Exploiting Multi-Modal Fusion for Urban Autonomous Driving using Latent Deep Reinforcement Learning

by

Yasser Khalil

Thesis submitted to the Faculty of Engineering
in partial fulfillment of the requirements for the Ph.D. degree
in Electrical and Computer Engineering

School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Ontario, Canada

ABSTRACT

Human driving decisions are the leading cause of road fatalities. Autonomous driving naturally eliminates such incompetent decisions and thus can improve traffic safety and efficiency. Deep reinforcement learning (DRL) has shown great potential in learning complex tasks. Recently, researchers investigated various DRL-based approaches for autonomous driving. However, exploiting multi-modal fusion to generate pixel-wise perception and motion prediction and then leveraging these predictions to train a latent DRL has not been targeted yet. Unlike other DRL algorithms, the latent DRL algorithm distinguishes representation learning from task learning, enhancing sampling efficiency for reinforcement learning. In addition, supplying the latent DRL algorithm with accurate perception and motion prediction simplifies the surrounding urban scenes, improving training and thus learning a better driving policy. To that end, this Ph.D. research initially develops *LiCaNext*, a novel real-time multi-modal fusion network to produce accurate joint perception and motion prediction at a pixel level. Our proposed approach relies merely on a LIDAR sensor, where its multi-modal input is composed of bird's-eye view (BEV), range view (RV), and range residual images. Further, this Ph.D. thesis proposes leveraging these predictions with another simple BEV image to train a sequential latent maximum entropy reinforcement learning (MaxEnt RL) algorithm. A sequential latent model is deployed to learn a more compact latent representation from high-dimensional inputs. Subsequently, the MaxEnt RL model trains on this latent space to learn a driving policy.

The proposed *LiCaNext* is trained on the public nuScenes dataset. Results demonstrated that *LiCaNext* operates in real-time and performs better than the state-of-the-art in perception and motion prediction, especially for small and distant objects. Furthermore, simulation experiments are conducted on CARLA to evaluate the performance of our proposed approach that exploits *LiCaNext* predictions to train sequential latent MaxEnt RL algorithm. The simulated experiments manifest that our proposed approach learns a better driving policy outperforming other prevalent DRL-based algorithms. The learned driving policy achieves the objectives of safety, efficiency, and comfort. Experiments also reveal that the learned policy maintains its effectiveness under different environments and varying weather conditions.

ACKNOWLEDGMENTS

As my Ph.D. journey is drawing to an end, I would like to send my sincere acknowledgment to my supervisor, Professor Hussein Mouftah, whose immense knowledge and experience helped me construct and complete this Ph.D. thesis smoothly. Throughout my journey, Professor Mouftah was highly supportive to the point where he would continuously provide me with insightful suggestions regardless of when I sought them, whether it was in the morning, on the weekend, or even on a public holiday.

My genuine gratitude goes to my parents and siblings, who were supportive since day one of me pursuing my dreams despite the uncertainty that resulted from the outbreak of the COVID-19 pandemic. Living far from your family in such conditions, especially when your family contracts the virus twice, is a harrowing experience. I used to video chat with them almost daily, yet I still felt unsatisfactory as we were physically very distant. Studying abroad was my first experience living away from my parents and being completely independent. Indeed, this experience was one-of-a-kind that forced me to learn a lot of things the hard way, especially cooking. Here, I would like to deeply thank my mother and grandmother, who over the phone instructed me with all the recipes and tricks to prepare delicious food. Besides completing my Ph.D. thesis, I can proudly say that I accomplished another achievement: becoming a perfect cook! Finally, I would like to express my appreciation to my friends who gave me good company where we had unforgettable moments.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

Acronym	Definition
2D	Two Dimensional
3D	Three Dimensional
AMVNet	Assertion-based Multi-View Fusion Network
BEV	Bird's-Eye View
BEV-MODNet	Bird's-Eye View Moving Object Detection Network
CARLA	CAR Learning to Act
CAV	Connected Autonomous Vehicle
CNN	Convolutional Neural Network
Conv	Convolution
COVID-19	COronaVIrus Disease 2019
CPU	Central Processing Unit
DARPA	Defense Advanced Research Projects Agency
DDPG	Deep Deterministic Policy Gradient
DDQN	Double Deep Q-Network
DNN	Deep Neural Network
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
ELBO	Evidence Lower BOund
FC	Fully Connected layers
FaF	Fast and Furious
Faster R-CNN	Faster Region-based Convolutional Neural Network
FlowNet3D	Flow Network in Three Dimensional
FlyingThings3D	Flying Things Three Dimensional
FOV	Field-Of-View
FPN	Feature Pyramid Network
FuseMODNet	Fuse Moving Object Detection Network

GCN	Graphical Convolution Network
GNN	Graph Neural Network
GPS	Global Positioning System
GPU	Graphical Processing Unit
HD	High Definition
HPLFlowNet	Hierarchical Permutohedral Lattice Flow Network
Hz	Hertz
IL	Imitation Learning
IMU	Inertial Measurement Unit
INTEL	INTegrated ELectronics
IntentNet	Intent Network
KITTI	Karlsruhe Institute of Technology and Toyota Technological Institute
kNN	k-Nearest Neighbor
LaserFlow	Laser Flow
LaserNet++	Laser Network next version
LiCaNet	LIDAR Camera Network
LiCaNext	LIDAR Camera Next Network
LIDAR	LIght Detection And Ranging
LSTM	Long Short-Term Memory
LSTM-Encoder-Decoder	Long Short-Term Memory Encoder Decoder
MaxEnt	Maximum Entropy
MCA	Mean Classification Accuracy
MDP	Markov Decision Process
MobileNetv2	Mobile Network version Two
MODNet	Moving Object Detection Network
MOS	Moving Object Segmentation
MotionNet	Motion Network
MultiXNet	Multiclass Multistage Multimodal Motion prediction Network

