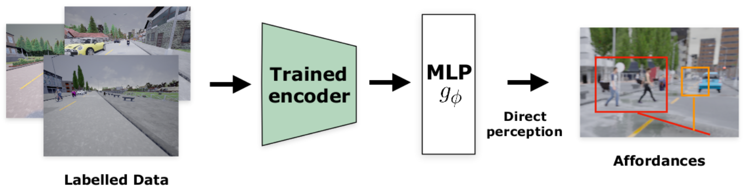


Figure 3.1: Our affordances illustrated on images from CARLA. Classification ones are binary variables (t/f), and regression runs on $[-\pi,\pi]$rad. See Section 3.3.3 for details.



(a) Action-based supervised training stage (end-to-end driving)



(b) Weakly supervised training stage (direct perception)

Figure 3.2: Approach overview: (a) an encoder is trained following an end-to-end driving setting (*e.g.*, using BC or inverse model); (b) this pre-trained encoder and a multi-layer perceptron (MLP) are used for predicting affordances. The affordances are used as input to a simple PID controller to drive the vehicle.

3.3 Action-based Representation Learning

3.3.1 Overall Approach

As can be seen in Figure 3.2, we study our action-based representation learning strategy by learning affordances in two stages. The first stage relies on non-manually labeled data to learn a representation (encoder). We will consider different methods to learn this representation (Section 3.3.2), all of them based on predicting driving actions from onboard data. The second stage uses this pre-trained representation together with a multi-layer perceptron (MLP) to learn the considered affordances (Section 3.3.3).

Therefore, for the first stage of our approach, we assume that we have access to a sequence of N_u data samples $\mathcal{D}^u = \{d_t\}_{t=1}^{N_u}$, which have been acquired on board a human-driven sensorized vehicle but without human labeling. Thus, we have $d_t = \{o_t, a_t\}$, where o_t and a_t are respectively, at a certain time t, the observation acquired by the vehicle's sensors and the driving action taken by the expert driver. We must understand that a_t is the expert reaction to the environment when o_t was acquired. In general, we will have different $\mathcal{D}^u$ sequences acquired at different driving runs, however, without losing generality and for the sake of keeping a simpler notation, we can assume all of them appended in one single sequence. In this work, we assume that each observation o_t contains an *image* capture of the driving environment, the *vehicle speed* (v_t) at the moment the image is acquired, and a high level *navigation command* such as `continue` in the same lane, or in the next intersection go `straight/left/right`, *i.e.*, as introduced in the so-called conditional imitation learning (CIL) [39] in the form of one-hot vector c_t. The corresponding action a_t is defined in terms of the *steering angle*, *acceleration*, and *break* values that must be applied to maneuver the vehicle.

For the second stage of our approach, we assume a relatively small dataset of on-board images with image-level affordance labeling (weak supervision), *i.e.*, $\mathcal{D}^l = \{x_j\}_{j=1}^{N_l}$, with $x_j = \{o_j, y_j\}$, being o_j the observation and y_j the corresponding affordance labeling. In particular, y_j contains variables indicating situations such as a *pedestrian hazard*, a *vehicle hazard*, a *red traffic light*, and a *relative heading angle* (Figure 3.1). In this setting, we can assume that the images used in $\mathcal{D}^l$ come from sub-sequences of $\mathcal{D}^u$, but are selected so that $N_u \ggg N_l$. Accordingly, the two stages can be summarized as follows: (1) use $\mathcal{D}^u$ to train a deep encoder h_θ; (2) use h_θ and $\mathcal{D}^l$ to train a projection network, g_ϕ, for predicting affordances. For actual driving, we develop a controller $C: g_\phi(h_\theta(o_t)) \rightarrow \hat{a}_t$; *i.e.*, given the affordances $g_\phi(h_\theta(o_t))$ predicted from an observation o_t, C estimates the action $\hat{a}_t$ to maneuver the vehicle. In order to show driving results, we will use a simple PID controller.

3.3.2 Action-based Supervised Stage

At the action-based supervised stage the objective is to train an encoder h_θ to produce a set of features z_t (encoder's bottleneck) given an input observation o_t. With this purpose, we have studied some alternatives illustrated in Figure 3.3. At training time, all of them rely on $\mathcal{D}^u$, but they use different inputs and losses to be

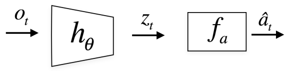
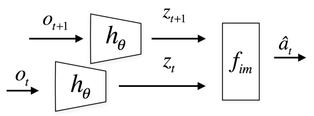
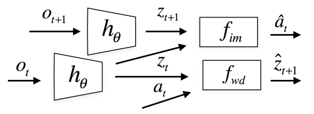
	Method	Architecture	Loss
(a)	Behavior Cloning (BC)		$s_{bc} = \|\|f_a(z_t) - a_t\|\|$
(b)	Inverse Model		$s_{im} = \|\|f_{im}(z_t, z_{t+1}) - a_t\|\|$
(c)	Forward Model		$s_{fm} = s_{im} + \|\|f_{wd}(z_t, a_t) - z_{t+1}\|\|$

Figure 3.3: Proposed action-based supervised losses based on expert actions. To train the encoder h_θ, these are minimized over the dataset $\mathscr{D}^u$.

minimized. We summarize these alternatives in the following.

Behavior cloning (BC). Common deep architectures trained by BC consist of an encoder h_θ extracting features z_t from observations o_t (*i.e.*, $z_t = h_\theta(o_t)$), and a fully-connected projection network, $f_a(z_t)$, which predicts an expert action $\hat{a}_t$ from such features. In this work , we follow the architecture presented in [40]; however, instead of using the high-level command (c_t) for branching to different projection functions, since our goal is to pre-train a useful representation not performing actual driving with it, we use c_t as part of o_t. Therefore, the encoder of [40] is modified to input the high-level command the same way as the speed variable (v_t). The input image is processed by a ResNet34 backbone. We detail the architecture of our BC encoder in Figure 3.4. The following alternatives also rely on this encoder architecture.

Inverse model. By considering not only o_t but also the subsequent observation o_{t+1} as input, we turn BC into an Inverse model [1, 157]. In this case, thinking of the encoder's bottleneck as encoding an agent (driver) internal state, the problem to solve consists of predicting the action that transforms the state z_t into the state z_{t+1}. Differently than BC, with an inverse model, the actions can come from either an expert driver or just random roaming (or poor driving).

Forward model. In this case, we want to learn an encoder that is able to output a state $z_t = h_\theta(o_t)$ such that, given an action a_t, we can predict the future state as

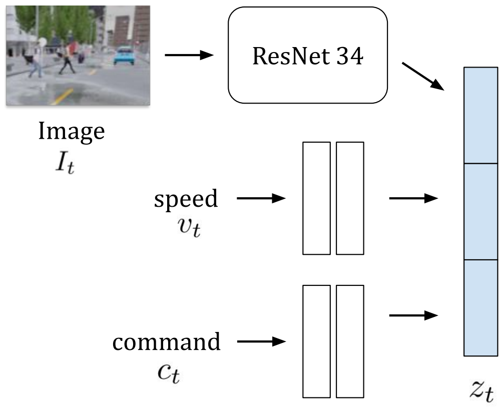


Figure 3.4: The architecture of the encoder h_θ. The blue rectangles indicate that the features are concatenated. The white rectangles indicate fully connected layers.

$z_{t+1} = f_{wd}(z_t, a_t)$. However, this can lead to the trivial solution $z_t = \mathbf{0}$ from which f_{wd} can still produce z_{t+1}. Such degenerated encoders h_θ are of course not of interest. Therefore, we use the regularization strategy of [1], consisting of adding also the Inverse model so that the encoded observation z_t is able to predict the action too. Again, the actions can come from either an expert driver or just random roaming.

3.3.3 Weakly Supervised Stage: Learning Affordances

The affordances used in this work (Figure 3.1) consider explicit detection of hazards involving pedestrians and vehicles, respecting traffic lights, and considering the heading of the vehicle within the current lane. More specifically, we have considered the following four affordances:

- **Pedestrian hazard (hp_t).** This variable is set to one if there is a pedestrian in our lane at a distance lower than 10 m; otherwise, is set to zero.

- **Vehicle hazard (hv_t).** This variable is set to one if there is a vehicle in our lane at a distance lower than 10 m; otherwise, is set to zero. Vehicles refer to cars, vans, motorbikes, and cyclists.

- **Red traffic light (hr_t).** This variable is set to one if there is a traffic light in red affecting our lane at a distance lower than 10 m; otherwise, is set to zero.

- **Relative heading angle (ψ_t).** This variable accounts for the relative angle of the longitudinal vehicle axis with respect to the lane in which it is navigating. The variable runs on $[-\pi, \pi]$ rad with $\psi_t = 0$ when the vehicle and lane are aligned (no matter the lateral vehicle position within the lane).

Note that $\{hp_t, hv_t, hr_t\}$ are binary variables, so predicting them is a binary classification problem, while ψ_t is a real number, thus, predicting it is a regression problem. These binary variables are critical to performing stop-&-go maneuvers by any controller relying on these affordances, while the regressed angle is critical to properly navigating without going out of the lane. Overall, the idea behind these affordances is that those relevant visual competencies for driving emerge when training the corresponding models; for instance, some kind of pedestrian and vehicle detection, red traffic light detection, and localization of the vehicle within the lane for proper navigation.

The affordance prediction model, g_ϕ, that predicts $\{hp_t, hv_t, hr_t, \psi_t\}$ is obtained by training a MLP, which receives the output from the pre-trained $h_\theta(o_t)$ as input. In Table 3.1, we detail the parameters of network architectures for different action-based representation learning approaches, as well as the network for affordance projection.

3.4 Experiments

3.4.1 Environment

As in most previous works addressing autonomous driving, we perform our experiments and data collection in the CARLA simulator [50], in particular, using version 0.9.6. We rely on the widely used Town 1 for training and the unseen new town Town 2 for testing.

3.4.2 Training Dataset

In order to collect the action-based supervised dataset ($\mathcal{D}^u$) and the weakly supervised dataset ($\mathcal{D}^l$), we modified the default CARLA's autopilot for not only recording o_t and a_t but also our image-level labeled affordances (y_t). We collected ~ 50 hours of image sequences in Town 1 for training purposes, balancing the training weather conditions, at 20 fps. In this data collection process, we have three cameras, a forward-facing (central) camera from which we will drive at testing time, and two lateral cameras only used for training purposes as in [20, 39]. Thus, in terms of samples to train h_θ, we have $N_u \sim 10,800,000$. This dataset plays the role of $\mathcal{D}^u$,

*Classification: M = 2; Regression: M = 1)

Module	Input Dimension	Output Channels	Num. of Dropout
ResNet 34	200 × 88 × 3	512	0.0
Speed	1	128	0.0
	128	128	0.0
Command	4	128	0.0
	128	128	0.0
Join	512+128+128	512	0.0
Action Branch	512	256	0.0
	256	256	0.5
	256	3	0.0
Speed Branch	512	256	0.0
	256	256	0.5
	256	1	0.0

(a) Behavior Cloning (BC)

Module	Input Dimension	Output Channels	Num. of Dropout
ResNet 34	200 × 88 × 3	512	0.0
Speed	1	128	0.0
	128	128	0.0
Command	4	128	0.0
	128	128	0.0
Join	512+128+128	512	0.0
Join (Z_t,Z_{t+1})	512 + 512	512	0.0
Action (a_t)	512	256	0.0
	256	256	0.0
	256	3	0.0

(b) Inverse model

Module	Input Dimension	Output Channels	Num. of Dropout
ResNet 34	200 × 88 × 3	512	0.0
Speed	1	128	0.0
	128	128	0.0
Command	4	128	0.0
	128	128	0.0
Action (a_t)	3	512	0.0
	512	256	0.0
	256	512	0.0
Join	512+128+128	512	0.0
Join (Z_t, Z_{t+1})	512 + 512	512	0.0
Action (a_t)	512	256	0.0
	256	256	0.5
	256	3	0.0
Join (Z_t, Action)	512 + 512	512	0.0

(c) Forward Model

Module	Input Dimension	Output Channels	Num. of Dropout
Affordances	512	512	0.0
	512	256	0.0
	256	M*	0.0

(d) Affordances Network

Table 3.1: Network architecture details for the encoders h_θ: (a) Behavior Cloning, (b) Inverse model, (c) Forward model and (d) the affordance projection network g_ϕ when fine-tuned for driving.

while to play the role of $\mathscr{D}^l$ we selected $\mathscr{D}^u$'s sub-sequences corresponding to 1% (30 minutes) and 10% (5 hours) of the total amount. In this case, we only consider images acquired by the central camera; thus, totaling $N_l \sim 36,000$ for 1% and $N_l \sim 360,000$ for 10%. These sub-sequences were selected semi-randomly to ensure that jointly form a dataset where the relative heading angle approximates a Gaussian distribution centered at $\psi_t = 0$. Finally, for testing purposes, two new datasets were collected, namely, by driving $\sim$ 1 hour in Town 1 and also $\sim$ 1 hour in Town 2. This driving was balanced among all weather conditions, and only the central camera is considered; thus, for each town we have $\sim 72,000$ images.

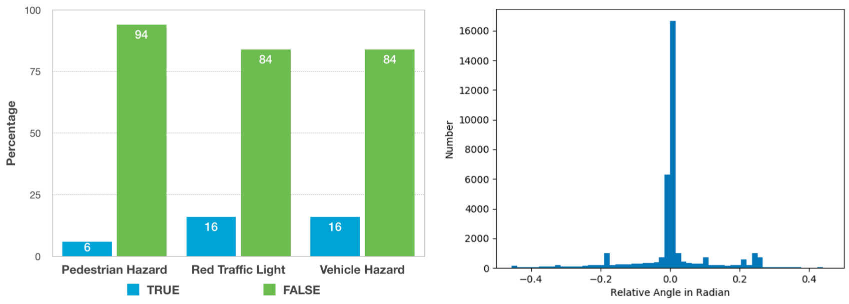


Figure 3.5: The histogram and distribution of affordances on the $\sim$ 30 minutes training dataset.

For reference, in Figure 3.5, we show the histogram and distributions of the 1% (30 minutes) $\mathscr{D}^l$ dataset, which was used for training the affordances prediction network g_ϕ. This dataset is a subset of the full 50 hours dataset, carefully sampled to maintain similar distribution as the full dataset. Note that for the hazard cases in traffic, the samples of *False* are more than *True*, which is reasonable since it is the norm in actual daily driving.

3.4.3 Training Protocal

Baselines. In Section 3.3.2 we have presented the action-based pre-training strategies for h_θ that we want to study. In addition, we have incorporated ST-DIM [4], a contrastive representation learning baseline used by agents playing Atari games. We have modified the code provided by the authors just to include ResNet34 as the backbone, *i.e.*, as for the rest of the pre-training strategies. In short, ST-DIM is trained to answer if two frames are consecutive or not, without any action-related information involved. Moreover, for the Inverse, Forward, and ST-DIM strategies, we have included seldom variants that require collecting additional $\sim$ 20 hours of image sequences in Town 1. However, in this case, instead of relying on our expert driver autopilot, the driving was *random*; thus, eventually running into accidents, driving over the sidewalk, in the wrong lane, *etc.* In short, navigating by random actions. As additional baselines, we have used ImageNet and no pre-training (random initialization).

Training Details. We train the encoder h_θ for 100K iterations (mini-batches) using $\mathscr{D}^u$. Then, using $\mathscr{D}^l$ we train the affordance prediction model, g_ϕ, for 20K iterations, no matter if this stage relies on linear classification/regression or fine-tuning. These iteration values were found by preliminary experiments where we monitored training convergence. When $\mathscr{D}^l$ assumes 10% of the dataset, we iterated 100K. In all cases, we used the Adam optimizer with an initial learning rate of 0.0002 and batch size of 120. Moreover, for any kind of training, after 75K iterations, the learning rate becomes half. As observed in [40], we saw a meaningful random seed variation for training the encoders. Thus, for obtaining reliable results, we repeat both the encoder and affordance training over three different random seeds and pick up the model with the best performance on training town for driving.

3.4.4 Evaluation Metrics

Linear Probing. We start by evaluating the representation learning capabilities of action-based methods using the commonly applied linear probing technique [2, 31, 74, 124]. This can be seen as an offline evaluation to assess the trained encoder h_θ. More specifically, using a frozen h_θ as a feature extractor, we train a linear classifier to predict affordances with an affordance dataset $\mathscr{D}^l = 1\%$. Each trained model is tested in the $\sim$ 1 hour static testing sets for both Town 1 and Town 2, while we take Town 2 results as our main analysis since it is an unseen town during training. In order to assess the performance of binary-affordance models we use the F1-score, while for assessing the performance of the relative heading angle, we use MAE. For better analysis, we divide the relative heading angle into three cases, left turn, straight and right turn, according to the navigation situation. We consider as left regime those cases where the relative heading angle ground truth is lower than -0.1 rad, as right regime when it is larger than 0.1 rad, and as straight regime otherwise.

NoCrash Benchmark. For online evaluation, we fine-tune the encoder network h_θ using three layers as the projection g_ϕ to output the affordances needed for a PID controller. We assess the driving performance on the CARLA NoCrash benchmark [40], which mainly focuses on the capabilities of models to drive in the presence of pedestrians and vehicles. The objective is to complete a set of goal-oriented episodes *without crashing*. In other words, once the AI driver crashes an obstacle (*i.e.* pedestrian, vehicle, or static asset), the episode is terminated and considered a failure case. This makes the biggest difference from the original CARLA benchmark, where the AI driver was still able to continue driving even if it hit other objects. Moreover, instead of assessing four tasks of *Straight, One Turn, Navigation, Dynamic Navigation* from the CARLA default benchmark, NoCrash consists of three tasks with increasing levels of difficulty: *Empty*, *Regular*, and *Dense*, according to the number of dynamic objects in the scene (*i.e.*, pedestrians and vehicles).

We updated the CARLA NoCrash benchmark to CARLA 0.9.6 version augmented with the new pedestrian crossing algorithms (recently incorporated into the last CARLA version). CARLA 0.9.6 version has some differences compared with CARLA 0.8.4, in which the main one is related to pedestrians. More specifically, at present, pedestrians are capable to cross roads at intersections, which was not possible in the old version. In addition, they are able to cross roads in groups and are more equally spread over the town. As a consequence, it is necessary to change the number of pedestrians spawned in the town in order to reproduce the benchmark from version 0.8.4. For the regular task, we increased the number of pedestrians from 50 to 125 for Town 1, and 50 to 100 for Town 2. In dense task, we increased from 250 to 400 for Town 1, and from 150 to 300 for Town 2.

The routes have been also changed since the new API allowed a much more controlled sampling of the routes to be used on the benchmark. Thus, we made some changes in the routes to guarantee they followed the restrictions described in [50]. We restrict at least 1000 meters distance between the start and end points for Town 1 and 500 for Town 2.

The visuals also have been slightly changed especially with respect to the dynamic obstacles. We can see in Figure 3.6 some new features of the benchmark that are present in version 0.9.6. At present, the vehicles include bikes, motorbikes, and infant pedestrians. Our update was analogous to the one done by [28], however, instead of implementing it ourselves, we added to version 0.9.6 the pedestrian navigation algorithm provided by the official CARLA 0.9.7 version. Note that for this work, the version used was still CARLA 0.9.6, while only the pedestrian navigation from 0.9.7 was added. We decided to stay on version 0.9.6 since newer versions of CARLA have incorporated major differences in traffic management that drastically changed vehicle behavior.

Controller. In order to perform driving evaluations, we have tuned a PID controller that takes the estimated affordances as input and outputs the action commands (a_t) to control the AV. Given a set of perfect affordances (*i.e.*, those labeled as ground truth) at time t, $\{\psi(t), hv(t), hp(t), hr(t)\}$, the controller outputs an action $a(t)$ defined by $\{S(t), T(t), B(t)\}$, *i.e.*, steering, throttle, and break, respectively. In particular, for lateral control (*i.e.*, $S(t)$) and for longitudinal control (*i.e.*, $T(t)$ and

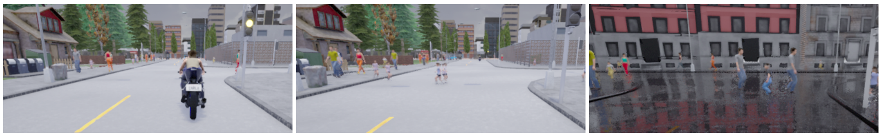


Figure 3.6: New types of vehicles and pedestrians present on the updated CARLA 0.9.6 version of the benchmark. Left: motorbikes were not included in the previous benchmark. Middle: children are added as part of pedestrians. Right: pedestrians are able to better agglomerate when crossing roads.

	↑ Binary Affordances			↓ Relative Angle (ψ_t)		
Pre-training	Pedestrian (hp)	Vehicle (hv)	Red T.L. (hr)	Left Turn	Straight	Right Turn
No pre-training	26 ± 0	50 ± 1	42 ± 0	11.38 ± 0.18	1.85 ± 0.03	24.68 ± 0.03
ImageNet	37 ± 2	75 ± 0	47 ± 0	11.69 ± 0.57	2.83 ± 0.07	25.55 ± 0.10
Contrastive (ST-DIM)	47 ± 1	53 ± 0	53 ± 0	10.43 ± 0.21	2.62 ± 0.03	18.75 ± 0.23
Forward	**50 ± 0**	63 ± 0	60 ± 0	5.35 ± 0.03	0.52 ± 0.00	6.61 ± 0.03
Inverse	49 ± 0	78 ± 0	70 ± 0	**3.57 ± 0.03**	**0.46 ± 0.00**	**3.78 ± 0.06**
Behavior Cloning (BC)	47 ± 0	**81 ± 0**	**75 ± 0**	4.89 ± 0.03	1.24 ± 0.03	6.25 ± 0.10

Table 3.2: Linear probing results in unseen Town 2. Left: F1 score for the binary affordances (higher is better). Results are scaled by 100 for visualization purposes. Right: MAE of the relative angle (lower is better), shown for different navigation maneuvers. MAE is shown in degrees for an easier understanding.

$B(t)$) we use the following PID-based equations:

$$\begin{aligned}
S(t) &= PID(\psi(t)) = K_p\psi(t) + K_i \int_0^t \psi(\tau)d\tau + K_d \frac{\partial\psi(t)}{\partial t}, \\
B(t) &= max(hr(t),\ hp(t),\ hv(t)), \\
T(t) &= PID(0 \text{ if } B(t) > 0,\ v \text{ otherwise}),
\end{aligned}$$

where the hazard functions either equal to 1 or 0, and v is the target maximum speed, 20Km/h in these experiments. We tuned the constants K_p, K_i, and K_d in Town 1 to obtain a perfect driving (no errors, all episodes completed) for the *dense* condition of the NoCrash benchmark provided we use ground truth (perfect) affordances. We did it in that way to provide a driving evaluation directly depending on the quality of the affordance predictions, not in the controller itself since it is not the focus of this work.

	↑ Binary Affordances			↓ Relative Angle (ψ_t)		
Pre-training	Pedestrian (hp)	Vehicle (hv)	Red T.L. (hr)	Left Turn	Straight	Right Turn
No pre-training	26 ± 0	50 ± 1	42 ± 0	11.38 ± 0.18	1.85 ± 0.03	24.68 ± 0.03
Contrastive (ST-DIM)	41 ± 0	62 ± 1	63 ± 1	9.01 ± 0.46	2.77 ± 0.18	18.37 ± 0.45
Contrastive Random (ST-DIM)	39 ± 1	$\mathbf{73 \pm 1}$	47 ± 0	9.70 ± 0.41	2.98 ± 0.11	15.89 ± 0.41
Forward	$\mathbf{50 \pm 0}$	51 ± 0	58 ± 0	4.87 ± 0.00	0.52 ± 0.00	6.07 ± 0.06
Forward Random	20 ± 1	38 ± 0	16 ± 0	11.54 ± 0.03	1.20 ± 0.00	19.14 ± 0.00
Inverse	45 ± 0	66 ± 0	$\mathbf{73 \pm 0}$	$\mathbf{3.02 \pm 0.03}$	$\mathbf{0.42 \pm 0.03}$	$\mathbf{5.06 \pm 0.17}$
Inverse Random	26 ± 0	49 ± 0	59 ± 0	8.50 ± 0.53	1.45 ± 0.03	13.14 ± 0.34

Table 3.3: Linear probing results comparing encoders trained with random policy training data versus expert demonstration data. Note that, to provide a fair comparison, the encoders here were trained with 20 hours of data, which are different than the ones from Table 3.2.

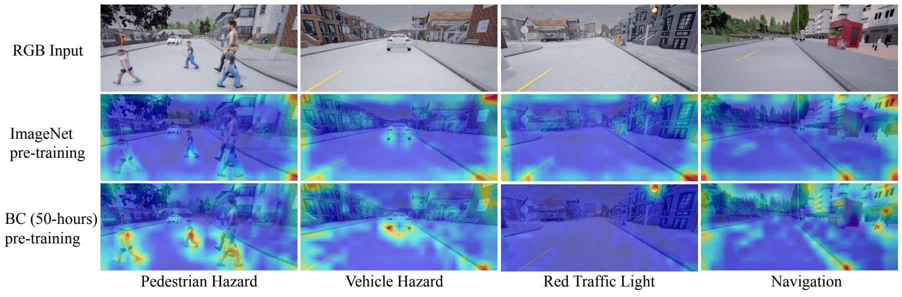


Figure 3.7: **Attention heatmaps.** Top: RGB input images from a town unseen during training. Mid: attention heatmaps of ImageNet pre-trained encoder. Bottom: attention heatmaps of a BC encoder pre-trained with 50 hours of expert driving data. From left to right, we show some cases involving different affordances: pedestrian hazard, vehicle hazard, red traffic light detection, and navigation.

3.4.5 Experimental Results

Linear probing. Table 3.2 shows the F1/MAE scores for the affordances prediction. For each pre-training strategy, we repeat linear classifier training with three random seeds and compute its mean and standard deviation. Note that those results consist of zero-shot generalization to an unseen town (Town 2). The main observation is that action-based pre-training (Forward/Inverse/BC) outperforms all the other reported pre-training strategies. However, we see that the action-based pre-training is mostly beneficial to help reliably estimate the vehicle's relative angle with respect to the road. The contrastive method, ST-DIM, shows promising results for the binary affordances but results on very poor relative angle estimations. We

	↑ Binary Affordances			↓ Relative Angle (ψ_t)		
Pre-training	Pedestrian (hp)	Vehicle (hv)	Red T.L. (hr)	Left Turn	Straight	Right Turn
No pre-training	38 ± 1	59 ± 0	45 ± 0	10.03 ± 0.06	1.80 ± 0.03	17.57 ± 0.03
ImageNet	35 ± 1	67 ± 0	54 ± 1	14.46 ± 0.35	2.31 ± 0.09	19.06 ± 0.18
Contrastive (ST-DIM)	38 ± 0	57 ± 1	76 ± 1	6.72 ± 0.03	2.65 ± 0.03	13.12 ± 0.11
Forward	56 ± 0	62 ± 0	55 ± 0	6.97 ± 0.12	**0.17** ± 0.00	5.88 ± 0.03
Inverse	**57** ± 1	82 ± 0	**89** ± 0	3.84 ± 0.06	0.21 ± 0.03	**2.96** ± 0.09
Behavior Cloning (BC)	46 ± 0	**83** ± 0	86 ± 0	**3.76** ± 0.03	0.63 ± 0.00	4.35 ± 0.06

Table 3.4: Linear probing results on Town 1 testing set. Left: F1 score for the binary affordances (higher is better). Results are scaled by 100 for visualization purposes. Right: MAE of the relative angle (lower is better), shown for different navigation maneuvers. MAE is shown in degrees for an easier understanding.

also observed a poor generalization capability for ImageNet pre-training. These results suggest that a useful scene representation is learned by training encoders with expert demonstration data. Additional evidence is presented in Figure 3.7, which shows examples of attention heatmaps from an ImageNet encoder, and a BC encoder that was trained with 50 hours of data. These attention maps are calculated by the simple average of the feature maps from the third ResNet34 block. We can see that the necessity to imitate expert demonstrations creates activations on useful objects such as pedestrians, vehicles, traffic lights, and lane markings; which is in agreement with the fact that a linear classifier can predict well the set of affordances with the action-based pre-training.

We also further study if the source data needs to come from expert driving or from random actions as in [1]. The general intuition is that the Inverse model can learn a good representation by learning the dynamics of the scene. We show on Table 3.3 that inverse model can, indeed, outperform the no pre-training condition even when using random actions. However, we show that there is a lot more benefit for representation learning obtained from expert action information than from random action. For ST-DIM, as expected, the difference between the random and expert policy is smaller since it is not based on action.

As a reference, we also provide results the linear probing evaluation results of models tested on the ~ 1 hour Town 1 testing set in Table 3.4 and Table 3.5. This testing set has a similar appearance to the training data. We observe a similar tendency to the results obtained on Town 2.

Driving results We report the success rate (higher is better) on the driving tasks and the percentage of traffic lights crossed in red (lower is better). For each model, we repeat driving three times and compute its mean and standard deviation. Table

	↑ Binary Affordances			↓ Relative Angle (ψ_t)		
Pre-training	Pedestrian (hp)	Vehicle (hv)	Red T.L. (hr)	Left Turn	Straight	Right Turn
No pre-training	38 ± 1	59 ± 0	45 ± 0	10.03 ± 0.06	1.80 ± 0.03	17.57 ± 0.03
Contrastive (ST-DIM)	36 ± 1	60 ± 2	67 ± 1	7.43 ± 0.17	2.79 ± 0.18	12.11 ± 0.18
Contrastive Random (ST-DIM)	34 ± 2	$\mathbf{78} \pm 1$	52 ± 2	11.10 ± 0.52	2.02 ± 0.09	14.65 ± 0.29
Forward	$\mathbf{52} \pm 0$	53 ± 0	60 ± 0	3.95 ± 0.00	0.11 ± 0.00	4.37 ± 0.03
Forward Random	5 ± 0	27 ± 0	2 ± 0	19.10 ± 0.03	0.63 ± 0.00	17.97 ± 0.03
Inverse	48 ± 1	71 ± 0	$\mathbf{90} \pm 0$	$\mathbf{2.03} \pm 0.07$	$\mathbf{0.21} \pm 0.03$	$\mathbf{2.54} \pm 0.03$
Inverse Random	34 ± 0	53 ± 0	55 ± 0	12.49 ± 0.45	0.84 ± 0.03	13.22 ± 0.54

Table 3.5: Linear probing results on Town 1 testing set, comparing encoders trained with random policy training data versus expert demonstration data. Note that, to provide a fair comparison, the encoders here were trained with 20 hours of data.

	Training Town				New Town			
Technique	Empty	Regular	Dense	T.L.	Empty	Regular	Dense	T.L.
No pre-training	78 ± 4	79 ± 7	48 ± 4	12 ± 1	57 ± 1	37 ± 4	10 ± 1	18 ± 2
Image Net	86 ± 1	$\underline{89 \pm 3}$	66 ± 1	12 ± 0	21 ± 3	19 ± 3	13 ± 5	23 ± 2
Contrastive (ST-DIM)	73 ± 2	84 ± 4	$\underline{\mathbf{71} \pm 3}$	10 ± 1	66 ± 5	49 ± 2	17 ± 0	21 ± 1
Forward	68 ± 3	84 ± 3	$\underline{68 \pm 8}$	$\underline{9 \pm 1}$	49 ± 5	37 ± 3	18 ± 5	13 ± 2
Inverse	73 ± 4	82 ± 2	61 ± 3	$\underline{\mathbf{8} \pm 1}$	$\underline{\mathbf{83} \pm 2}$	$\mathbf{67} \pm 8$	$\underline{\mathbf{26} \pm 8}$	$\underline{\mathbf{8} \pm 0}$
Behavior Cloning (BC)	$\mathbf{91} \pm 1$	$\underline{\mathbf{91} \pm 4}$	$\underline{68 \pm 5}$	$\underline{\mathbf{8} \pm 1}$	$\underline{83 \pm 3}$	61 ± 4	$\underline{25 \pm 4}$	$\underline{\mathbf{8} \pm 1}$
CILRS 0.8.4 [40]	97 ± 2	83 ± 0	42 ± 2	47	66 ± 2	49 ± 5	23 ± 1	64
LBC [28]	97 ± 1	93 ± 1	71 ± 5	N/A	100 ± 0	94 ± 3	51 ± 3	N/A

Table 3.6: Comparison of the success rate of action-based pre-training with baselines (top) and other methods from the literature (bottom). We are able to surpass ImageNet pre-training and the CILRS baseline. Results are from the CARLA 0.9.6 NoCrash benchmark.