MVFuseNet	Multi-View Fusion Network
MVX-Net	Multimodal VoXel Network
nuScenes	nuTonomy Scenes
OA	Overall Accuracy
PAN	Path Aggregation Network
PGM	Probabilistic Graphical Models
PnPNet	Perception and Prediction Network
PointPainting	Point Painting
PointRCNN	Point Region-based Convolution Neural Network
POMDP	Partially Observable Markov Decision Process
PPO	Proximal Policy Optimization
PV-RCNN	Point-Voxel Region-based Convolutional Neural Network
RADAR	RAdio Detection And Ranging
R-CNN	Region-based Convolutional Neural Network
ResNet	Residual Network
ResNeXt	Residual Next Network
R-FCN	Region-based Fully Convolutional Networks
RGB	Red Green Blue
RL	Reinforcement Learning
RNN	Recurrent Neural Network
RST-MODNet	Real-time Spatio-Temporal Moving Object Detection Network
RSU	Roadside Unit
RTX	Ray Tracing Texel eXtreme
RV	Range View
RV-FuseNet	Range View based Fusion Network
SAC	Soft Actor-Critic
SAE	Society of Automotive Engineers
SalsaNext	SegmentAtion Next Network

SE	Special Euclidean group
SGD	Stochastic Gradient Descent
SpAGNN	Spatially-Aware Graph Neural Network
SqueezeSegV3	Squeeze Segmentation version Three
SSD	Single-Shot Detector
STC	Spatio-Temporal Convolution
STPN	Spatio-Temporal Pyramid Network
SUMO	Simulation of Urban MObility
TD3	Twin Delayed Deep Deterministic Policy Gradient
TensorRT	Tensor Real-Time
TORCS	The Open Racing Car Simulator
U-Net	U-Network
UPSNet	Unified Panoptic Segmentation Network
VGG	Visual Geometry Group
VoxelNet	Voxel Network
WHO	World Health Organization
WOD	Waymo Open Dataset
YOLO	You Only Look Once
YOLOv4-5D	You Only Look Once version Four - Five scale Detections

LIST OF SYMBOLS

Symbol	Definition
(i, j)	Pixel coordinate
(i', j')	Pixel coordinate of next frame
(l_{hlcf}, w_{hlcf})	Length and width of the high-level camera features
(u_C, v_C)	Angular coordinates mapped on camera image
$(u_{C_{scaled}}, v_{C_{scaled}})$	Scaled angular coordinates mapped on camera image
(u_{RV}, v_{RV})	Angular coordinates of RV image
$(\hat{x}, \hat{y}, \hat{z})$	Coordinates of LIDAR point $\hat{p}_i$ in xyz-axis
μ	Mean of Gaussian layer
a	Action
$\mathcal{A}$	Action space
a'	Next action
$a_{1:T}$	Actions from time t to T
$a_{1:\tau}$	Actions from time 1 to τ
Adv	Advantage function
a_t	Action at time t
a_τ	Action at time τ
$a_{\tau+1}$	Action at time $\tau+1$
$a_{\tau+1:T}$	Actions from time $\tau+1$ to T
bev_{dim}	Number of BEV dimensions
$buffer_{size}$	Size of replay buffer
b_α	Balancing factor for $\mathcal{L}_s$
b_β	Balancing factor for $\mathcal{L}_{ft}$
b_γ	Balancing factor for $\mathcal{L}_{bt}$
C	Camera
ch_{RV}	Channels of RV image
$Class_j$	Category class j

$count$	Counter
$dist$	Distance from lane center
$\mathcal{D}$	Replay buffer
D_{KL}	Kullback-Leibler divergence
$\mathcal{E}$	MDP framework
f	LIDAR vertical field-of-view
f_{down}	Lower-part of LIDAR vertical FOV
f_{up}	Upper-part of LIDAR vertical FOV
G	Return
h_{bev}	Height dimension of BEV image
$head_{max}$	Maximum allowed $head_{sim}$
$head_{sim}$	Heading cosine similarity
$image_k$	Image k in dataset $Kimages$
J_M	Environmental objective function
J_Q	Critic loss function
J_π	Policy loss function
$\mathcal{K}$	Camera intrinsic calibration matrix
$Kimages$	Total number of images in a dataset
L	LIDAR
$laneCenter_{max}$	Maximum allowed distance from lane center
L_{BEV}	Projected LIDAR feature into BEV form
l_{bev}	Length dimension of BEV image
l_C	Length of camera image
L_{dim}	LIDAR points' dimensions
L_n	Number of LIDAR points
$loss$	Loss to reach optimal Q-function
L_{RV}	Projected LIDAR feature into RV form
l_{RV}	Length of RV image

$\mathcal{L}$	MotionNet loss function
$\mathcal{L}_{bt}$	Background consistency loss
$\mathcal{L}_{class}$	Cell classification loss
$\mathcal{L}_{ft}$	Foreground consistency loss
$\mathcal{L}_{motion}$	Motion prediction loss
$\mathcal{L}_{s}$	Spatial consistency loss
$\mathcal{L}_{state}$	State estimation loss
m	Image pixel
$mask$	Mask filter
n	Frame number
$\hat{N}$	Number of residual images
$Nclasses$	Number of category classes
$Npixels_{image_k}$	Number of pixels in image k ($image_k$)
O	Optimality
$O_{1:T}$	Optimality from time t to T
obj_e	Object
$obj_{e_{pixels}}$	Pixels of object o_e
O_t	Optimality at time t
$O_{\tau+1:T}$	Optimality from time $\tau+1$ to T
PM	Predicted motion
$\tilde{PM}$	Predicted motion in next frame
PRV	Projected RV features
$p(\cdot)$	Probability
$\mathcal{P}$	Transition function
$\hat{p}$	LIDAR points
$\hat{p}_i$	LIDAR point i
$\hat{p}_{i_{\hat{x},\hat{y}}}$	LIDAR point i - $(\hat{x},\hat{y})$ coordinate
$q(\cdot)$	Distribution function

$\hat{q}$	State-action value function
$\hat{q}^*$	Optimal state-action value function
Q_θ	Q-Network parameterized with θ
$Q_{\bar{\theta}}$	Q-Network parameterized with $\bar{\theta}$
$\hat{Q}_\theta$	Delayed Q-Network parameterized with θ
r	Reward
$range$	Range
$range^0$	Range image of current frame
$range^0_m$	Range value at cell m of the current frame
$range^n$	Previous n^{th} transformed range image
$range^n_m$	Range value at cell m of the n^{th} frame
r^c_ω	Comfort reward for ω
r^e_{lat}	Efficiency reward for lateral acceleration
res	Residual
$res^0_{n,m}$	Residual at cell m between the 0^{th} and the n^{th} frames
r^e_{steer}	Efficiency reward for steering
r^e_{step}	Efficiency reward per step
r^e_{vel}	Efficiency reward for speed
r^s_{col}	Safety reward for collision
r^s_{fast}	Safety reward for fast
r^s_{lane}	Safety reward for lane center
r_t	Reward at time t
RV	RV features
r_τ	Reward at time τ
s	State
$\mathcal{S}$	State space
s'	Next state
$scale_l$	Scaled length