3.6 compares the performance (success rate) of the action-based methods with the baselines and other methods from the literature. For all implemented methods, we considered fine-tuning with $\mathscr{D}^l = 10\%$. Firstly, we see that the Inverse model and BC are the best representation learning strategies to pre-train the encoder, especially in the new town. Both models also clearly outperform the contrastive-based baseline (ST-DIM) in the new town. However, in the training town, the contrastive method obtained relevant results, especially under dense traffic. As a reference, we report results from Learning by Cheating (LBC) [28] and the CILRS method [40], presented at the bottom of Table 3.6. Shown results are copied from the corresponding papers. Our proposed approaches also achieved very close results to the LBC method [28] and outperform CILRS especially when reacting to traffic lights. Note that the LBC method requires high supervision for training a teacher network, which teaches a student network to drive end-to-end. Our method uses much less densely labeled

		Training Town				New Town			
(D^l)	(D^u)	Empty	Regular	Dense	T.L.	Empty	Regular	Dense	T.L.
0.5 hours	No Data	25 ± 4	11 ± 3	3 ± 2	51 ± 2	1 ± 1	1 ± 1	0 ± 0	57 ± 4
	5 hours	68 ± 5	58 ± 5	27 ± 5	$\underline{\mathbf{10} \pm 1}$	14 ± 1	7 ± 1	3 ± 1	23 ± 2
	20 hours	**98** ± 1	**93** ± 2	**61** ± 6	$\underline{\mathbf{10} \pm 1}$	17 ± 2	12 ± 2	2 ± 2	15 ± 2
	50 hours	95 ± 2	87 ± 2	47 ± 2	12 ± 1	**25** ± 3	**20** ± 4	**6** ± 1	**10** ± 1
5 hours	No Data	78 ± 4	79 ± 7	48 ± 4	12 ± 1	57 ± 1	37 ± 4	10 ± 1	18 ± 2
	5 hours	58 ± 6	80 ± 4	$\underline{69 \pm 5}$	$\underline{9 \pm 1}$	68 ± 6	55 ± 5	23 ± 3	12 ± 2
	20 hours	74 ± 2	81 ± 4	$\underline{\mathbf{70} \pm 5}$	10 ± 1	$\underline{82 \pm 2}$	$\underline{\mathbf{66} \pm 5}$	**34** ± 0	12 ± 1
	50 hours	**91** ± 1	**91** ± 4	$\underline{68 \pm 5}$	$\underline{\mathbf{8} \pm \mathbf{1}}$	$\underline{\mathbf{83} \pm 3}$	$\underline{61 \pm 4}$	25 ± 4	**8** ± 1

Table 3.7: Driving performance ablation study of the BC pre-training encoder investigating the quantity of expert driving data (D^u) given a smaller amount of labeled affordances (D^l).

	Technique	Empty	Regular	Dense	T.L.
Training	BC driving	72 ± 5	47 ± 2	20 ± 3	74 ± 2
	BC pre-training	**91** ± 1	**91** ± 4	**68** ± 5	**8** ± 1
New Town	BC driving	71 ± 2	42 ± 7	12 ± 3	63 ± 1
	BC pre-training	**83** ± 3	**61** ± 4	**25** ± 4	**8** ± 1

Table 3.8: We compare the behavior cloning (BC) technique used as driving technique (BC driving) to it used as a pre-training technique (BC pre-training) for an affordances-based model.

data and does not use dataset aggregation (Dagger [144]). Moreover, note that reported results from CILRS are on CARLA version 0.8.4, the benchmark from the newer version is considerably more difficult.

An important observation is that using expert demonstration as pre-training seems to be more beneficial than training a model end-to-end to directly perform control. In Table 3.8, we show a behavior cloning encoder trained with 50 hours of expert demonstrations. We compare two different uses of this encoder: directly producing driving controls and serving as representation learning for an affordance prediction model. With our pre-training strategy and the complementary affordance training, our model (BC pre-training) is able to greatly outperform the end-to-end driving results (BC driving). This difference is expressive especially when comparing the capability to stop at red traffic lights. Finally, note that the “BC driving" results from Table 3.8 are in practice a re-training of the CILRS [40] baseline to work on version 0.9.6.

Ablation study In Table 3.7, we analyze the data amount impact of both supervised data and expert demonstrations. We used the simplest pre-training encoding, BC. We observe a clear correlation between performance improvement and the quantity of action-based supervised data D^u. We can see that with only 30 minutes (0.5 hours) of labeled data, the pre-trained network has a performance close to our best-reported results in training town. For the generalization conditions, however, only 0.5 hours of labeled data are not enough to obtain satisfactory results. Pre-training is useful also when using higher amounts of labeled data (5 hours), as shown on the bottom of Table 3.7. However, the impact of pre-training when more labeled data is available is clearer in New Town. Note that when comparing 50 hours with 20 hours, on pre-training, the results are similar, since 20 hours driving seems to be sufficient to capture the inherent variability of Town 1.

3.5 Conclusion

In this chapter, we have shown that representations learned by using action-based methods (BC or Inverse model) are promising as pre-trained representations for autonomous driving controllers based on affordances. Moreover, we have found that most of the benefit comes when the driving experiences (actions) are captured from proper driving (humans). In other words, expert driving outperforms random roaming for representation learning. While considerable future research is needed to improve the raw performance of the methods explored here, the fact that the required data can be easily obtained by simply recording the actions of good drivers, highlights the potential of action-based methods for learning representations for autonomous vehicles beyond pure end-to-end autonomous driving models.

It would be relevant to explore if the pre-training strategy presented here can be also helpful for training labeling-intensive visual models, such as those for semantic class/instance segmentation or object detection. Further, it would be interesting to examine other strategies to use the expert driver data in order to further improve performance.

4 Scaling Vision-based End-to-End Autonomous Driving with Multi-View Attention Learning

4.1 Introduction

End-to-end autonomous driving (EtE-AD) refers to deep models trained to process sensor data for performing maneuvers that imitate human (expert) driving [168]. Broadly, we can classify these models according to their training supervision requirements. Some models only require vehicle signals (*e.g.*, steering angle, acceleration) as supervision. Eventually, they can be directly trained from millions of human driving experiences. Other models also require costly human-based sensor data labeling as supervision (*e.g.*, pixel/voxel-wise semantic labels, or 2D/3D object bounding boxes). In fact, these models follow a kind of hybrid approach leveraging from pure EtE-AD and traditional AD pipelines [68, 197].

Despite the appealing idea of developing vision-based pure EtE-AD models, their progress has mostly stalled, giving space to hybrid models supervised by a significantly large amount of labeled sensor data [27, 28, 35, 80, 105, 118, 133, 155, 181, 199], or by privileged information from the driving environment as required by reinforcement learning [173, 199]. This situation may be due to the apparent lack of scalability of pure EtE-AD models raised in a few works [44, 164]; including [40], where the model known as CILRS is proposed. CILRS was developed in suboptimal conditions: limited driving episodes based on a single and rather deterministic expert driver, very low-resolution images depicting a relatively narrow horizontal field of view (on-board images roughly display a single lane), and without applying any attention mechanism. Overall, this gives rise to poor performance in newer benchmarks, eventually misleading non-expert readers and new practitioners in the field regarding the potential of vision-based pure EtE-AD.

Since having strong baselines is important to avoid illusory gains when developing new models, we present *CIL++*, a strong vision-based pure EtE-AD model trained by conditional imitation learning, *i.e.*, CILRS. We improve on CILRS key limitations, rising the performance of CIL++ to be on par with top-performing hybrid methods. First, we drastically increase the image view by using a horizontal field of view (HFOV) similar to human drivers, which runs on $180° - 220°$. Further, we propose a visual transformer-based architecture [176] which acts as a mid-level attention mechanism for these views, allowing CIL++ to associate feature map patches (tokens) across different views. These changes allow CIL++ to perform at the expert level on the CARLA *NoCrash* metrics. Moreover, CIL++ is the first vision-based pure EtE-AD model capable of obtaining competitive results on complex CARLA towns.

We present the work as follows. Section 4.2 reviews the papers relating to this work. Section 4.3 presents the CIL++ which is a vision-based EtE-AD baseline with multi-view attention learning. In Section 4.4, we summarize the experimental setting and the obtained results. Finally, Section 4.5 draws our main contributions and conclusions.

4.2 Related Work

Learning a driving policy from experts, instead of handcrafting it, is a really appealing idea. Accordingly, in EtE-AD, a deep model is trained by imitation learning, which has become an active research topic [168]. Pioneering works following this approach are [20, 21, 100, 131]. However, it was the public release of the CARLA simulator [50], together with a vision-based pure EtE-AD model trained and tested on CARLA [39], that attracted great attention to this paradigm [14, 17, 27, 35, 40, 80, 118, 133, 155, 181, 185, 199].

Basically, we can find two ways of approaching EtE-AD according to the output of the underlying deep model. On the one hand, different proposals output waypoints on bird-eye-view (BeV) coordinates [14, 27, 28, 35, 118, 133, 155]. These are then used by a controller for providing the proper steering and acceleration in driving. Note that generating BeVs involves 3D knowledge of the scene. On the other hand, other proposals directly output the ego-vehicle signals (steering, acceleration) [17, 39, 40, 80, 105, 126, 185, 199], which act on the vehicle either directly or after some signal-stabilizer filtering (*e.g.*, using a PID). Both approaches were combined, *e.g.*, in [181], where a branch of the EtE-AD model is used to predict waypoints and another to predict ego-vehicle signals, being the model output a combination of both, weighted according to the perceived road curvature.

Another important difference among EtE-AD models is the type of required supervision for their training. For instance, some models require semantic segmentation labels at training time [28, 35, 80], sometimes together with object bounding boxes (BBs) [133], or BBs and HD maps [14]. Other models are only supervised by ego-vehicle available signals [20, 21, 40, 100, 131], thus no human-labeled sensor data is required. EtE-AD models can also fuse multi-modal sensor data such as RGB image and depth from either LiDAR [27, 133, 155] or monocular depth estimation [185].

In recent literature, vision-based pure EtE-AD models, such as CILRS [39], are clearly outperformed by those using additional supervision (pixel-wise semantic labels, object BBs, *etc*). For instance, the currently top-performing hybrid model, termed as MILE [80], relies on a training procedure requiring ≈ 3M images labeled for semantic segmentation. Another top-performing hybrid model, NEAT [35], requires ≈ 130K of those. In addition, Roach [199] leverages the teacher/student mechanism. First, a *teacher* model (Roach RL) is trained using environment-privileged information and reinforcement learning. Then, it supervises the training of a *student* model (Roach IL) by applying imitation learning. In this work, we refer to the student model as RIM.

CILRS was developed under completely different conditions than we have today: only using single-lane towns in the CARLA simulator (Town 1 and Town 2), using very low-resolution images depicting a narrow horizontal field of view (roughly, one-lane-width views), without including any attention mechanism, and relying on the default CARLA's expert driver, which follows handcrafted rules. Overall, this drives it to perform poorly on newer benchmarks.

Our model, CIL++, is a direct successor of CILRS, aiming at bringing back the competitiveness of vision-based pure EtE-AD. For training and testing CIL++, we use the multi-town setting available from CARLA 0.9.13. Moreover, as in recent works [80], to collect training data (images and ego-vehicle signals), we use an expert [199] with better driving performance than the CARLA's default one. Note that, in simulation, this expert plays the role of a human driver in the real world, who would drive to collect onboard data. In addition, we use a wider horizontal field of view (HFOV=180°), in the range of human drivers. This kind of inductive bias is crucial to avoid injecting undesired causal confusion when training the model. For instance, while collecting driving episodes for training, the ego-vehicle may be stopped at a red traffic light because the expert driver has access to the privileged information of the environment. However, this red light may not even be captured by the onboard camera due to a narrow HFOV. This was observed in CILRS which was using HFOV=100°. In fact, using wide HFOVs has become a common practice to develop EtE-AD models (*e.g.*, [35, 80, 133, 155, 199]). As we would do in the real world to minimize image distortion, we use three forward-facing cameras with HFOV=60° each. Finally, in order to jointly consider the image content from the three cameras (views), we propose to use Transformer [176] which acts as a mid-level attention mechanism for these views. All these improvements over CILRS make CIL++ competitive.

Like CILRS, we use the current ego-vehicle speed and high-level navigation commands as input signals to the model. However, unlike CILRS, we also consider left/right lane changes as possible command values at testing time. Note that some supervised methods such as MILE use as input the road shape pattern to be expected according to the current position of the ego-vehicle, instead of processing a high-level navigation command. In CILRS and CIL++, high-level commands (*e.g.*, *continue* in this lane) are equivalent to those from a navigation system for global planning. In fact, such navigation commands together with the ego-vehicle speed are the only information that CIL++ uses beyond the multi-view images. This is in contrast with other models such as NEAT [35], which use explicit traffic light detection at testing time.

As we will see, even CIL++ is not using supervision at all, it outperforms RIM and is quite on par with MILE.

4.3 Building CIL++: Vision-based EtE-AD with Multi-view Attention Learning

4.3.1 Problem Setup

CIL++ is trained by imitation learning, which we formalize as follows. Expert demonstrators (drivers) produce an action $\mathbf{a}_i$ (ego-vehicle maneuver) when encountering an observation $\mathcal{O}_i$, (sensor data, signals) given the expert policy $\pi^\star(\mathcal{O}_i)$ (driving skills, attitude, *etc*). The basic idea behind imitation learning is to train an agent (here CIL++) that mimics an expert by using these observations.

Prior to training an agent, we need to collect a dataset comprised of observation/action pairs $\mathcal{D} = \{(\mathbf{o}_i, \mathbf{a}_i)\}_{i=1}^{N}$ generated by the expert. This dataset is in turn used to train a policy $\pi_\theta(\mathbf{o}_i)$ which approximates the expert policy. The general imitation learning objective is then

$$\underset{\theta}{\operatorname{argmin}} \mathbb{E}_{(\mathbf{o}_i, \mathbf{a}_i) \sim \mathcal{D}} \left[\mathcal{L}(\pi_\theta(\mathbf{o}_i), \mathbf{a}_i) \right] \quad . \tag{4.1}$$

During testing time, we assume that only the trained policy $\pi_\theta(\mathbf{o}_i)$ will be used and no expert will be available.

4.3.2 Architecture

Figure 4.1 overviews the architecture of CIL++. Our model is mainly comprised of three parts: state embedding, transformer encoder, and action prediction module.

State Embedding. At the time t, the current state $\mathbf{X}_t$ consists of a set of images from the left, central, and right cameras $\mathbf{X}_t = \{\mathbf{x}_{l,t}, \mathbf{x}_{c,t}, \mathbf{x}_{r,t}\}$, the ego vehicle's forward speed $s_t \in \mathbb{R}$, and a high-level navigation command $\mathbf{c}_t \in \mathbb{R}^k$ which is encoded as a one-hot vector.

As discussed in [49], the lack of inductive biases makes transformer models require more data to achieve good performance. In order to possess the inherent

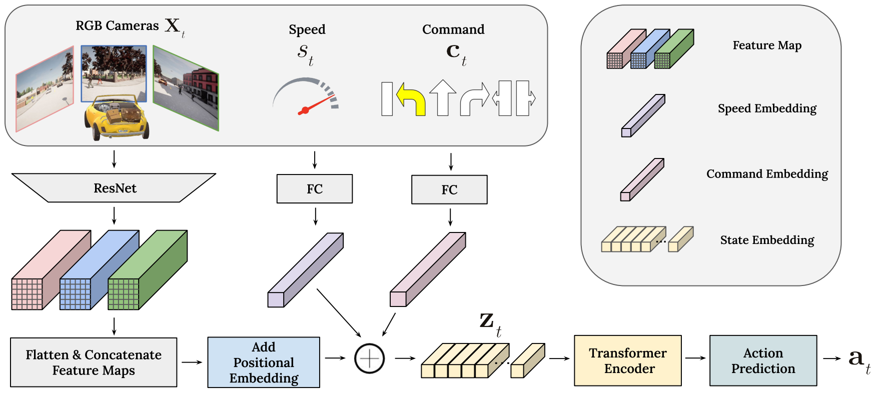


Figure 4.1: CIL++'s architecture: it is mainly comprised of three parts: state embedding, transformer encoder, and action prediction. Specifically, the input observation consists of multi-view RGB images $\mathbf{X}_t$, ego-vehicle's speed s_t, and high-level navigation command $\mathbf{c}_t$. The output is action $\mathbf{a}_t$, which is directly applied to maneuver the ego-vehicle. Since $\mathbf{a}_t$ consists of the ego vehicle's steering angle and acceleration, CIL++ is a vision-based pure EtE-AD model. Note that these action components are automatically read and associated with the captured images during data collection.

properties of CNNs (*i.e.*, exploiting locality and translation equivariance), as well as leveraging the attention mechanism of transformers, we propose to adapt the hybrid model suggested in [49] to our case. At time t, each image $\mathbf{x}_{v,t} \in \mathbb{R}^{W\times H\times 3}$ from the multi-view camera setting is processed by a share-weight ResNet34 [76], pre-trained on ImageNet [45]. Then, for each view v, we take the resulting feature map $\mathbf{f}_{v,t} \in \mathbb{R}^{w\times h\times c}$ from the last convolutional layer of ResNet34, where $w \times h$ is the spatial size and c indicates the feature dimension. Each feature map is then flattened along the spatial dimensions, resulting in $P \times c$ tokens, where $P = w * h$ is the number of spatial features per image. Since we will set our cameras to a resolution of $W \times H = 300 \times 300$ pixels, for each one we obtain $P = 100$ patches with $c = 512$ from the ResNet34 backbone.

Since we use $|\mathbf{X}_t| = 3$ views (left, central, and right cameras), we take the flattened patches for each view and tokenize them as the whole sequence with length $S = |\mathbf{X}_t| * w * h$ for further feeding into the transformer model. To provide the positional information for each token, we apply the standard learnable 1D positional embedding $\mathbf{p} \in \mathbb{R}^{S\times c}$ as done in [49], which is added directly to the token.

The forward speed s_t and command $\mathbf{c}_t$ are linearly projected to $\mathbb{R}^c$ using a fully

Module	Input Size	Output Size	Normal-ization	Non-linearity
ResNet34	$3 \times 300 \times 300 \times 3$	$3 \times 10 \times 10 \times 512$	BN	ReLu
Flatten	$3 \times 10 \times 10 \times 512$	300×512	/	/
Command	1×4 / 1×6	1×512	/	/
Speed	1	1×512	/	/
Addition	300×512 1×512 1×512	300×512	/ / /	/ / /
Positional Embedding	300×512	300×512	/	/
Transformer Encoder	300×512	300×512	LN	ReLu
Average Pooling	300×512	1×512	/	/
Action	1×512 1×512 1×256	1×512 1×256 1×2	/ / /	ReLu ReLu /

Table 4.1: Architecture details of CIL++. Dimensions are per timestep and channel-last ($NHWC$ or NC, when applicable). Dropout was not used in any layer or module (*i.e.*, set to 0.0 when applicable). For normalization, we use either BatchNorm (BN) [84] or LayerNorm (LN) [11].

connected layer. The resulting state embedding $\mathbf{z}_t$ is obtained by the addition of these input embeddings. Formally, we define the state encoder of the current state $\mathcal{X}_t = (\mathbf{X}_t, s_t, \mathbf{c}_t)$, parameterized by θ, as

$$e_\theta : \mathbb{R}^{|\mathbf{X}_t| \times W \times H \times 3} \times \mathbb{R} \times \mathbb{R}^k \to \mathbb{R}^{S \times c} \quad . \tag{4.2}$$

In CIL [39] and CILRS [40], $\mathbf{c}_t$ is treated as a switcher (condition) to trigger different MLP branches for predicting $\mathbf{a}_t$. Here we treat $\mathbf{c}_t$ as an input signal to be later processed by a transformer. This is because we have not observed obvious differences between these two approaches while treating $\mathbf{c}_t$ as input signal simplifies the training.

Attention Learning. To naturally associate intra-view and inter-view information, we adopt the attention mechanism of transformers [176]. We expect it to be effective to learn the mutual relevance between distant image patches (tokens), helping our model to associate feature map patches across views (*i.e.*, coming from visual information to the left, center, and right of the ego-vehicle).

Over the embedded space $\mathbf{Z}_t$, we learn a scenario embedding using a transformer encoder that consists of L multi-head attention layers. Each one includes Multi-headed Self-Attention (MHSA) [176], layer normalization (LN) [11], and feed-forward MLP blocks. The final output is a linear projection of the concatenated output of each attention head, which is then fed into the action prediction module. We use $L = 4$ layers, with 4 heads each. The hidden dimension D of the transformer layer is set to be equal to the ResNet output dimension, *i.e.*, $D = 512$.

Action Prediction The output of the transformer encoder with a size of $S \times c$ is average-pooled and fed into an MLP. The MLP consists of three fully connected layers (FC) with ReLu non-linearity applied between each FC. The final output action $\mathbf{a}_t \in \mathbb{R}^2$ comprises of the steering angle and acceleration (brake/throttle), *i.e.*, $\mathbf{a}_t = (a_{\mathrm{s},t}, a_{\mathrm{acc},t})$.

For a deeper understanding of the architecture, we present the input and output dimensions of each module used in our entire pipeline in Table 4.1. The pipeline flow is row-wise, that is, we start with 3 RGB images at 300 × 300 resolution, and our final output is of shape 1 × 2, corresponding to the acceleration and steering angle that the ego vehicle should apply at a given timestep.

4.3.3 Loss Function

At time t, given a predicted action $\mathbf{a}_t$ and a ground truth action $\hat{\mathbf{a}}_t$, we define the training loss as:

$$\mathscr{L}(\mathbf{a}_t, \hat{\mathbf{a}}_t) = \lambda_{\mathrm{acc}} \| a_{\mathrm{acc},t} - \hat{a}_{\mathrm{acc,t}} \|_1 + \lambda_{\mathrm{s}} \| a_{\mathrm{s},t} - \hat{a}_{\mathrm{s},t} \|_1 \ , \tag{4.3}$$

where $\|\cdot\|_1$ is the L_1 distance, λ_{acc} and λ_s indicate the weights given to the acceleration and steering angle loss, respectively. In our case, we consider steering angle and acceleration to be both in the range of $[-1, 1]$. Negative values of the acceleration correspond to braking, while positive ones to throttle. The weights are set to $\lambda_{acc} = \lambda_s = 0.5$.

In CILRS [40], speed prediction regularization is applied in the training loss to avoid the inertia problem caused by the overwhelming probability of ego staying static in the training data. We do not observe this problem in our case, thus the speed prediction branch is not applied in our setting. Our result suggests that a simple L_1 loss is able to provide compelling performance, even in a new town.

4.4 Experiments

4.4.1 Environment

For performing the driving evaluation, we updated the *NoCrash* benchmark to work with the latest CARLA version, 0.9.13. Given we have compared CIL++ with RIM and MILE, both of which were validated in CARLA 0.9.11, we provide some of the main differences between these two versions:

- **Pedestrians Seed.** The latest CARLA version (0.9.13) added a new API function “set pedestrians seed" for better reproducibility. In our experiments, we keep the same pedestrian and traffic manager seeds for all models’ running.
- **Weather Parameters.** The default weather setting parameters have been slightly changed between versions (such as ClearNoon, ClearSunset, *etc*). We provide an example in Figure 4.2, showing some appearance differences of the same scene between these two versions under the same weather condition, ClearNoon. In Table 4.2 we show an example of the changes in the weather parameters for each version of CARLA.

4.4.2 Training Datasets

As recent top-performing methods [80], for on-board data collection in CARLA, we use the *teacher* expert driver from [199], termed as Roach RL since it is based on reinforcement learning and was trained with privileged information. Roach RL shows a more realistic and diverse behavior than the default (handcrafted) expert driver in CARLA. Note that in real-world experiments we would use different human drivers as experts. We use the default settings of [199], so as in the *student* driver of [199] (RIM) as well as in [80] (MILE), the ego-vehicle is the Lincoln 2017 available in CARLA. Each of our three forward-facing cameras on board the ego-vehicle has a resolution of $W \times H = 300 \times 300$ pixels, covering an HFOV of 60°. They are placed without overlapping so that they jointly cover an HFOV=180° centered in the main axis of the ego-vehicle.

With the expert driver, ego-vehicle, and onboard cameras, we collect data for increasingly complex experiments. First, we collect a dataset from Town01 in CARLA, which is a small town only enabling single-lane driving, *i.e.*, lane change maneuvers are not possible. In particular, we collect 15 hours of data at 10 fps (~540K frames from each camera view), under four training weather conditions, namely, ClearNoon, ClearSunset, HardRainNoon, and WetNoon. In this case, CARLA’s Town02 is used for generalization testing under SoftRainSunset and WetSunset weather conditions. Second, we collect a dataset from multiple CARLA towns to include more complex scenarios such as multi-lane driving, entering and exiting highways, passing crossroads, *etc.* In order to keep the same setting as [80], we hold Town05 for testing and collect 25 hours of data at 10 fps from Town01 to Town06 (5 hours per town; ~900K frames from each camera). Training and testing weather conditions are the same for both Town 1 and Town 2.

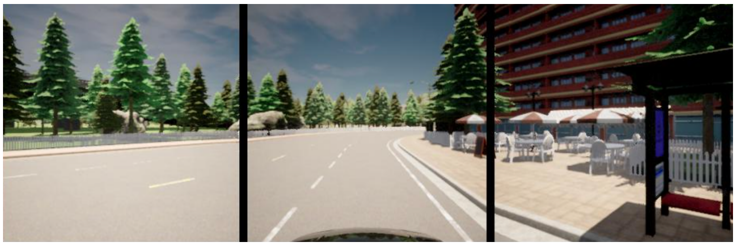
For each episode, we spawn the expert vehicle and its attached sensor suite in a random location on each map and assign a random route to it. The expert will follow this route until it has driven for the predefined episode duration. Each town has a random range which the number of pedestrians and vehicles will be (uniformly) drawn from. The route settings and towns used to collect the dataset are further detailed in Table 4.3. In Figure 4.3, we show the data distributions for both the steering angle and acceleration, for the datasets collected for training CIL++. The two datasets are of 15 and 25 hours of driving, corresponding, respectively, to the smaller, single-lane towns and multi-lane towns.

The road layouts of each town mainly influence the steering angle distribution. The concentration around 0.0 of steering angle for both datasets is due to the fact that most of the driving is done following the straight lane, *i.e.*in the towns used, the roads are mainly comprised of straight roads. Note that all of the towns in CARLA are designed with right-hand traffic, that is, in bidirectional traffic, the ego vehicle drives on the right side of the road. This results in right turns needing a higher steering angle (turning radius will be small) and left turns needing a lower steering angle (turning radius will be large).

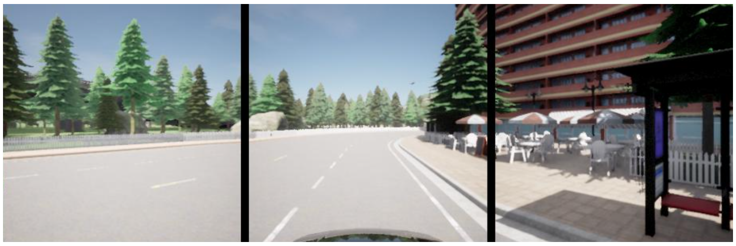
In contrast, the acceleration distribution is mainly affected by other dynamic objects in the scene. Concretely, the red traffic light or Stop sign, pedestrians crossing the road, and other leading vehicles will force the expert vehicle to stop or adjust its speed, hence leading to the large concentration of around -1.0 in the acceleration distribution for both datasets. The other concentrations around 1.0 and 0.4 are the acceleration after coming to a stop (either by blockage by other dynamic objects or traffic lights/Stop signs) and the acceleration needed to maintain a constant driving speed.

4.4.3 Training Details

To optimize Eq. (4.3), we use the Adam [94] with an initial learning rate of 10^{-4} and weight decay of 0.01. We train for 80 epochs on 2 NVIDIA A40 GPUs in parallel, with a batch size of 120. The learning rate decays by half at epochs 30, 50, and 65. More details on the training hyperparameters can be found in Table 4.4.



(a) A typical urban scene in Town 5 on CARLA 0.9.11.



(b) The same scene as (a) on CARLA 0.9.13. Note the difference in shadow length, clouds, and general lightning.

Figure 4.2: The same ClearNoon weather condition in two versions of CARLA.

	CARLA 0.9.11	**CARLA 0.9.13**
`cloudiness`	15.0000	5.0000
`precipitation`	0.0000	0.0000
`precipitation_deposits`	0.0000	0.0000
`wind_intensity`	0.3500	10.0000
`sun_azimuth_angle`	0.0000	−1.0000
`sun_altitude_angle`	75.0000	45.0000
`fog_density`	0.0000	2.0000
`fog_distance`	0.0000	0.7500
`fog_falloff`	0.0000	0.1000
`wetness`	0.0000	0.0000
`scattering_intensity`	N/A	1.0000
`mie_scattering_scale`	N/A	0.0300
`rayleigh_scattering_scale`	N/A	0.0331

Table 4.2: We take ClearNoon as an example to show weather settings differences between different versions of CARLA. Note that the last three parameters, namely `scattering_intensity`, `mie_scattering_scale`, and `rayleigh_scattering_scale` are available only since CARLA 0.9.12. For the full explanation and usage of each variable, please refer to the CARLA documentation.

	Item	**Value**			
RGB Cameras	views	3			
	Resolution	300×300			
	Field of view	60°			
	Position (x, y, z)	$(0.0, 0.0, 2.0)$			
	Rotation (ϕ, θ, ψ)	**View**	**Roll**	**Pitch**	**Yaw**
		left	0.0°	0.0°	-60.0°
		central	0.0°	0.0°	0.0°
		right	0.0°	0.0°	60.0°
	Lens circle	None			
	Frequency	10 Hz			
Episode	Dynamic Objects (random choice)	**Town**	**Pedestrians**	**Vehicles**	
		Town 1	$\mathcal{U}(80, 160)$	$\mathcal{U}(80, 160)$	
		Town 2	$\mathcal{U}(60, 80)$	$\mathcal{U}(60, 80)$	
		Town 3	$\mathcal{U}(50, 120)$	$\mathcal{U}(50, 120)$	
		Town 4	$\mathcal{U}(40, 160)$	$\mathcal{U}(60, 160)$	
		Town 6	$\mathcal{U}(60, 160)$	$\mathcal{U}(60, 160)$	
	Duration	300 s			
	Spawn point	Random			
	Route plan	Random			
	Pedestrian crossing factor	1.0			

Table 4.3: Camera and episode settings for data collection with the expert driver.

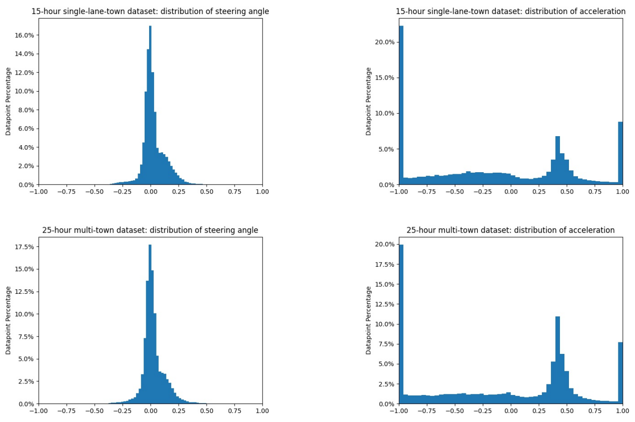


Figure 4.3: Top left to right: 15 hours single-lane-town dataset distribution of steering angle and acceleration; Bottom left to right: 25 hours multi-town dataset distribution of steering angle and acceleration. Regarding the steering angle and acceleration, negative values correspond to turning left and braking, and positive values correspond to turning right and accelerating, respectively.

Category	**Name**	**Value**
Training	GPUs	2×NVIDIA A40
	batch size	120
	seed	1314
	epochs	80
Optimizer	name	Adam
	weight decay	0.01
	β_1	0.9
	β_2	0.999
	ε	1×10^{-8}
	initial learning rate	1×10^{-4}
	learning rate decay	×0.5 at epochs 30, 50, and 65
	minimal learning rate	1×10^{-5}
Input image	resolution	300×300
	normalization	mean: [0.485, 0.456, 0.406]
		stdev: [0.229, 0.224, 0.225]
	pre-training	ImageNet
Speed Input	normalization	[-1.0, 12.0]
Transformer	number of encoder layers	4
	heads per layer	4
Loss	action weights	$\lambda_{\text{acc}} = 0.5$ (acceleration)
		$\lambda_{\text{s}} = 0.5$ (steering angle)
	output ranges	steer angle $[-1.0, 1.0]$
		acceleration $[-1.0, 1.0]$

Table 4.4: Training hyperparameters of CIL++.