$scale_w$	Scaled width
$s\hat{G}_{\hat{p}_i}$	Features of $\hat{p}_i$ in the source form; where $\hat{G} \in \{1,2,3\}$
s_t	State at time t
s_{t+1}	State at time $t+1$
$steer_\alpha$	Steering angle
T	End time of episode
t	Time
$target_{\hat{p}_i}$	Feature of $\hat{p}_i$ in the target form
t_C	Camera capture time
t_L	LIDAR capture time
tmp	Temporary variable
$Trans$	Transformation matrix
$Trans_{C \leftarrow L}$	Transform from LIDAR to camera image
$Trans_{C \leftarrow veh_{t_C}}$	Transform from vehicle's frame at t_c to camera
$Trans_{veh_{t_C} \leftarrow veh_{t_L}}$	Transform from vehicle's frame at t_L to t_C
$Trans_{veh_{t_L} \leftarrow L}$	Transform from LIDAR to vehicle's frame at t_L
$\hat{v}$	State value function
$\hat{v}^*$	Optimal state value function
veh	Vehicle
vel	Velocity
vel_{max}	Maximum velocity
w_{bev}	Width of BEV image
w_C	Width of camera image
w_{RV}	Width of LIDAR RV image
x	Observations
$x_{1:\tau+1}$	Observations from time 1 to $\tau+1$
x_t	Observations at time t
x_{t+1}	Observations at time $t+1$

$y_t^{double\hat{q}}$	Double DQN state-action value function
$y_t^{\hat{q}}$	DQN state-action value function
z	Latent state
$z_{1:T}$	Latent states from time 1 to T
$z_{1:\tau+1}$	Latent states from time 1 to $\tau+1$
z_t	Latent state at time t
z_{t+1}	Latent state at time $t+1$
z_τ	Latent state at time τ
$z_{\tau+1}$	Latent state at time $\tau+1$
$z_{\tau+1:T}$	Latent states from time $\tau+1$ to T
α	Learning rate
$\Delta\hat{h}_{bev}$	BEV height resolution
$\Delta\hat{l}_{bev}$	BEV length resolution
$\Delta range_{n,m}^0$	Non-normalized residual image between $range^0$ and $range^n$
Δt	Time difference
$\Delta\hat{w}_{bev}$	BEV width resolution
ε	Constant to trade-off exploitation and exploration
γ	Discount factor
γ^t	Discount factor at time t
ω	Angular velocity
ω_{max}	Maximum angular velocity
ϕ	Network weights for actor network
ϕ^*	Optimal network weights
π	Policy
π^*	Optimal Policy
π'	A better policy compared to π
ψ	Network weights for critic network
σ	Standard deviation of Gaussian layer

$\boldsymbol{\tau}$	Current time
$\boldsymbol{\theta}$	Network weights for Q-Network
$\bar{\boldsymbol{\theta}}$	Delayed network weights
$\boldsymbol{\theta}_t$	Network weights at time t
$\hat{\boldsymbol{\theta}}_t$	Delayed network weights at time t

CHAPTER 1
INTRODUCTION

This chapter begins by giving general background information on autonomous driving. Second, the motivation behind this Ph.D. thesis is presented. Next, the objectives of this Ph.D. thesis addressing the challenges facing researchers in developing urban autonomous driving are demonstrated. Subsequently, the contributions made in this Ph.D. thesis are discussed, and a list of publications is provided. Lastly, the chapter is concluded with the structure of this Ph.D. thesis.

1.1 Background

The field of autonomous driving is blooming at an expedited rate since the Defense Advanced Research Projects Agency (DARPA) pioneered 'DARPA Grand Challenge' [3, 4]. Outstanding findings and enhancements on autonomous driving are continuously being discovered and released by researchers. Today, scientists from different fields are collaborating closely to make autonomous driving become a reality sooner than expected, as the capabilities of autonomous driving can have profound impacts on the world [5, 6]. Even during the global COVID-19 pandemic, the relevant research maintained its peak level. It is believed that it is only a matter of tweaking the technology before autonomous vehicles are fully developed and are ready to be deployed on roads.

For vehicles to have a degree of autonomy, they need to perceive and interpret their surroundings. Perception can be done by merging data from high-end sensors deployed on the vehicle, and interpretation is conducted through intelligent and sophisticated algorithms. Depending on the task assigned to autonomous driving, the sensors required to accomplish that specific task can vary. Usually, one or more sensors are used to assist a vehicle in perceiving the environment. These sensors include a camera, light detection and ranging (LIDAR), radio detection and ranging (RADAR), ultrasound, global positioning system (GPS), inertial measurement unit (IMU), and many others. Over the last decade, many companies started their adventure in developing autonomous driving. Today, some top companies specializing in autonomous driving are Waymo, Tesla Motors Inc., BMW, Volvo cars, Mitsubishi, and Uber. Figure 1.1 shows an autonomous vehicle designed