4.4.4 Driving Benchmark

We follow the *NoCrash* benchmark [40] and the offline CARLA leaderboard benchmark [80, 199] for experiments on small single-lane towns (Sec. 4.4.5) and multiple towns (Sec. 4.4.5), respectively.

NoCrash Metrics. It consists of three tasks with increasing levels of difficulty: *Empty*, *Regular*, and *Dense*, according to the number of dynamic objects in the scene (*i.e.*, pedestrians and vehicles). In the *Dense* case, the default traffic density set in NoCrash always leads to congestion deadlocks at intersections [199]. Thus, we follow the *Busy* case as redefined in [199]. Each task corresponds to 25 goal-directed episodes under 2 new kinds of weather. The episode will be terminated and counted as a failure once a collision occurs. For the other infractions, the driving score will be deduced according to the penalty rule in NoCrash.

The main metric to compare driving models is the success rate (*SR*), which is the percentage of episodes successfully completed. For a fine-grained comparison, in addition, we provide the strict success rate (*S.SR*). It reflects the percentage of successful episodes under zero tolerance for any traffic infraction, such as failing to stop at a red traffic light, route deviation, *etc.* As a complement, we also include additional infraction metrics: *T.L* is the number of times not stopping at a red traffic light; *C.V* is the number of collisions with other vehicles; *R.Dev* is the number of route deviations, *i.e.*, when the high-level command is not well-executed; *O.L* accounts for the ego-vehicle driving out-of-lane (*e.g.*, in the opposite lane or in the sidewalk); *C.L* is the number of collisions with the town layout. All infraction values are normalized per driven kilometer.

Offline Leaderboard Metrics. To align our evaluation with [80], we use the offline CARLA's Leaderboard metrics for multiple towns. The most important metrics are the average driving score (*Avg.DS*) and the average route completion (*Avg.RC*). *Avg.DS* is based on penalizing driving performance according to the terms defined in CARLA's Leaderboard, while *Avg.RC* is the average distance towards the goal that the ego-vehicle is able to travel.

High-level Navigation Commands As in CILRS [40], at training time we use simple navigation commands such as *continue* in the lane, or *go-straight/turn-left/turn-right* next time an intersection is reached. However, in complex towns, after crossing an intersection in any direction, we may legally enter any of the multiple lanes. Thus, since this can be known by the global navigation system when the ego-vehicle enters a lane out of the pre-planned global trajectory, a corrective command is forced,

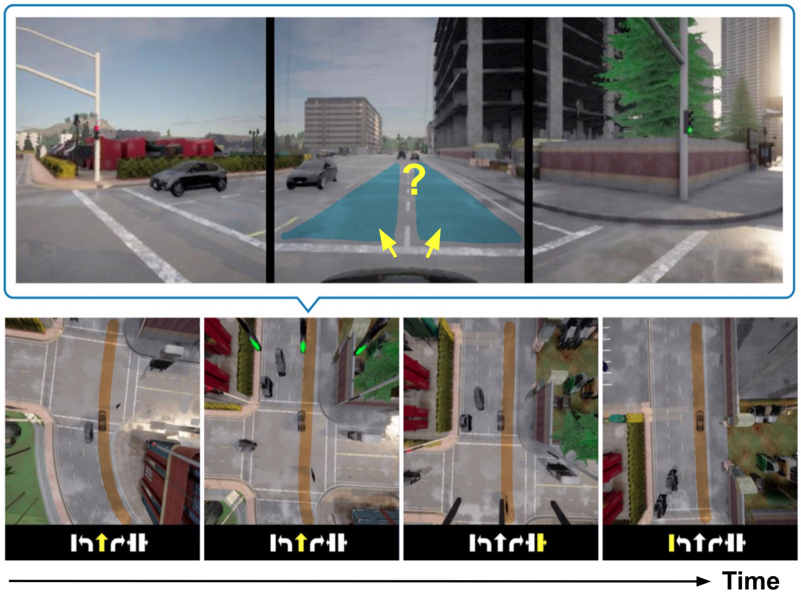


Figure 4.4: Top: when the ego-vehicle is entering a new road segment from an intersection, the *go-straight* navigation command is ambiguous. The ego vehicle can legally move to any of the highlighted lanes. Bottom: four aerial views at different times with the pre-planned global trajectory shown in **orange**. They illustrate how the high-level navigation command changes to *move-to-right-lane*, to inform how to get back to the desired trajectory.

like *move-to-left-lane* or *move-to-right-lane* as soon as possible. This corrective mechanism is used only at testing time. Figure 4.4 provides an example.

Intersection deadlock detection We observe that occasionally there are some intersection deadlocks, even though the dynamic density is not set to a high level. This could be caused by the random spawning and autopilot setting of the other vehicles in CARLA. To eliminate these cases for better comparison, we applied traffic detection. Once the traffic manager detects that some other vehicles stop within an intersection that is 10 meters from the ego-vehicle for more than 90 seconds, the vehicles will be re-spawned to other points. As an illustration, we show an example of traffic deadlocks in Figure 4.5.

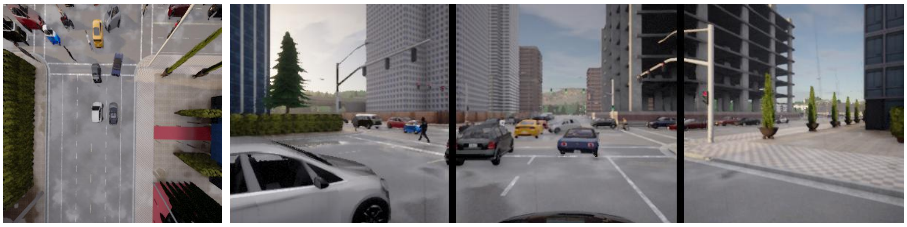


Figure 4.5: An example of traffic deadlocks in Town 5 that could lead to a timeout in route completion.

Global route re-planning As mentioned in our main paper, we applied real-time calibration on high-level navigation commands. When the ego-vehicle enters a lane out of the pre-planned global trajectory, a corrective command is forced, like *move-to-left-lane* or *move-to-right-lane* as soon as possible. However, in some cases, the ego-vehicle could not perform a lane change since there might be some other vehicles on the target lane, which forces the ego-vehicle to maintain on the driving lane. In this case, we apply the global route re-planning: once the traffic manager detects that the ego-vehicle deviates from the pre-planned route for more than 30 meters, a new route to the destination is computed, along with a new set of high-level commands to provide the ego-vehicle for driving.

4.4.5 Experimental Results

We compare CIL++ with two SOTA vision-based EtE-AD models, namely, the Roach IL model (here RIM) [199] and MILE [80]. CIL++ does not require human-labeled sensor data for training. In contrast, MILE is trained with semantic BeV as supervision, while RIM requires teaching from the Roach RL expert which was trained with privileged information.

Small Single-lane Towns We first use CARLA's Town 1 and Town 2 along with the NoCrash metrics (Sec. 4.4.4) for initial experiments. Town 1 is used for training and Town 2 for testing (Sec. 4.4.2). MILE only provides a model trained on CARLA's multiple towns, but there is no model trained only on Town 1, while RIM has versions trained on Town 1 and multiple towns. Thus, for a fair comparison, we only use RIM's single-town trained model. We show in Table 4.5 SR and S.SR for the considered traffic densities (Empty, Regular, Busy). In order to have a more focused evaluation, we show T.L only for the Empty and Regular cases, while C.V is shown only for the Busy case. Note that scenarios with no or few dynamic obstacles can

	Empty			Regular			Busy		
	↑ **SR(%)**	↑ **S.SR(%)**	↓ T.L	↑ **SR(%)**	↑ **S.SR(%)**	↓ T.L	↑ **SR(%)**	↑ **S.SR(%)**	↓ C.V
RIM	100 ± 0.0	85 ± 1.2	66 ± 5.0	97 ± 2.3	86 ± 7.2	66 ± 54	81 ± 5.0	68 ± 7.2	63 ± 52.7
CIL++	100 ± 0.0	100 ± 0.0	0 ± 0.0	99 ± 2.3	97 ± 3.1	7 ± 7.9	83 ± 7.6	77 ± 7.6	45 ± 21.5
Expert	100 ± 0.0	100 ± 0.0	0 ± 0.0	100 ± 0.0	97 ± 0.0	13 ± 4.6	84 ± 2.0	82 ± 2.0	37 ± 14.1

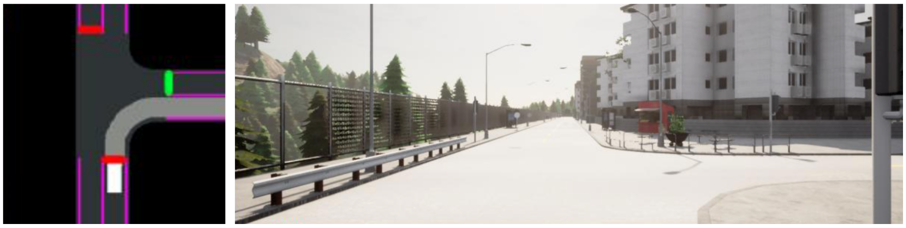
Table 4.5: Town 2 NoCrash results. RIM stands for Roach IL and the Expert is Roach RL [199]. All models are tested on CARLA 0.9.13. Mean and standard deviations are computed using three runs with different seeds. For ↑, the higher the better, while for ↓ is the opposite.

better show the ego-vehicle reaction to red traffic lights, while collisions are better evaluated in scenarios with more dynamic objects.

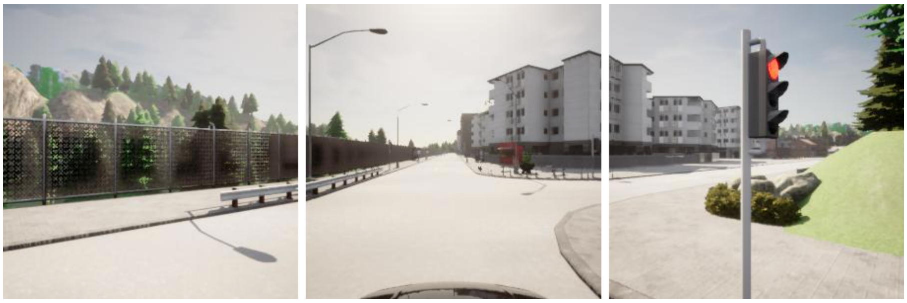
In general, CIL++ achieves the best results in all the tasks. In the Empty case, CIL++ clearly outperforms RIM in avoiding traffic light infractions, which also contributes to a better S.SR. We reach the same conclusion in the Regular case. In the Busy case, CIL++ reaches almost the expert's performance, again, being clearly better than RIM for S.SR and producing fewer collisions with vehicles. For the expert, the failure cases in Busy scenarios are due to traffic deadlocks, which lead to a timeout in route completion. Thus, its performance still can be considered as a proper upper bound.

Traffic lights tend to be on sidewalks, so detecting them from a close distance requires a sensor setting with a proper HFOV. Otherwise, causal confusion can appear. We think that the poor performance of RIM on the T.L metric is due to a narrow HFOV as illustrated in Fig. 4.6. To confirm this hypothesis, we conduct experiments using two HFOV settings for CIL++, 100 and 180. As seen in Table 4.6, we note that the number of infractions(T.L, C.L, O.L, R.Dev) increase when we use a lower HFOV=100° compared to HFOV=180°. For HFOV=100°, we have observed that the track of the road shoulder is easily out-of-observation at intersections, leading to more O.L, C.L, and R.Dev. For HFOV=180°, the ego-vehicle can better perform the right driving maneuver, thanks to having the road shoulder as a reference.

Multi-town Generalization In this section, we assess the performance of CIL++ in much more complex scenarios, as provided by CARLA's multiple towns. As mentioned in Sec. 4.4.2, for a fair comparison, we align the training and testing settings with MILE [80], using CARLA's offline Leaderboard metrics (Sec. 4.4.4). The results for all models trained on multi-town data are shown in Table 4.7. RIM shows the worst performance among the three models, incurring more infractions, thus obtaining a significantly lower Avg.DS. CIL++ achieves 98% Avg.RC, which is on



(a) Roach RL [199]: using semantic BeV as input during training time



(b) CIL++: only images from three views as input

Figure 4.6: Top Left: The expert we use for data collection is the teacher agent Roach RL [199], which has access to semantic BeVs. Note that, in simulation, this expert plays the role of a human driver in the real world, who would drive to collect onboard data. Top Right: Using an insufficient FOV, the red traffic light is not observable in the image when the expert stops close to it, which may cause causal confusion when applying imitation learning to train the student agent RIM. Bottom: CIL++ avoids this causal confusion by using a larger HFOV based on three complementary images (from different cameras). More specifically, RIM uses HFOV=100°, while CIL++ uses HFOV=180°.

	↑ **SR(%)**	↑ **S.SR(%)**	↑ **Avg.RC(%)**	↑ **Avg.DS**	↓ C.L	↓ T.L	↓ O.L	↓ R.Dev
HFOV 100°	52	40	83	69.1	428.6	19.6	339.6	10.7
HFOV 180°	100	98	100	99.2	7.3	0.0	0.0	0.0
Expert	100	100	100	100.0	0.0	0.0	0.0	0.0

Table 4.6: Impact of sensor suite HFOV in the NoCrash Regular case. For HFOV=100° we use a single camera with a resolution of $W \times H = 600 \times 170$ pixels, while for HFOV=180° we use the multi-view setting detailed in the main text.

	↑ **Avg.RC(%)**	↑ **Avg.DS**	↓ C.V	↓ C.L	↓ T.L	↓ O.L	↓ R.Dev
RIM	92 ± 3.1	51 ± 7.9	7.5 ± 1.3	4.3 ± 1.6	26.0 ± 8.9	5.4 ± 2.7	3.0 ± 3.2
MILE	98 ± 2.2	73 ± 2.9	6.0 ± 3.7	0.0 ± 0.0	3.6 ± 3.8	3.5 ± 1.5	0.0 ± 0.0
CIL++	98 ± 1.7	68 ± 2.7	6.0 ± 0.5	3.8 ± 0.7	5.8 ± 5.1	6.1 ± 2.2	9.4 ± 3.6
Expert	99 ± 0.8	89 ± 1.7	3.2 ± 1.1	0.0 ± 0.0	1.3 ± 0.4	0.0 ± 0.0	0.0 ± 0.0

Table 4.7: Town 5 results according to CARLA's offline metrics. All models are tested on CARLA 0.9.13. Mean and standard deviations are computed using three runs with different seeds. For ↑, the higher the better; for ↓, the opposite.

par with MILE. In terms of Avg.DS, MILE remains the best scoring, yielding a 73% while CIL++ achieves a 68%. We observe that this is because MILE seldom drives outside the pre-planned lane, given the route map as input. On the contrary, CIL++ lacks the explicit use of this route map since it only receives high-level navigation commands.

Visualizing CIL++'s Attention We are interested in the image content to which CIL++ pays attention. Following Grad-CAM [154], gradients flow from the action space to the final convolutional layer of the ResNet backbone. This should produce a map that highlights the important image areas for action prediction. However, since CIL++ solves a regression task, its output could be either negative or positive values, while Grad-CAM is originally designed for image classification tasks which always provide positive outputs. To adapt Grad-CAM to our case, we cannot merely take into account the positive gradient of the feature map. The computation should be divided into two cases depending on the sign of the output value. Negative gradients are used to calculate the weights for the feature map when the acceleration or steering angle value is lower than zero, otherwise, the positive gradient is used.

Fig. 4.7 shows the activation map at an intersection. Three image areas are highly activated: the traffic light in the right image, the crossing pedestrians in the central one, and the lane shoulder in the left one. Thus, we believe that CIL++ shows a proper understanding of this scene, and a clear causality between observation and action since it decides to brake due to the pedestrians, even though the traffic light is in green and a *turn-left* navigation command is given.