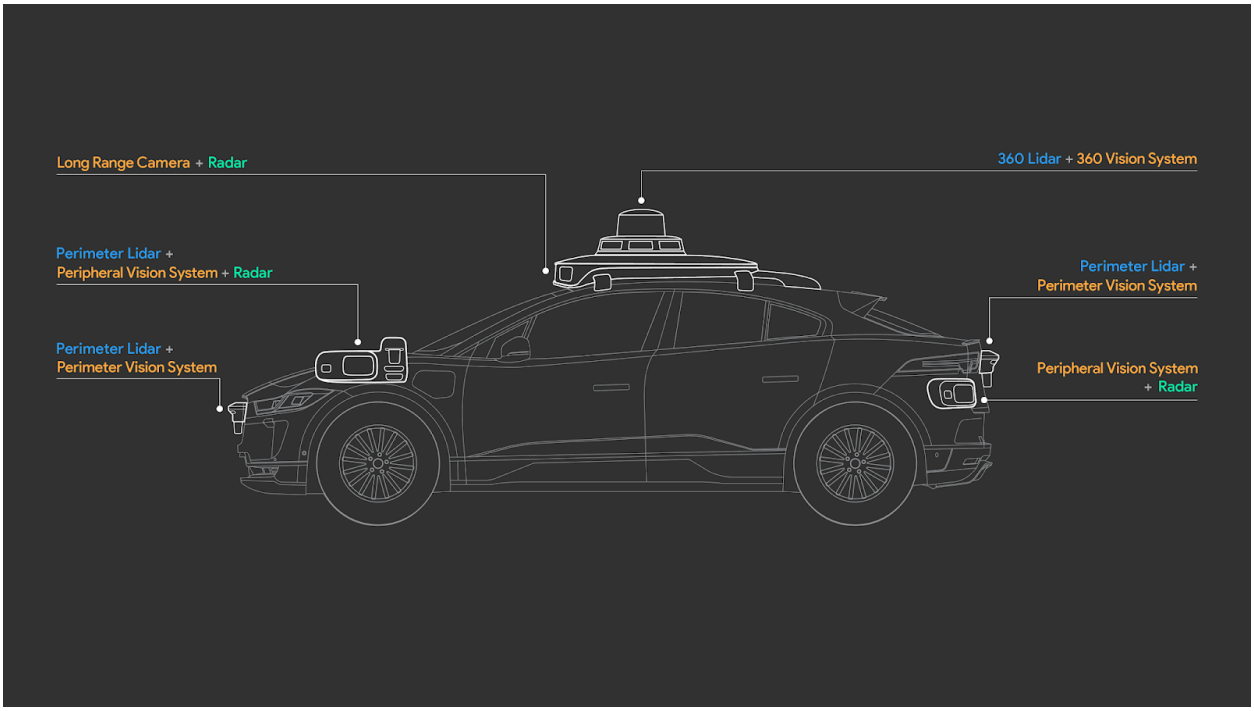


Figure 1.1 Waymo perception sensors [1].

by Waymo [1]. The figure shows that the main sensors used in Waymo to attain autonomy are camera, LIDAR, and RADAR. Different companies have different designs for autonomous driving, especially with the number and variety of sensors, and even the designated sensor positions can differ. A sensor's efficiency depends on many factors, including the hardware used, precision, speed, and cost. Figure 1.2 provides a general idea on the detection ranges for different sensors and their potential applications [2]. The complexity of developing a fully autonomous vehicle can be deduced from Figure 1.2. Autonomous driving may involve many intricate tasks, and sometimes, for better outcomes, these tasks need to be connected.

The Society of Automotive Engineers (SAE) [7] defined six levels of autonomy, starting at level 0 and up to level 5. The higher the levels, the degree of autonomy increases, and the vehicle take over more driving tasks.

- ***Level 0*** (No Automation) - The driver is in complete control of driving as no automation is installed in the vehicle. The driver is continuously exercising both longitudinal and lateral control.
- ***Level 1*** (Assisted Driving) - The diver is continuously exercising either longitudinal or lateral

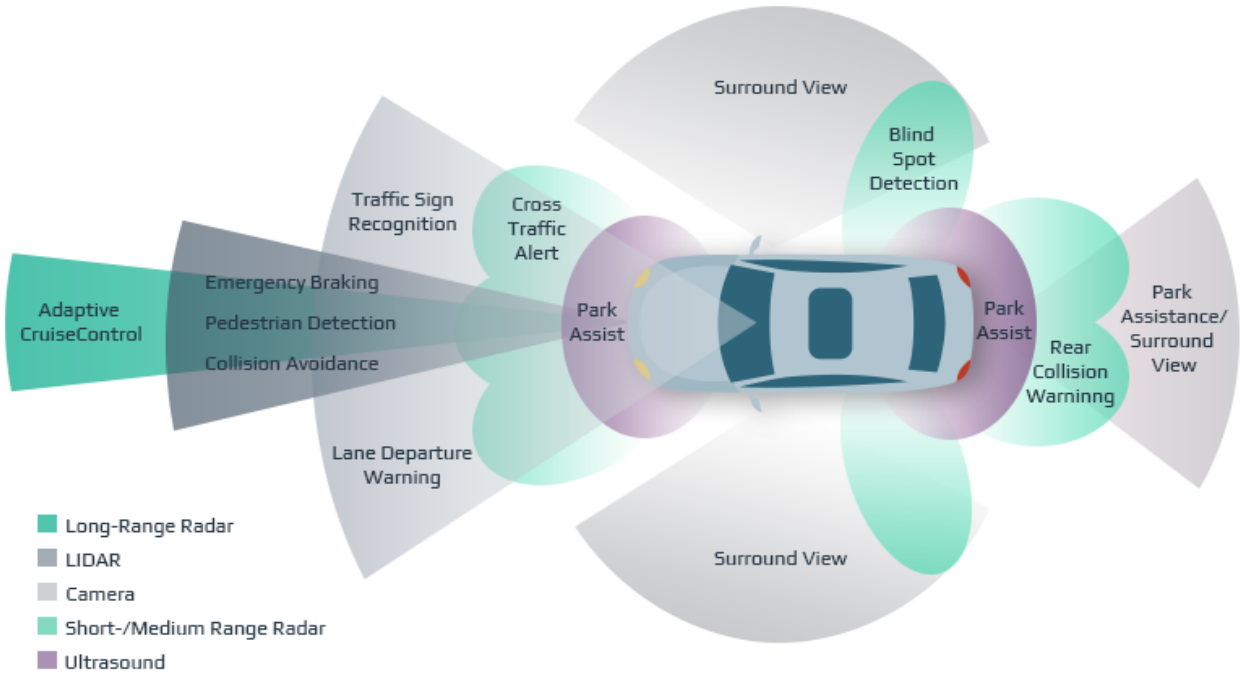


Figure 1.2 Sensor ranges and their applications [2].

control, and the system accomplishes the other driving task.

- ***Level 2*** (Partial Automation) - Includes both longitudinal and lateral control. However, the driver is still required to monitor the system at all times.
- ***Level 3*** (Conditional Automation) - The driver does not have to monitor the system at all times but must be vigilant to take over at any moment.
- ***Level 4*** (High Automation) - The driver is not required during some defined use cases. The system can handle the situation autonomously, just in defined situations. Highway driving is an example where the system can handle autonomously.
- ***Level 5*** (Full Automation) - The system can cope with all situations automatically during the entire journey. No driver is required, as the system does not rely on human input. An example of level 5 is autonomous driving, where the system can handle all scenarios and not just a defined use case like highway driving.