Ablation Study To inspect the impact of some components of CIL++, we provide an ablation study. Specifically, we are interested in the fusions of input data and multi-view.

- **Input Data Fusion.** EtE-AD models require not only sensor data but also

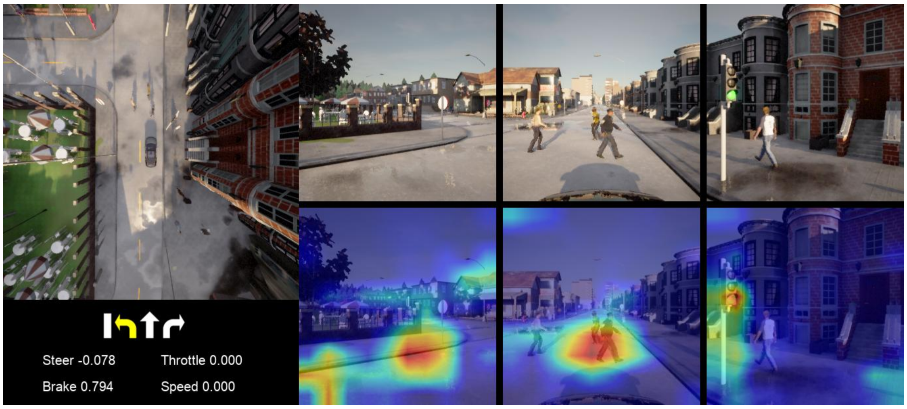


Figure 4.7: Activation maps of CIL++ at an intersection in Town 2. Three image areas from different views are highly activated: the traffic light in the right image, the crossing pedestrians in the central one, and the lane shoulder in the left one. Causality between observation and action is shown as strong braking (0.794) due to the pedestrians, even though the traffic light is green and the *turn-left* command.

signal information, like the ego-vehicle speed and a high-level navigation command. It is interesting to study how to properly fuse these inputs. To compare, we implement several types of input data fusion in Table 4.8: adding, concatenation, and tokenization. In the first, the speed and command features are simply added to the image features. This addition could be done either before (the default operation in CIL++) or after the Transformer Encoder block. We name the latter as late fusion adding (LF.A). Another common data fusion method is concatenation, which firstly stacks all the features and takes an extra join FC layer to fuse them, which we term late fusion concatenation (LF.C). Since the transformer model uses a self-attention mechanism to fuse features between tokens, we can tokenize the speed and navigation command features and feed them into the transformer block along with the image features; which we term Token in Table 4.8. Our results suggest that there is no obvious difference between tokenization and early adding fusion. These two approaches show better results than the late fusion.

- **Multi-view Fusion.** CIL++ uses attention layers to fuse multi-view information. To understand their contribution, we remove the transformer block and simply use the ResNet34 which retains the average pooling and an FC layer for embedding each image view. The embedding outputs are then stacked and

	↑ SR(%)	↑ S.SR(%)	↑ Avg.RC(%)	↑ Avg.DS
LF.A	72	62	86	78.1
LF.C	76	66	88	80.6
Token	88	80	93	88.4
CIL++	88	84	93	88.8

Table 4.8: Results of different data input fusion approaches for NoCrash Busy scenarios.

	↑ SR(%)	↑ S.SR(%)	↑ Avg.RC(%)	↑ Avg.DS
GAP	64	46	87	75.1
VS	70	62	89	78.6
CIL++	88	84	93	88.8

Table 4.9: Results of different multi-view fusion approaches for NoCrash Busy scenarios.

fed to the FC join layers for fusion. We term this approach as view stacking (VS) in Table 4.9. The speed and command features are added to the joint embedding before feeding into the action prediction MLP. We see that without the self-attention layers, the SR drops from 88% to 70%. We think this is because the average pooling layer causes a loss of spatial information, while this information is very important for actual driving. The agent should take different actions according to the location of dynamic obstacles. We also use a transformer block to process the output of the ResNet average pooling layer (GAP), instead of using the flattened feature map from the last convolutional layer of ResNet34. The results drop significantly, *e.g.*, and the SR goes from 88% to 64%.

Failure Cases We show one of our failure cases in Figure 4.8. The lane change command is given when the ego is close to the intersection while there is already another green vehicle on the right side lane. The ego agent could not make the right turn and change lanes since it recognizes there is an obstacle. As the ego proceeds, the green vehicle also turns right and continues to occupy the space that is needed for the ego to turn right. Eventually, when the ego arrives at the center of the intersection, the ego does not have enough space to turn right and stops.

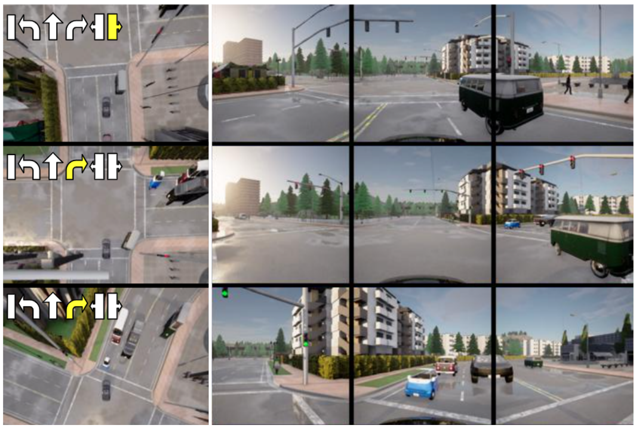


Figure 4.8: One failure case of CIL++ at intersection caused by a dilemma situation: the *move-to-right-lane* navigation command is provided around the intersection, while there is another vehicle blocking the target lane. As the ego proceeds a right turn, the other vehicle also turns right and occupies the space required for the ego vehicle to make a right turn, which leads to a failure.

4.5 Conclusion

We have presented CIL++, which aims at becoming a new strong baseline representing vision-based pure EtE-AD models trained by imitation learning. This is required because recent literature may lead to the conclusion that such approaches are poorly performing compared to those relying on additional and costly supervision. We have argued that previous vision-based pure EtE-AD models were developed in sub-optimal conditions. Thus, we have developed a model which relies on three cameras (views) to reach an HFOV=180° and a more realistic expert driver to collect onboard data in the CARLA simulator. We have proposed a visual transformer that acts as a mid-level attention mechanism for these views, allowing CIL++ to associate feature map patches (tokens) across different views. CIL++ performs at the expert level on NoCrash metrics and is a pure EtE-AD model capable of obtaining competitive results in complex towns. We have presented an ablative study showing the relevance of all the components of CIL++. In future work, we plan to add rear-view cameras, to improve lane change maneuvers.

5 Conclusions and Future Work

5.1 Conclusions

In this thesis, we aim at leveraging a large amount of sensor data that do not require human labeling, as well as deep learning techniques, to develop visuomotor driving models by applying human behavior cloning (BC), which is a type of imitation learning (IL).

In Chapter 2, we propose to fuse depth (D) and appearance (RGB) information as input for end-to-end conditional imitation learning (CIL) models, where the depth maps can be obtained either from active sensors such as LiDAR or from a trained monocular depth estimation model. We perform experiments based on single-modal models (*i.e.*, the input is either RGB images or depth maps) and multimodal models (*i.e.*, RGBD being their input) on the CARLA simulation environment. As for the multimodal model, we train models with different fusion schemes, namely, early, mid, and late. The obtained results clearly lead us to conclude that multimodality, no matter in which fusion scheme outperforms the single-modality. Moreover, early fusion outperforms mid and late fusion schemes.

In Chapter 3, we propose a two-step training protocol that leverages end-to-end models (such as BC) for learning a representation of driving knowledge to develop driving models based on affordances. The training data required for learning a representation can be easily obtained by simply recording the actions of drivers, thus, reducing the cost of human labeling. The results show that an affordance model trained on BC pre-training performs better than using random initialization, and even better than using the widespread supervised pre-training on ImageNet. The presented results also lead us to conclude that, compared to the data collected by random roaming, expert driving data (*i.e.*, coming from human drivers) is an important source for good representation learning. Moreover, to some extent, affordance models can improve the interpretability of driving actions compared to pure end-to-end models.

In Chapter 4, we propose a model named CIL++, which aims at being a new strong baseline for pure vision-based end-to-end autonomous driving models trained by imitation learning. Compared to the sub-optimal pioneer CIL model, we enriched CIL++ with some new elements, *e.g.*, setting up three cameras (views) to reach an HFOV=180°, and using a more realistic expert driver to collect on-board data in CARLA simulator. In addition, we propose to use a visual transformer that acts as a mid-level attention mechanism to associate feature map patches (tokens) across different views. Our experimental results in the CARLA simulator show that CIL++ performs at an expert level on NoCrash metrics (single-lane towns) and obtains competitive results in complex (multi-lane) towns when compared to those models that require pixel-wise labeled data for their training.

5.2 Behind the scenes

Overall, all this work aims at improving the stability and performance of pure vision-based end-to-end autonomous driving models. For such a data-driven training approach, one of the advantages is that of being possible to easily collect petabytes of onboard data (raw sensor data, vehicle state variables, *etc*). However, it also brings a series of challenges, a key one of which is: *how to prepare suitable datasets to train models for good driving performance?* We found that beyond the architecture and methodology, an easily overlooked consideration to deliver good driving performance is the training dataset. Usually, the following aspects are very important:

- **Image Data Diversity.** As we all know, the generalization problem has always been a difficult problem in deep learning models. Some work [19, 75, 196] have already shown that using diverse data can improve the stability or/and performance of models to some extent. This is why the training of models is always based on some large pre-training proposals, such as ImageNet [45] for images, BERT [46] for languages. Thus, in order to improve the stability of driving models, we always consider increasing the diversity of our data appearance, including adding different kinds of weather, towns, obstacles, and even the density of dynamic obstacles (*i.e.*, the number of pedestrians and vehicles). We believe that having a diverse dataset is an important prerequisite for training good-performing end-to-end driving models.

- **Image Data Configuration.** Compared to pioneer CIL, the data configuration in CIL++ has been changed, both in resolution (from 200×88 images captured from one central camera to 900×300 images captured from three cameras) and HFOV (from 100° to 180°). These changes have been possible during the last period of my Ph.D., due to the improvement in computing resources in our research team. The presented results lead us to conclude that these settings are essential to improve the performance of driving models. Due to the limitation of experimental time, this thesis cannot exhaustively search more hyperparameters of CIL++, such as using multiple frames to capture temporal information, applying augmentation such as cropping and adding noise, *etc*, which we leave as part of future work.

- **Signal Data Distribution.** CIL models directly learn how to drive from the demonstrations coming from expert drivers, thus their performance can be vulnerable to corner cases that are rarely presented in the training dataset. The distribution statistics of signal data used as training supervision is crucial, such as 1) balancing the proportion of high-level commands (*i.e.*, going straight, turning left/right, lane following) since it has a great impact on the steering angle distribution; 2) collecting as many driving scenarios as possible that lead to different driving behaviors, such as suddenly stopping, following a leading vehicle at a certain speed, avoiding a collision with a pedestrian rushing into the lane; 3) using several expert drivers who provide different driving policies; 4) adding artificial disturbance or applying techniques like Dagger during data collection to improve the models' ability to handle unseen scenes and ease the long-tail cases.

In addition to the training dataset, another concern is how to efficiently come across the limited computational resources characteristic of academic environments. For instance, at the beginning of my Ph.D., the available resources for training and testing models were four NVIDIA GEFORCE RTX 2080 (the latest-generation GPU at that time). Training all models that are mentioned in Chapter 2 (eight types in total) requires ten days of full-time usage of these GPUs, not to mention the time spent on the upfront hyperparameters searching, and the online driving test on CARLA benchmarks. Therefore, training protocol and model architecture need to be reasonably designed considering the available computational capacity. For instance, the input images need to be down-scaled to 200×88 pixel resolution, the batch size is set to below 120, and the dimension of FC layers can not be too high, *etc.* The situation becomes more concerning when training multimodal models that require more memory than single-modal models, due to the increase of model parameters.

Moreover, to assess variability in performance, all the models need to be trained and tested several times. Specifically, this variability comes from two sources: 1) Due to training convergence of the underlying CNNs, we Follow standard machine learning good practices to train each model five times and select the best in each case based on the offline validation; 2) Due to different circumstances of each driving on CARLA benchmark, for each model, we run for three times and report the mean and standard deviation of the success rate. In fact, the benchmark already segregates the most important sources of variability for a better understanding of the performance of a given model: four environment conditions (training and new, town and weather), and four traffic conditions (one straight, making one turn, navigation, navigation with dynamics). In other words, one single model is tested in 16 different situations, and each of those tests over either 100 or 50 driving runs, depending on the environmental condition. Completing the benchmark for one single model takes around 10 to 12 hours on a single GPU.

After that, due to the increase in the number and capacity of GPUs in our research group, some sub-optimal conditions for model training have been improved, allowing for more extensive research. We can try more hyperparameters searching, such as using larger batch sizes, scaling up the resolution of RGB input, as well as adding more layers, or even using a visual transformer model that requires considerable computing power at training time. Nevertheless, regarding the time required for training, it is still challenging to explore large-capacity deep models. For instance, training a CIL++ model in Chapter 4 requires around 5 days of full-time usage of an NVIDIA A40 (a new-generation powerful GPU with a memory size of 48 GB). In fact, for some large models, the computational resource is still the main bottleneck. For instance, the ViT [49] needs to be trained by a standard cloud TPUv3 with 8 cores for approximately 30 days, which is huge and beyond the computational capacities of many groups.

While developing the research in this Ph.D., the CARLA simulator has been constantly changing for improvements. From Chapter 2 to Chapter 4, the driving test for models is always carried out on the latest benchmark at that time, ranging from the original CARLA benchmark to the later NoCrash and Leaderboard. In the meantime, dealing with the compatibility of different CARLA versions has been another inevitable ordeal in the research process. For one thing, the hardware and software required for installation and compilation may change; and for another, comparing model results with peers can be complicated due to the experimental environment of different CARLA versions and benchmarks. For maximum fairness, in our work, we always provide clear baselines, and compare our models with those most relevant SOTA models which are tested in the same or close versions.

5.3 Future work

Delivering autonomous vehicles requires extensive efforts, ranging from obtaining the permission of relevant departments and passing relevant legal procedures to preparing capital costs, *etc*. Due to the constraints of human resources and equipment costs, all of the experiments in this thesis were performed on the CARLA simulator. For future work, we are interested in conducting driving experiments of our CIL++ model in the physical world. To evaluate the model, a qualitative metric and systematic tests will be designed, considering various weather and driving route conditions. Apart from this, based on the existing computing resources, we believe it is also worth exploring the multi-frame input setting for CIL++, with the intuition that the temporal information is somehow useful for taking driving decisions. Another possible direction to extend this work is to train a monocular depth estimation module to generate depth maps for real car driving. From the conclusion in Section 2.5, we believe that with the assistance of depth information, the performance of our current CIL++ model can be further boosted and achieve better driving performance, even compared to other models that require costly human-labeled data for training.

Publications

1. **Yi Xiao**, Felipe Codevilla, Akhil Gurram, Onay Urfalioglu, Antonio M. López. *Multimodal End-to-End Autonomous Driving.* IEEE Transactions on Intelligent Transportation Systems (IEEE T-ITS), 2019.

2. **Yi Xiao**, Felipe Codevilla, Christopher Pal, Antonio M. López. *Action-based Representation Learning for Autonomous Driving.* Conference on Robot Learning (CoRL), 2020.

3. **Yi Xiao**, Felipe Codevilla, Diego Porres, Antonio M. López. *Scaling Vision-based End-to-End Autonomous Driving with Multi-View Attention Learning.* Under review.

Bibliography

[1] Pulkit Agrawal, Ashvin V Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[2] Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. In *Inter. Conf. on Learning Representation (ICLR) Workshops*, 2017.

[3] Alexander Amini, Wilko Schwarting, Guy Rosman, Brandon Araki, Sertac Karaman, and Daniela Rus. Variational autoencoder for end-to-end control of autonomous driving with novelty detection and training de-biasing. In *Inter. Conf. on Intelligent Robots and Systems (IROS)*, 2018.

[4] Ankesh Anand, Evan Racah, Sherjil Ozair, Yoshua Bengio, Marc-Alexandre Côté, and R Devon Hjelm. Unsupervised state representation learning in Atari. In *Advances in Neural Information Processing Systems (NIPS)*, 2019.

[5] David C Andrade, Felipe Bueno, Felipe R Franco, Rodrigo Adamshuk Silva, João Henrique Z Neme, Erick Margraf, William T Omoto, Felipe A Farinelli, Angelo M Tusset, Sergio Okida, et al. A novel strategy for road lane detection and tracking based on a vehicle's forward monocular camera. In *IEEE Trans. on Intelligent Transportation Systems*, 2018.

[6] Szilárd Aradi. Survey of deep reinforcement learning for motion planning of autonomous vehicles. *IEEE Trans. on Intelligent Transportation Systems*, 2020.

[7] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[8] Anurag Arnab and Philip HS Torr. Pixelwise instance segmentation with a dynamically instantiated network. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[9] Eduardo Arnold, Omar Y. Al-Jarrah, Mehrdad Dianati, Saber Fallah, David Oxtoby, and Alex Mouzakitis. A survey on 3D object detection methods for autonomous driving applications. *IEEE Trans. on Intelligent Transportation Systems*, January 2019.

[10] Alireza Asvadi, Luis Garrote, Cristiano Premebida, Paulo Peixoto, and Urbano J. Nunes. Multimodal vehicle detection: fusing 3D-LIDAR and color camera data. *Pattern Recognition Letters*, 115:20–29, November 2018.

[11] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization, 2016.

[12] Hernán Badino, Uwe Franke, and David Pfeiffer. The stixel world - a compact medium level representation of the 3D-world. In *DAGM: Joint Pattern Recognition Symposium*, 2009.

[13] M. Bai and R. Urtasun. Deep watershed transform for instance segmentation. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[14] Mayank Bansal, Alex Krizhevsky, and Abhijit S. Ogale. ChauffeurNet: Learning to drive by imitating the best and synthesizing the worst. In *Robotics: Science and Systems (RSS)*, 2019.

[15] RichardE Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, 6(5):679–684, 1957.

[16] Rodrigo Benenson, Timofte Radu, and Luc Van Gool. Stixels estimation without depth map computation. In *Inter. Conf. on Computer Vision (ICCV) CVVT Workshop*, 2009.

[17] Alex Bewley, Jessica Rigley, Yuxuan Liu, Jeffrey Hawke, Richard Shen, Vinh-Dieu Lam, and Alex Kendall. Learning to drive from simulation without real world labels. In *Inter. Conf. on Robotics and Automation (ICRA)*, 2019.

[18] Goutam Bhat, Joakim Johnander, Martin Danelljan, Fahad Shahbaz Khan, and Michael Felsberg. Unveiling the power of deep tracking. In *European Conf. on Computer Vision (ECCV)*, 2018.

[19] Yijun Bian and Huanhuan Chen. A survey of end-to-end driving: Architectures and training methods. *IEEE Trans. on Cybernetics*, 52(9):9059–9075, 2022.

[20] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. arXiv:1604.07316, 2016.

[21] Mariusz Bojarski, Philip Yeres, Anna Choromanska, Krzysztof Choromanski, Bernhard Firner, Lawrence Jackel, and Urs Muller. Explaining how a deep neural network trained with end-to-end learning steers a car. arXiv:1704.07911, 2017.

[22] Guillaume Bresson, Zayed Alsayed, Li Yu, and Sebastien Glaser. Simultaneous localization and mapping: A survey of current trends in autonomous driving. *IEEE Trans. on Intelligent Vehicles*, 2(3):194–220, 2017.

[23] Fabian Brickwedde, Steffen Abraham, and Rudolf Mester. Mono-stixels: Monocular depth reconstruction of dynamic street scenes. In *Inter. Conf. on Robotics and Automation (ICRA)*, 2018.

[24] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[25] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teuliere, and T. Chateau. Deep MANTA: A coarse-to-fine many-task network for joint 2D and 3D vehicle analysis from monocular image. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[26] Chenyi Chen, Ari Seff, Alain L. Kornhauser, and Jianxiong Xiao. DeepDriving: Learning affordance for direct perception in autonomous driving. In *Inter. Conf. on Computer Vision (ICCV)*, 2015.

[27] Dian Chen and Philipp Krähenbühl. Learning from all vehicles. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[28] Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In *Conf. on Robot Learning (CoRL)*, 2019.

[29] Jianyu Chen, Shengbo Eben Li, and Masayoshi Tomizuka. Interpretable end-to-end urban autonomous driving with latent deep reinforcement learning. *IEEE Trans. on Intelligent Transportation Systems*, 2021.

[30] Jianyu Chen, Zining Wang, and Masayoshi Tomizuka. Deep hierarchical reinforcement learning for autonomous driving with distinct behaviors. In *Intelligent Vehicles Symposium (IV)*, 2018.

[31] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Inter. Conf. on Machine Learning (ICML)*, 2020.

[32] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3D object detection network for autonomous driving. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[33] Xieyuanli Chen, Hui Zhang, Huimin Lu, Junhao Xiao, Qihang Qiu, and Yi Li. Robust SLAM system based on monocular vision and LiDAR for robotic urban search and rescue. In *Safety, Security and Rescue Robotics (SSRR)*, 2017.

[34] Zhilu Chen and Xinming Huang. End-to-end learning for lane keeping of self-driving cars. In *Intelligent Vehicles Symposium (IV)*, 2017.

[35] Kashyap Chitta, Aditya Prakash, and Andreas Geiger. NEAT: Neural attention fields for end-to-end autonomous driving. In *Inter. Conf. on Computer Vision (ICCV)*, 2021.

[36] Wongun Choi. Near-online multi-target tracking with aggregated local flow descriptor. In *Inter. Conf. on Computer Vision (ICCV)*, 2015.

[37] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems (NIPS)*, 2018.

[38] Felipe Codevilla, Antonio M. López, Vladlen Koltun, and Alexey Dosovitskiy. On offline evaluation of vision-based driving models. In *European Conf. on Computer Vision (ECCV)*, 2018.

[39] Felipe Codevilla, Matthias Müller, Antonio M. López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In *Inter. Conf. on Robotics and Automation (ICRA)*, 2018.

[40] Felipe Codevilla, Edgar Santana, Antonio M. López, and Adrien Gaidon. Exploring the limitations of behavior cloning for autonomous driving. In *Inter. Conf. on Computer Vision (ICCV)*, 2019.

[41] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes dataset for semantic urban scene understanding. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[42] Marius Cordts, Timo Rehfeld, Lukas Schneider, David Pfeiffer, Markus Enzweiler, Stefan Roth, Marc Pollefeys, and Uwe Franke. The stixel world: A medium-level representation of traffic scenes. *Image and Vision Computing (IV)*, 68:40–52, December 2017.

[43] Xiaohui Dai, Chi-Kwong Li, and Ahmad B Rad. An approach to tune fuzzy controllers based on reinforcement learning for autonomous vehicle control. *IEEE Trans. on Intelligent Transportation Systems*, 6(3):285–293, 2005.

[44] Pim De Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. In *Conf. on Neural Information Processing Systems (NeurIPS)*, 2019.

[45] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[46] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805, 2019.

[47] Martin Dimitrievski, Peter Veelaert, and Wilfried Philips. Behavioral pedestrian tracking using a camera and LiDAR sensors on a moving vehicle. *Sensors*, 19(2), 2019.

[48] Carl Doersch and Andrew Zisserman. Multi-task selfsupervised visual learning. In *Inter. Conf. on Computer Vision (ICCV)*, 2017.

[49] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Inter. Conf. on Learning Representation (ICLR)*, 2021.

[50] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio López, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Conf. on Robot Learning (CoRL)*, 2017.

[51] Markus Enzweiler and Dariu M. Gavrila. A multilevel mixture-of-experts framework for pedestrian classification. *IEEE Trans. on Image Processing*, 20(10):2967–2979, 2011.

[52] Hesham M. Eraqi, Mohamed N. Moustafa, and Jens Honer. End-to-end deep learning for steering autonomous vehicles considering temporal dependencies. In *Advances in Neural Information Processing Systems (NIPS)ML for ITS WS*, 2017.

[53] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R Qi, Yin Zhou, et al. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In *Inter. Conf. on Computer Vision (ICCV)*, 2021.

[54] Mingyuan Fan, Shenqi Lai, Junshi Huang, Xiaoming Wei, Zhenhua Chai, Junfeng Luo, and Xiaolin Wei. Rethinking bisenet for real-time semantic segmentation. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[55] Di Feng, Christian Haase-Schütz, Lars Rosenbaum, Heinz Hertlein, Claudius Glaeser, Fabian Timm, Werner Wiesbeck, and Klaus Dietmayer. Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges. *IEEE Trans. on Intelligent Transportation Systems*, 22(3):1341–1360, 2020.

[56] Zeyu Feng, Chang Xu, and Dacheng Tao. Self-supervised representation learning by rotation feature decoupling. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[57] Uwe Franke. Autonomous driving. In *Computer Vision in Vehicle Technology*. 2017.

[58] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao. Deep ordinal regression network for monocular depth estimation. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[59] Yukang Gan, Xiangyu Xu, Wenxiu Sun, and Liang Lin. Monocular depth estimation with affinity, vertical pooling, and label enhancement. In *European Conf. on Computer Vision (ECCV)*, 2018.

[60] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[61] Laurent George, Thibault Buhet, Emilie Wirbel, Gaetan Le-Gall, and Xavier Perrotton. Imitation learning for end to end vehicle longitudinal control with forward camera. In *Advances in Neural Information Processing Systems (NIPS) Imitation Learning WS*, 2018.

[62] James J Gibson. *The ecological approach to visual perception: classic edition*. Psychology Press, 2014.

[63] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *Inter. Conf. on Learning Representation (ICLR)*, 2018.

[64] C. Godard, O.M. Aodha, and G.J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[65] Alejandro González, Zhijie Fang, Yainuvis Socarras, Joan Serrat, David Vázquez, Jiaolong Xu, and Antonio M. López. Pedestrian detection at day/night time with visible and fir cameras: A comparison. *Sensors*, 16(6), 2016.

[66] Alejandro González, David Vázquez, Antonio M. López, and Jaume Amores. On-board object detection: Multicue, multimodal, and multiview random forest of local experts. *IEEE Trans. on Cybernetics*, 47(11):3980–3990, 2017.

[67] Priya Goyal, Dhruv Mahajan, Abhinav Gupta, and Ishan Misra. Scaling and benchmarking self-supervised visual representation learning. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[68] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3):362–386, 2020.

[69] Rui Guo, Jianbo Liu, Na Li, Shibin Liu, Fu Chen, Bo Cheng, Jianbo Duan, Xinpeng Li, and Caihong Ma. Pixel-wise classification method for high resolution remote sensing imagery using deep neural networks. *ISPRS International Journal of Geo-Information*, 7(3):110, 2018.

[70] A. Gurram, O. Urfalioglu, I. Halfaoui, F. Bouzaraa, and Antonio M. Lopez. Monocular depth estimation by learning from heterogeneous datasets. In *Intelligent Vehicles Symposium (IV)*, 2018.

[71] Qishen Ha, Kohei Watanabe, Takumi Karasawa, Yoshitaka Ushiku, and Tatsuya Harada. MFNet: Towards real-time semantic segmentation for autonomous vehicles with multi-spectral scenes. In *Inter. Conf. on Intelligent Robots and Systems (IROS)*, 2017.

[72] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[73] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *Inter. Conf. on Computer Vision (ICCV)*, 2019.

[74] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[75] Kaiming He, Ross Girshick, and Piotr Dollár. Rethinking imagenet pretraining. arXiv:1811.08883, 2019.

[76] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[77] Simon Hecker, Dengxin Dai, and Luc Van Gool. End-to-end learning of driving models with surround-view cameras and route planners. In *European Conf. on Computer Vision (ECCV)*, 2018.

[78] Daniel Hernández-Juarez, Lukas Schneider, Antonio Espinosa, David Vázquez, Antonio M. López, Uwe Franke, Marc Pollefeys, and Juan C. Moure. Slanted Stixels: Representing San Francisco's steepest streets. In *British Machine Vision Conference (BMVC)*, 2017.

[79] Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Patwary, Mostofa Ali, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. arXiv:1712.00409, 2017.

[80] Anthony Hu, Gianluca Corrado, Nicolas Griffiths, Zak Murez, Corina Gurau, Hudson Yeo, Alex Kendall, Roberto Cipolla, and Jamie Shotton. Model-based imitation learning for urban driving. In *Conf. on Neural Information Processing Systems (NeurIPS)*, 2022.

[81] Xinyu Huang, Xinjing Cheng, Qichuan Geng, Binbin Cao, Dingfu Zhou, Peng Wang, Yuanqing Lin, and Ruigang Yang. The apolloscape dataset for autonomous driving. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR) Workshop*, 2018.

[82] Christian Hubschneider, Andre Bauer, Michael Weber, and J. Marius Zollner. Adding navigation to the equation: Turning decisions for end-to-end vehicle control. In *Intelligent Transportation Systems Conference (ITSC) Workshops*, 2017.

[83] Christopher Innocenti, Henrik Lindén, Ghazaleh Panahandeh, Lennart Svensson, and Nasser Mohammadiha. Imitation learning for vision-based lane keeping assistance. In *Intelligent Transportation Systems Conference (ITSC)*, 2017.

[84] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Inter. Conf. on Machine Learning (ICML)*, 2015.

[85] Joel Janai, Fatma Güney, Aseem Behl, Andreas Geiger, et al. Computer vision for autonomous vehicles: Problems, datasets and state of the art. *Foundations and Trends in Computer Graphics and Vision*, 12(1–3):1–308, 2020.

[86] Maximilian Jaritz, Raoul de Charette, Marin Toromanoff, Etienne Perot, and Fawzi Nashashibi. End-to-end race driving with deep reinforcement learning. In *Inter. Conf. on Robotics and Automation (ICRA)*, 2018.

[87] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conf. on Robot Learning (CoRL)*, 2018.

[88] Tsung-Wei Ke, Jyh-Jing Hwang, and Stella X Yu. Universal weakly supervised segmentation by pixel-to-segment contrastive learning. arXiv:2105.00957, 2021.

[89] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[90] Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. Learning to drive in a day. In *Inter. Conf. on Robotics and Automation (ICRA)*, 2019.

[91] Hoel Kervadec, Jose Dolz, Shanshan Wang, Eric Granger, and Ismail Ben Ayed. Bounding boxes for weakly supervised segmentation: Global constraints get close to full supervision. In *Medical Imaging with Deep Learning*, 2020.

[92] Qadeer Khan, Torsten Schön, and Patrick Wenzel. Towards self-supervised high level sensor fusion. arXiv:1902.04272, 2019.

[93] Jinkyu Kim and John Canny. Interpretable learning for self-driving cars by visualizing causal attention. In *Inter. Conf. on Computer Vision (ICCV)*, 2017.

[94] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Inter. Conf. on Learning Representation (ICLR)*, 2015.

[95] Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. Revisiting self-supervised visual representation learning. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[96] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. Waslander. Joint 3D proposal generation and object detection from view aggregation. In *Inter. Conf. on Intelligent Robots and Systems (IROS)*, 2018.

[97] Guillaume Lample and Devendra Singh Chaplot. Playing fps games with deep reinforcement learning. In *Conf. on Artificial Intelligence (AAAI)*, 2017.

[98] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[99] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Colorization as a proxy task for visual understanding. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[100] Yann LeCun, Urs Muller, Jan Ben, Eric Cosatto, and Beat Flepp. Off-road obstacle avoidance through end-to-end learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2005.

[101] Chengyang Li, Dan Song, Ruofeng Tong, and Min Tang. Illumination-aware Faster R-CNN for robust multispectral pedestrian detection. *Pattern Recognition*, 85:161–171, January 2019.

[102] L. Li, Z. Liu, O. Ozgüner, J. Lian, Y. Zhou, and Y. Zhao. Dense 3D semantic SLAM of traffic environment based on stereo vision. In *Intelligent Vehicles Symposium (IV)*, 2018.

[103] Peiliang Li, Tong Qin, et al. Stereo vision-based semantic 3d object and ego-motion tracking for autonomous driving. In *European Conf. on Computer Vision (ECCV)*, 2018.

[104] Zhihao Li, Toshiyuki Motoyoshi, Kazuma Sasaki, Tetsuya Ogata, and Shigeki Sugano. Rethinking self-driving: Multi-task knowledge for better generalization and accident explanation ability. arXiv:1809.11100, 2018.