The history of autonomous vehicles and their chronological development can be found in [8, 9]. Autonomous vehicles must attain the objectives of safety, efficiency, and comfort to be deployed on roads and gain peoples' trust to ride them or even accept the idea of autonomous vehicles being

deployed among them, i.e., on roads.

Typically, the architecture of autonomous driving constitutes several pipelines that range from perception, motion prediction, planning to control [10, 11, 12, 13]. There exist two broad categories for developing autonomous driving: modular and end-to-end approaches [14, 15]. Modular approach is where each pipeline is designed and trained independently, and in the last stage, all developed components are stitched together to achieve autonomy. Whereas, in end-to-end approach, the entire system is treated as one complete problem. A direct correlation is generated between the input and the output. Therefore, a single system is built, trained, and tested, covering all independent components. Observations from the environment are fed into the system and processed in a network such as deep reinforcement learning (DRL) [16, 17]. Finally, control commands are engendered to drive the vehicle autonomously. One of the critical advantages of end-to-end learning over modular is that it is less prone to propagating errors. If a glitch happens at an early stage in a modular design, the problem will get propagated and exacerbated towards the end.

1.2 Motivation

Human driving is prone to accidents, imposing the need for better on-road driving decisions to avoid catastrophic consequences. According to WHO [18], 1.39 million people die annually due to driving accidents, and an additional 20 to 50 million people suffer from injuries, with some getting disabilities. Based on the study in [19], human driving errors are the leading cause of these tragedies. During the COVID-19 lockdown period, the traffic volume dropped dramatically, resulting in an 80% reduction in road fatalities [20]. This strengthens the direct correlation between fatality rates and incompetent human driving decisions. On the other hand, Morando et al. [21] investigated the exceptional impact of autonomous driving on saving people's lives, and findings revealed that autonomous driving could indeed reduce traffic accidents. Moreover, the study in [21] adds that autonomous driving can save a plethora of lives as 90% of accidents are caused by human error. By its very nature, autonomous driving eliminates potential human driving errors, leading to safer and more efficient traffic [5].

With the striking achievements of deep learning, DRL has emerged from combining RL and

deep learning techniques. DRL has witnessed exciting developments and registered remarkable breakthroughs in various domains since the DARPA Grand Challenge, especially in autonomous driving [22, 23, 24]. Different types of DRL algorithms exist and are mainly categorized as value-based [25, 26, 27], policy-based [28], and actor-critic-based [29, 30, 31, 32] methods. The success and proficiency behind DRL algorithms in autonomous driving are mainly due to their capability to solve complex high-dimensional problems from raw observations such as images [14]. Even though DRL algorithms are considered competent in solving complex problems; however, they attempt to solve representation learning and task learning concurrently. This ensues in two challenges that affect the learning efficiency and performance of the DRL algorithms. The first challenge is the difficulty of learning a representation from high-dimensional observations. Second, high-dimensional observations increase sampling complexity, making task learning harder to accomplish, thus negatively impacting learning efficiency and performance. Furthermore, a DRL algorithm must have a good trade-off between exploring the environment to collect additional knowledge and utilizing the learned knowledge to accomplish its task [33]. This trade-off is of paramount importance to provoke learning a driving policy that meets the objectives of safety, efficiency, and comfort. To overcome these challenges, this Ph.D. thesis employs a DRL algorithm that separates representation learning from task learning and embeds strong exploration and reinforcement capabilities.

As mentioned earlier, the architecture of autonomous driving consists of several pipelines. In general, DRL-based algorithms solve all pipelines in an end-to-end manner. Initially, inputs are passed into the DRL agent; then, all pipelines are processed internally to produce the control signals (i.e., throttle and steering) as outcomes. In autonomous driving, the two most pivotal pipelines are perception and motion prediction, as they allow the vehicle to observe the environment and forecast the dynamics in its surroundings. All subsequent pipelines rely on the accuracy of both perception and motion prediction. An autonomous vehicle cannot attain the objectives of safety, efficiency, and comfort without these pipelines. Supplying the DRL agent with perception and motion prediction information would boost the agent's internal perception and motion prediction pipelines. In addition, improvements will be reflected on the other pipelines as they depend on the outcomes of the perception and motion prediction pipelines. Incorporating perception and

motion prediction information into the DRL-based approach provides a relevant low-dimensional representation of surrounding scenes, enhancing the overall performance and resulting in a better driving policy. This valuable supplementary information fed into the DRL algorithm contains the objects in its proximity and their future trajectories, helping the agent adapt its dynamics more effectively, thus leading to a better driving policy. To that extent, this Ph.D. thesis proposes feeding perception and motion prediction into the DRL algorithm to learn a better driving policy.

Furthermore, the perception and motion prediction information fed into the DRL agent must be accurate as their goal is to nourish the agent's internal pipelines and enrich performance. Moreover, they must be generated at a high inference speed due to the real-time constraint of autonomous driving. Therefore, fundamental to the network that generates perception and motion prediction are its compact and highly informative input data, accuracy - especially for small and distant objects, and speed efficiency. Various perception and motion prediction networks exist [34, 35, 36, 37]; however, they either lack a rich and comprehensive input source that strongly affects the accuracy level or have slow inference speed. Such networks cannot be deployed in the real-time autonomous driving application as accuracy and speed are critical.

Recent works demonstrated that the use of multi-modal fusion provides a perception and motion prediction network with rich and complementary input information, which in turn positively influences accuracy [38, 39, 40, 41, 42]. Furthermore, accuracy can be further enhanced by performing predictions at the pixel level predictions instead of object level predictions [43]. Small and distant objects are harder to detect, and so their detection accuracy benefits most with pixel-wise predictions compared to object level predictions. Another advantage of pixel level compared to object level predictors is that they are easier to generalize to object classes not available in the training set. In terms of speed, several techniques can be utilized to maintain the real-time requirement of the autonomous driving application. The main ones are avoiding the use of 3D convolutions, which are considered speed inefficient [44], and having a lightweight network [34]. Furthermore, TensorRT [45] can be applied to boost inference speedup. Thus, this Ph.D. thesis proposes exploiting multi-modal fusion network that is accurate, speed efficient, and performs pixel-wise joint perception and motion prediction. These predictions are then incorporated into the DRL algorithm, which separates representation learning from task learning and includes powerful exploration and

reinforcement to learn a driving policy that attains the objectives of safety, efficiency, and comfort.