[105] Xiaodan Liang, Tairui Wang, Luona Yang, and Eric Xing. CIRL: Controllable imitative reinforcement learning for vision-based self-driving. In *European Conf. on Computer Vision (ECCV)*, 2018.

[106] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2015.

[107] S. Liu, J. Jia, S. Fidle, and R. Urtasun. SGN: sequential grouping networks for instance segmentation. In *Inter. Conf. on Computer Vision (ICCV)*, 2017.

[108] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[109] Tianrui Liu and Tania Stathaki. Faster r-cnn for robust pedestrian detection using semantic segmentation network. *Frontiers in neurorobotics*, 12:64, 2018.

[110] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. Berg. SSD: single shot multibox detector. In *European Conf. on Computer Vision (ECCV)*, 2016.

[111] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[112] Yue Luo, Jimmy Ren, Mude Lin, Jiahao Pang, Wenxiu Sun, Hongsheng Li, and Liang Lin. Single view stereo matching. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[113] W. Maddern, G. Pascoe, C. Linegar, and P. Newman. 1 Year, 1000 Km: The Oxford RobotCar dataset. *Inter. Journal of Robotics Research (IJRR)*, 36(1):3–15, 2017.

[114] Ishan Misra, C Lawrence Zitnick, and Martial Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *European Conf. on Computer Vision (ECCV)*, 2016.

[115] Dennis Mitzel and Bastian Leibe. Taking mobile multi-object tracking to the next level: People, unknown objects, and carried items. In *European Conf. on Computer Vision (ECCV)*, 2012.

[116] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.

[117] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka. 3D bounding box estimation using deep learning and geometry. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[118] Matthias Müller, Alexey Dosovitskiy, Bernard Ghanem, and Vladen Koltun. Driving policy transfer via modularity and abstraction. In *Conf. on Robot Learning (CoRL)*, 2018.

[119] Davy Neven, Bert De Brabandere, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Towards end-to-end lane detection: an instance segmentation approach. In *Intelligent Vehicles Symposium (IV)*, 2018.

[120] Andrew Y Ng, Stuart Russell, et al. Algorithms for inverse reinforcement learning. In *Inter. Conf. on Machine Learning (ICML)*, 2000.

[121] Hai Nguyen and Hung La. Review of deep reinforcement learning for robot manipulation. In *Inter. Conf. on Robotic Computing (IRC)*, 2019.

[122] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *Inter. Conf. on Computer Vision (ICCV)*, 2015.

[123] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conf. on Computer Vision (ECCV)*, 2016.

[124] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. arXiv:1807.03748, 2018.

[125] Brian Paden, Michal Cáp, Sze Zheng Yong, Dmitry S. Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Trans. on Intelligent Vehicles*, 1(1):33–55, 2016.

[126] Yunpeng Pan, Ching-An Cheng, Kamil Saigol, Keuntaek Lee, Xinyan Yan, Evangelos A Theodorou, and Byron Boots. Agile autonomous driving using end-to-end deep imitation learning. In *Robotics: Science and Systems (RSS)*, 2018.

[127] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2017.

[128] Etienne Perot, Maximilian Jaritz, Marin Toromanoff, and Raoul de Charette. End-to-end driving in a realistic racing game with deep reinforcement learning. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2017.

[129] Andreas Pfeuffer and Klaus Dietmayer. Optimal sensor data fusion architecture for object detection in adverse weather conditions. In *Inter. Conf. on Information Fusion (FUSION)*, 2018.

[130] Florian Piewak, Peter Pinggera, Markus Enzweiler, David Pfeiffer, and Marius Zöllner. Improved semantic stixels via multimodal sensor fusion. In *German Conf. on Pattern Recognition (GCPR)*, 2018.

[131] Dean Pomerleau. ALVINN: An autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems (NIPS)*, 1989.

[132] Daniel Ponsa, Antonio M. López, Joan Serrat, Felipe Lumbreras, and Thorsten Graf. Multiple vehicle 3D tracking using an unscented kalman. In *Intelligent Transportation Systems Conference (ITSC)*, 2005.

[133] Aditya Prakash, Kashyap Chitta, and Andreas Geiger. Multi-modal fusion transformer for end-to-end autonomous driving. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[134] C. Premebida, J. Carreira, J. Batista, and U. Nunes. Pedestrian detection combining rgb and dense LiDAR. In *Inter. Conf. on Intelligent Robots and Systems (IROS)*, 2014.

[135] Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum PointNets for 3D object detection from RGB-D data. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[136] K. Qiu, Y. Ai, B. Tian, B. Wang, and D. Ca. Siamese-ResNet: implementing loop closure detection based on siamese network. In *Intelligent Vehicles Symposium (IV)*, 2018.

[137] Ahmed H Qureshi, Anthony Simeonov, Mayur J Bency, and Michael C Yip. Motion planning networks. In *Inter. Conf. on Robotics and Automation (ICRA)*, 2019.

[138] Nathan D. Ratliff, James A. Bagnell, and Siddhartha S. Srinivasa. Imitation learning for locomotion and manipulation. In *Inter. Conf. on Humanoid Robots (HUMANOIDS)*, 2007.

[139] J. Redmon and A. Farhadi. YOLO9000: better, faster, stronger. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[140] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[141] Nicholas Rhinehart, Rowan McAllister, and Sergey Levine. Deep imitative models for flexible inference, planning, and control. arXiv:1810.06544, 2018.

[142] G. Ros, A.D. Sappa, D. Ponsa, and A.M. López. Visual SLAM for driverless cars: A brief survey. In *Intelligent Vehicles Symposium (IV) Workshops*, 2012.

[143] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Inter. Conf. on Artificial Intelligence and Statistics (AISTATS)*, 2010.

[144] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Inter. Conf. on Artificial Intelligence and Statistics (AISTATS)*, 2011.

[145] Saumya Kumaar Saksena, B Navaneethkrishnan, Sinchana Hegde, Pragadeesh Raja, and Ravi M Vishwanath. Towards behavioural cloning for autonomous driving. In *Inter. Conf. on Robotic Computing (IRC)*, 2019.

[146] Ahmad El Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. End-to-end deep reinforcement learning for lane keeping assist. arXiv:1612.04340, 2016.

[147] Caude Sammut. *Behavioral Cloning*. Springer US, 2010.

[148] Claude Sammut, Scott Hurst, Dana Kedzier, and Donald Michie. Learning to fly. In *Machine Learning Proceedings 1992*, pages 385–393. Elsevier, 1992.

[149] Eder Santana and George Hotz. Learning a driving simulator. arXiv:1608.01230, 2016.

[150] Axel Sauer, Nikolay Savinov, and Andreas Geiger. Conditional affordance learning for driving in urban environments. In *Conf. on Robot Learning (CoRL)*, 2018.

[151] Lukas Schneider, Marius Cordts, Timo Rehfeld, David Pfeiffer, Markus Enzweiler, Uwe Franke, Marc Pollefeys, and Stefan Roth. Semantic stixels: Depth is not enough. In *Intelligent Vehicles Symposium (IV)*, 2016.

[152] Lukas Schneider, Manuel Jasch, Björn Fröhlich, Thomas Weber, Uwe Franke, Marc Pollefeys, and Matthias Rätsch. Multimodal neural networks: RGB-D for semantic segmentation and object detection. In *Scandinavian Conf. on Image Analysis (SCIA)*, 2017.

[153] W. Schwarting, J. Alonso, and D. Rus. Planning and decision-making for autonomous vehicles. *Annual Reviews of Control, Robotics, and Autonomous Systems*, 1:187–210, May 2018.

[154] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Inter. Conf. on Computer Vision (ICCV)*, 2017.

[155] Hao Shao, Letian Wang, Ruobing Chen, Hongsheng Li, and Yu Liu. InterFuser: Safety-enhanced autonomous driving using interpretable sensor fusion transformer. In *Conf. on Robot Learning (CoRL)*, 2022.

[156] Sarthak Sharma, Junaid Ahmed Ansari, J. Krishna Murthy, and K. Madhava Krishna. Beyond pixels: Leveraging geometry and shape cues for online multi-object tracking. In *Inter. Conf. on Robotics and Automation (ICRA)*, 2018.

[157] Evan Shelhamer, Parsa Mahmoudieh, Max Argus, and Trevor Darrell. Loss is its own reward: Self-supervision for reinforcement learning. In *Inter. Conf. on Learning Representation (ICLR)*, 2017.

[158] Young-Sik Shin, Yeong Sang Park, and Ayoung Kim. Direct visual SLAM using sparse depth for camera-LiDAR system. In *Inter. Conf. on Robotics and Automation (ICRA)*, 2018.

[159] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[160] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[161] Martin Simony, Stefan Milzy, Karl Amendey, and Horst-Michael Gross. Complex-yolo: An euler-region-proposal for real-time 3d object detection on point clouds. In *European Conf. on Computer Vision (ECCV) Workshops*, 2018.

[162] Ibrahim Sobh, Loay Amin, Sherif Abdelkarim, Khaled Elmadawy, Mahmoud Saeed, Omar Abdeltawab Valeo, Mostafa Gamal, and Ahmad El-Sallab. End-to-end multi-modal sensors fusion system for urban automated driving. In *Advances in Neural Information Processing Systems (NIPS) MLITS WS*, 2018.

[163] Sheng Song, Xuemin Hu, Jin Yu, Liyun Bai, and Long Chen. Learning a deep motion planning model for autonomous driving. In *Intelligent Vehicles Symposium (IV)*, 2018.

[164] Jonathan Spencer, Sanjiban Choudhury, Arun Venkatraman, Brian Ziebart, and J Andrew Bagnell. Feedback in imitation learning: The three regimes of covariate shift. arXiv:2102.02872, 2021.

[165] Simon Stepputtis, Joseph Campbell, Mariano Phielipp, Stefan Lee, Chitta Baral, and Heni Ben Amor. Advances in neural information processing systems. In *Advances in Neural Information Processing Systems (NIPS)*, 2020.

[166] C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Inter. Conf. on Computer Vision (ICCV)*, 2017.

[167] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[168] Ardi Tampuu, Tambet Matiisen, Maksym Semikin, Dmytro Fishman, and Naveed Muhammad. A survey of end-to-end driving: Architectures and training methods. *IEEE Trans. on Neural Networks and Learning Systems*, 33(4):1364–1384, 2022.

[169] Ning Tang, Fei Zhou, Zhaorui Gu, Haiyong Zheng, Zhibin Yu, and Bing Zheng. Unsupervised pixel-wise classification for chaetoceros image segmentation. *Neurocomputing*, 318:261–270, 2018.

[170] K. Tateno, F. Tombari, I. Laina, and N. Navab. CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[171] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, et al. Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics*, 23(9):661–692, 2006.

[172] Faraz Torabi, Garrett Warnell, and Peter Stone. Recent advances in imitation learning from observation. In *Inter. Joint Conf. on Artificial Intelligence (IJCAI)*, 2019.

[173] Marin Toromanoff, Emilie Wirbel, and Fabien Moutarde. End-to-end model-free reinforcement learning for urban driving using implicit affordances. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[174] J. Uhrig, M. Cordts, U. Franke, and T. Brox. Pixel-level encoding and depth layering for instance-level semantic labelling. In *German Conf. on Pattern Recognition (GCPR)*, 2016.

[175] Mohib Ullah, Ahmed Mohammed, and Faouzi Alaya Cheikh. Pednet: A spatio-temporal deep convolutional neural network for pedestrian segmentation. *Journal of Imaging*, 4(9):107, 2018.

[176] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Conf. on Neural Information Processing Systems (NeurIPS)*, 2017.

[177] Hengli Wang, Peide Cai, Yuxiang Sun, Lujia Wang, and Ming Liu. Learning interpretable end-to-end vision-based motion planning for autonomous driving with optical flow distillation. In *Inter. Conf. on Robotics and Automation (ICRA)*, 2021.

[178] Q. Wang, L. Chen, and W. Tian. End-to-end driving simulation via angle branched network. arXiv:1805.07545, 2018.

[179] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. Solov2: Dynamic and fast instance segmentation. In *Conf. on Neural Information Processing Systems (NeurIPS)*, 2020.

[180] N. Wojke, A. Bewley, and D. Paulus. Simple online and realtime tracking with a deep association metric. In *Inter. Conf. on Image Processing (ICIP)*, 2017.

[181] Penghao Wu, Xiaosong Jia, Li Chen, Junchi Yan, Hongyang Li, and Yu Qiao. Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline. In *Conf. on Neural Information Processing Systems (NeurIPS)*, 2022.

[182] Yingying Wu, Huacheng Qin, Tao Liu, Hao Liu, and Zhiqiang Wei. A 3D object detection based on multi-modality sensors of USV. *Applied Sciences*, 9(3), 2019.

[183] Bernhard Wymann, Eric Espié, Christophe Guionneau, Christos Dimitrakakis, Rémi Coulom, and Andrew Sumner. TORCS, The Open Racing Car Simulator. http://www.torcs.org.

[184] Yu Xiang, Alexandre Alahi, and Silvio Savarese. Learning to track: Online multi-object tracking by decision making. In *Inter. Conf. on Computer Vision (ICCV)*, 2015.

[185] Yi Xiao, Felipe Codevilla, Akhil Gurram, Onay Urfalioglu, and Antonio M López. Multimodal end-to-end autonomous driving. *IEEE Trans. on Intelligent Transportation Systems*, 23(1):537–547, 2020.

[186] Yi Xiao, Felipe Codevilla, Christopher Pal, and Antonio Lopez. Action-based representation learning for autonomous driving. In *Conf. on Robot Learning (CoRL)*, 2021.

[187] Dan Xu, Wei Wang, Hao Tang, Hong Liu, Nicu Sebe, and Elisa Ricci. Structured attention guided convolutional neural fields for monocular depth estimation. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[188] Huazhe Xu, Yang Gao, Fisher Yu, and Trevor Darrell. End-to-end learning of driving models from large-scale video datasets. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[189] Jiaolong Xu, Liang Xiao, and Antonio M. López. Self-supervised domain adaptation for computer vision tasks. *IEEE Access*, 7:156694–156706, October 2019.

[190] Yan Yan, Yuxing Mao, and Bo Li. SECOND: Sparsely embedded convolutional detection. *Sensors*, 18(10), 2018.

[191] B. Yang, W. Luo, and R. Urtasun. PIXOR: Real-time 3D object detection from point clouds. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[192] Zhengyuan Yang, Yixuan Zhang, Jerry Yu, Junjie Cai, and Jiebo Luo. End-to-end multi-modal multi-task vehicle control for self-driving cars with visual perceptions. In *Inter. Conf. on Pattern Recognition (ICPR)*, 2018.

[193] H. Yin, L. Tang, X. Ding, Y. Wang, and R. Xiong. LocNet: global localization in 3D point clouds for mobile vehicles. In *Intelligent Vehicles Symposium (IV)*, 2018.

[194] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *European Conf. on Computer Vision (ECCV)*, 2018.

[195] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *Inter. Conf. on Learning Representation (ICLR)*, 2016.

[196] Yu Yu, Shahram Khadivi, and Jia Xu. Can data diversity enhance learning generalization? In *Inter. Conf. on Computational Linguistics (ICCL)*, 2022.

[197] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access*, 8:58443–58469, 2020.

[198] Xinzheng Zhang, Ahmad B. Rad, and Yiu-Kwong Wong. Sensor fusion of monocular cameras and laser rangefinders for line-based simultaneous localization and mapping (SLAM) tasks in autonomous mobile robots. *Sensors*, 12(1):429–452, 2012.

[199] Zhejun Zhang, Alexander Liniger, Dengxin Dai, Fisher Yu, and Luc Van Gool. End-to-end urban driving by imitating a reinforcement learning coach. In *Inter. Conf. on Computer Vision (ICCV)*, 2021.

[200] Albert Zhao, Tong He, Yitao Liang, Haibin Huang, Guy Van den Broeck, and Stefano Soatto. Lates: Latent space distillation for teacher-student driving policy learning. arXiv:1912.02973, 2019.

[201] Y. Zhou and O. Tuzel. VoxelNet: End-to-end learning for point cloud based 3D object detection. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[202] J. Zhu, Y. Ai, B. Tian, D. Cao, and S. Scherer. Visual place recognition in long-term and large-scale environment based on CNN feature. In *Intelligent Vehicles Symposium (IV)*, 2018.

[203] Zhe Zhu, Dun Liang, Songhai Zhang, Xiaolei Huang, Baoli Li, and Shimin Hu. Traffic-sign detection and classification in the wild. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.