1.3 Objectives

The main objective of this Ph.D. thesis is to develop an autonomous vehicle that uses intelligence to achieve navigation in an urban environment while securing safety, efficiency, and comfort. In addition, this autonomous vehicle should fit well to different environments and varying weather conditions. Moreover, as the perception and motion prediction pipelines of an autonomous vehicle are the most pivotal, an accurate real-time perception and motion prediction model should be developed to enhance performance.

1.4 Contributions

The contributions of this Ph.D. study are as follows:

- A novel real-time multi-modal fusion network that generates accurate pixel-wise perception and motion prediction is proposed. The attained performance for perception and motion prediction outperforms the state-of-the-art, especially for small and distant objects. The inference time of our multi-modal fusion network has been improved using TensorRT optimizer.
- Predictions from our multi-modal fusion network are leveraged in an end-to-end sequential latent maximum entropy reinforcement learning (MaxEnt RL) model. The sequential model learns a low-dimensional latent space from the input, and the MaxEnt RL is trained on that latent space to produce the driving policy. The learned driving policy met the objectives of safety, efficiency, and comfort in an urban environment; and maintained its effectiveness under different environments and weather conditions.

1.5 Outline

This Ph.D. thesis is organized as follows. This chapter presented a brief introduction on the field of autonomous driving, the motivations of this Ph.D. thesis, objectives, and contributions. Chapter 2 begins with a detailed review of works that performs perception and motion prediction for autonomous driving. Additionally, an introduction to the field of RL and DRL is presented.

Further, an in-depth and comprehensive review is conducted of works that applied DRL to learn autonomous driving tasks. Chapter 3 presents our proposed end-to-end multi-view fusion network that fuses only bird's-eye view (BEV) and range view (RV) representations. Chapter 4 explains our proposed fusion network LiCaNet, which is an expansion to our earlier multi-view network to include camera images. Chapter 5 presents *LiCaNext*, the *next* version of LiCaNet, where range residuals are introduced to enhance accuracy even further. Chapter 6 provides details on our proposed approach that incorporates perception and motion prediction, extracted from multi-modal fusion network *LiCaNext*, into a sequential latent MaxEnt RL algorithm to learn a better driving policy. Finally, this Ph.D. thesis is concluded in Chapter 7.

1.6 List of Publications

This Ph.D. thesis consists of works that are published in the following journal paper, conference proceedings, and papers awaiting publications:

1. Y. H. Khalil and H. T. Mouftah, "LidNet: Boosting Perception and Motion Prediction from a Sequence of LIDAR Point Clouds for Autonomous Driving," in the Proceedings of the 2022 IEEE Global Communications Conference (GLOBECOM), IEEE, 2022 (under review).

2. Y. H. Khalil and H. T. Mouftah, "Exploiting Multi-Modal Fusion for Urban Autonomous Driving using Latent Deep Reinforcement Learning," in IEEE Transactions on Vehicular Technology, 2022 (under review).

3. Y. H. Khalil and H. T. Mouftah, "LiCaNext: Incorporating sequential range residuals for additional advancement in joint perception and motion prediction," in IEEE Access, vol. 9, pp. 146 244–146 255, 2021.

4. Y. H. Khalil and H. T. Mouftah, "LiCaNet: Further enhancement of joint perception and motion prediction based on multi-modal fusion," in IEEE Open Journal of Intelligent Transportation Systems, vol. 3, pp. 222–235, 2022.

5. Y. H. Khalil and H. T. Mouftah, "End-to-end multi-view fusion for enhanced perception and motion prediction," in the Proceedings of the 94th Vehicular Technology Conference

(VTC2021-Fall), IEEE, 2021, pp. 1–6.

6. Y. H. Khalil and H. T. Mouftah, “Integration of motion prediction with end-to-end latent RL for self-driving vehicles,” in Proceedings of the International Wireless Communications and Mobile Computing (IWCMC), IEEE, 2021, pp. 1111–1116.

CHAPTER 2
RELATED WORK

This chapter begins with a detailed review of prominent works that performed perception and motion prediction in the field of autonomous driving. Next, we provide background knowledge to introduce reinforcement learning (RL) and the paradigm of deep reinforcement learning (DRL), with a detailed discussion on some of its state-of-the-art algorithms. In addition, a comprehensive review is done of works that attempted solving autonomous driving tasks using DRL. Finally, we conclude the chapter with a discussion on the reviewed works.

2.1 Perception and Motion Prediction

Perception and motion prediction are of paramount importance for autonomous driving, where perception involves detecting objects in the surrounding scene, and motion prediction predicts future maneuvers of the detected objects. Various approaches have been proposed to model the task of perception and motion prediction in autonomous driving; however, works that encompass multi-modal fusion are the most prominent. Research works have established various methodologies for formulating their input, whether sourced from single or multiple sensors. Works that deployed a LIDAR, camera, and LIDAR and camera combined are targeted.

2.1.1 Predictions using LIDAR Sensor

There exist many models that depend on a single LIDAR sensor, each representing its input features differently. Some of the prior works processed the raw 3D point cloud directly without any transformations. To begin with, PointRCNN [46] is a point-based method that generates 3D proposals for 3D object detections using a two-stage method: the bottom-up 3D proposal generation and refining the proposals. Shi et al. [47] extends PointRCNN to achieve 3D object detections using part-aware and aggregation neural network. Another common approach is to transform the point cloud into 3D voxels. VoxelNet [48] predicts 3D detections using voxel encodings. Additionally, fast point R-CNN [49] and PV-RCNN [50] performed 3D object detection by incorporating voxel-based and point-based for better point cloud feature learning. FlowNet3D

[51] and HPLFlowNet [52] used point-based approach to estimate the scene flow between two point clouds, while LSTM-Encoder-Decoder [53] uses recurrent deep learning approach to produce long-term predictions of the vehicle environment in a grid map representation. The dataset used in [46, 47, 48, 49, 50, 51, 52] is KITTI dataset [54]. In [50], waymo open dataset (WOD) [55] is also used for experimental evaluation, while [51, 52] uses FlyingThings3D [56] in addition to KITTI dataset. A private dataset is used in [53].

Although perceiving the environment through point-based or 3D voxels has benefits; however, the runtime requirement suffers due to the processing of sparse representations and 3D convolutions. Typically, point-based models are data-intensive, and so in addition to being time-consuming, they face computation and memory bottlenecks; thus cannot be scaled easily. Recently, researchers have considered transforming point clouds into 2D images, such as bird's-eye view (BEV) and range view (RV), for their compactness, efficiency in processing, and effectiveness in improving the performance of point cloud classification and segmentation. There exist works that used 2D LIDAR-based image representations for perception by generating 3D object-level proposals [57, 58], but our focus is on works that performed both perception and motion prediction. Next, works that addressed perception and motion prediction are reviewed.

Casas et al. [37] proposed an end-to-end framework that simultaneously performs 3D detections and motion predictions. The model is named Fast and Furious (FaF) and relies on LIDAR 3D point clouds formulated in BEV images. The model's input is denoted in 4D tensors constituting of 3D occupancy map from the 3D LIDAR point clouds, and the fourth dimension is the time dimension. Detections are performed using one stage detector and operate on 3D occupancy maps. Even though the detector is fed with 4D tensors, it generates bounding boxes from different timestamps, with each input holding a shape of 3D tensors. The technique used to generate 3D occupancy maps is by creating voxels from the 3D LIDAR data. This is achieved by assigning each voxel a value of 1 if at least one point exists; otherwise, the value is 0. From the 3D occupancy maps, the detector creates 3D bounding boxes using its 2D convolutions. The third dimension of the 3D occupancy maps is treated as the channel dimension. To this point, only detections have been performed. So, the fourth dimension is required to perform motion prediction, which is time. All past point clouds are first discretized individually and then synchronized to the current times-

tamp's coordinate system. These discretized and synchronized frames are then stacked to form the 4D tensor. The authors in [37] evaluated the exploitation of the temporal dimension using two techniques: early and late fusion. In early fusion, all temporal features are inserted in the first layer. The advantage of early fusion is that it runs as fast as a single frame detector. However, late fusion is capable of extracting high-level motion features. The late fusion network consists of 3D convolutions. Finally, after generating bounding boxes for each timestamp, tracklets are produced by aggregating the bounding boxes related to the same object. Motion predictions are forecasted for 1 second, and the intentions of other vehicles are not involved in the process. The number of past LIDAR sweeps used as input to the model is 5. The proposed model runs in real-time at 33Hz. Results show that late fusion acquires higher performance than early fusion but is slower due to 3D convolutions.

Cascas et al. [59] proposed IntentNet, an advancement of FaF [37], an end-to-end deep neural network that reasons about other vehicles' intentions, predicts long-term trajectories, and gives better detections in terms of accuracy. IntentNet relies on LIDAR data and high definition maps (HD maps) as input and results in three outcomes: detections for vehicle and background classes, intentions, and predicted trajectories through bounding boxes in current and future timestamps. Inputs are in the form of BEV images. HD maps are used to provide information about the structure of the environment, including stop signs, the width of lanes, type of lane markings, etc. The predicted intentions are classified into one of 8 behavioral categories. Late fusion has been adopted as it proved useful in FaF, and the 3D convolutions are replaced with 2D. The two inputs are fed into two different convolutional networks for feature extractions. Then, the features are concatenated and inserted into the fusion subnetwork. Lastly, the outputs of the fusion subnetwork are given as input to the rest of the architecture, which results in three output predictions. The number of past LIDAR sweeps used as input to the model is 10. Predictions are generated for 3 seconds into the future. The technique used for discretization and synchronization of the LIDAR frames is the same as FaF.

Zeng et al. [36] proposed an end-to-end motion planner that is an enhancement to IntentNet [59]. Similar to IntentNet, the inputs for [36] are LIDAR point clouds and HD maps in BEV form. The proposed model outputs 3D detections and future trajectories for all autonomous vehicles

in the scene. Another output is space-time cost volume which evaluates each possible position the agent can take. The motion planner samples trajectories for the agent using the learned cost volume and assigns them scores. The planner will select the agent's forecasted trajectory with the lowest cost among the sampled ones. The LIDAR point clouds are preprocessed before being concatenated with the HD maps. The concatenated features are then fed into the convolutional networks. Equivalent to IntentNet, 10 LIDAR sweeps are used as input to the model. Discretization and synchronization of LIDAR frames are the same as in IntentNet. The backbone network is responsible for detecting bounding boxes, trajectories, and cost volume. The motion planner has no backpropagation as it does not predict, but it only samples trajectories and picks the one that has the lowest cost. The end-to-end models in [36, 59, 37] provide interpretability and allow uncertainty to get propagated.

Liang et al. proposed PnPNet [35], which is a model that performs detection, tracking, and motion prediction. The tracking module exploits the object detections to provide object tracks. A tracking module supplies more information about object states to the prediction module, which helps generate more accurate trajectories. PnPNet was developed and trained end-to-end as it only has one background network and shares computation across all modules. Both LIDAR point clouds and HD maps are used as inputs to PnPNet in BEV form. A novel trajectory level representation is introduced in PnPNet where it captures an object's temporal characteristics. The number of LIDAR sweeps used is 10, and predictions are performed for 3 seconds into the future.

Cascas at al. [60] proposed SpAGNN an end-to-end framework that takes into consideration social interactions between vehicles using graph neural networks (GNNs) [61] to predict motion and reduce uncertainty. The great success of GNNs comes from their capability to function under varying input size and their proficiency in learning good representations. A joint optimization can be done in the proposed model for detection and motion forecasting using one single model. The input parameterization is the same as in [36]; also, 10 LIDAR sweeps were used. The HD map is represented in 7 channels where each is encoded to a different semantic. Both forms of input are fed into a backbone network to extract features but with different configurations from [36]. The features are then concatenated and fed into a header convolutional network. In order to attain anchor scores and bounding boxes, different convolutional layers are applied to the outcomes of

the header convolutional network. The final detections are achieved by applying non-maximum suppression (NMS) to anchor scores and bounding boxes. The second part of the proposed model is dedicated to providing motion forecasts by encoding actor relations and considering the uncertainty. This is accomplished by the spatially-aware GNNs.

Meyer et al. [62] proposed LaserFlow, which makes use of RV, rather than BEV, representation of LIDAR data to detect and predict motion. A new methodology is introduced to fuse RV temporal images. RV images do not require discretization and hence encompass the uncompressed LIDAR's raw points. This means that RV images have fine-grained information and include accurate information about small objects like pedestrians and bicycles. The output is in BEV form and consists of detections and motion forecasting for each detected object. The concept of curriculum learning was applied to learn probability distributions over predicted trajectories. Predicted trajectories were 3 seconds in length.

Laddha et al. [63] proposed RV-FuseNet that depends merely on sequential RV images to formulate their input, which is eventually used to perform object detection and motion prediction. RV images are utilized instead of BEV images to avoid the loss of information due to voxelization errors. In addition, RV images are more compact and processed efficiently compared to BEV images. Information loss with RV fusion is minimized by using incremental fusion, where each RV sweep is projected sequentially into the viewpoint of the next sweep in time. Similar to LaserFlow, RV-FuseNet forecasts motion 3 seconds into the future. Performance analysis demonstrated that RV-FuseNet outperformed LaserFlow and SpAGNN in perception and motion prediction.

Djuric et al. [64] proposed MultiXNet a build-up on IntentNet. MultiXNet model is multistage and is specialized in detecting and predicting motion for multiple classes: vehicles, bicycles, and pedestrians. The inputs are LIDAR data and HD map in BEV form, and the output is multi-modal distribution of motion predictions for actors in the scene. The discretization and synchronization of the input are the same as in the prior works. The proposed method jointly learns trajectories and uncertainty. After detecting and predicting motion, the refinement network, which is the second part of the model, is responsible for refining the forecasted motion of each detected object. The authors designed the model to predict a constant number of motion predictions with their proba-

bilities for each detected object. A private dataset is used in [37, 59, 36] is collected over multiple cities across North America. Experiments for [35, 60, 62, 64] are conducted on the private ATG4D [65] and the public nuScenes [66] real-world self-driving datasets. However, in [63] nuScenses and internal private LSAD datasets were used.

All works reviewed so far engender 3D object-level predictions, even though different combinations of representations were used to formulate the input. The generation of 3D proposals is computed using 3D convolutions, which are computationally inefficient than 2D convolutions. Furthermore, a major weakness of object level predictions (i.e., bounding box-based systems) is their difficulty detecting objects that are not included in the training set. This constraint can be evaded with pixel-based predictions. Fewer works are conducted that adopt 2D input images for their input features and use 2D convolutions to perform pixel-wise predictions. SqueezeSegV3 [67] used RV images to perceive the driving environment through segmentation. SalsaNext [68] a real-time, uncertainty-aware semantic segmentation model that is based on RV representations. AMVNet [69] use RV images to perform semantic segmentation using an assertion-guided sampling strategy. Motion prediction was not performed in [67, 68, 69].

MotionNet [34] is a lightweight network that constructs its input from sequential BEV images to perform pixel-wise joint perception and motion prediction in real-time. MotionNet is trained to detect 5 object categories, including background, vehicles, pedestrians, bikes, and others. The *others* category is assigned to detect objects that are not categorized in any of the remaining four groups. Motion prediction attained by MotionNet is forecasted 1 second into the future. The main block in the MotionNet backbone constitutes two 2D convolutions followed by one pseudo-1D convolution. The lack of 3D convolutions is the key reason behind its speed efficiency. MotionNet performance surpassed state-of-the-art methods in both perception and motion prediction.

To conclude this section, fusing LIDAR-based 2D image representations is more effective than other LIDAR-based representations because they are compact, efficient to generate and process, and 3D convolutions can be avoided. Furthermore, it is possible to generate real-time pixel-wise predictions with such representations.

2.1.2 Predictions using Camera Sensor

Perception algorithms that utilize deep learning and depend on a camera sensor fall under the category of image-based detections based on convolutional neural network (CNN) architectures. Typically, such frameworks are divided into two streams: one-stage and two-stage object detections. One-stage detectors (e.g., YOLO [70] and SSD [71]) directly map input features to class probabilities and bounding box coordinates via a single-stage CNN model. Whereas two-stage detectors (e.g., Faster R-CNN [72] and R-FCN [73]) firstly extract region proposals, then they are refined down the pipeline to generate object classification and regression. Predominantly, one-stage detectors are faster than two-stage detectors but are inferior to two-stage detectors in terms of detection accuracy. An excellent performance assessment of one-stage and two-stage detectors in autonomous driving can be found in [74]. Over the past decade, image-based object detections have picked a staggering pace in the field of autonomous driving [75, 76, 77, 78].

A typical problem with image-based CNN detectors is that accuracies for large and small objects are unbalanced. Large objects are represented by sufficient features permitting them to be classified correctly with high confidence. In contrast, small objects are usually represented by inadequate features and thus left undetected or classified with low confidence. In the field of autonomous driving, it is essential to detect small objects (e.g., pedestrians and bikes) to maintain their safety. Recently, researchers proposed image-based CNN algorithms with a focus on detecting small objects. FPN [79] is a two-stage multiscale network that achieved high detection accuracy for small objects. The technique adopted in FPN is the fusion of multiscale features. PAN [80] is an enhanced version of FPN that also specializes in detecting small objects. YOLOv4-5D [75] proposed an improvement to the PAN backbone network to increase the detection accuracy for small objects. Even though these detectors could achieve better detection results for small objects, they are bounding box-based methods. Several successful attempts exist for building a pixel-wise detector using only camera sensors and registered challenging outcomes. Porzi et al. [81] proposed semantic segmentation using a single backbone network, outperforming UPSNet [82] which uses parameter-free panoptic head for segmentation. Yang et al. [83] proposed an end-to-end unsupervised learning framework to perform depth estimation and camera motion prediction. Re-

sults showed that using stereo image sequences surpasses scale ambiguity for depth estimation and increases motion prediction accuracy for temporal image sequences.

Therefore, from the review of work that relies merely on camera sensors, it is clear that pixel-wise predictions are essential for perceiving small and distant objects in autonomous driving; however, the inference time requirement remains a challenge. Furthermore, motion prediction using only a camera sensor is a recent research topic, and results are still not ideal, mainly due to the lack of depth information in camera semantics.

2.1.3 Predictions using Fusion of LIDAR and Camera Sensors

Presently, the utilization of multi-modal fusion for perception and motion prediction has gained much attention among researchers in autonomous driving. Multi-modal fusion is used to exploit the complementary properties of different sensors and representations. Feng et al. [39] presented a survey on different multi-modal methodologies for object detection and segmentation in autonomous driving. Although in Section 2.1.1 plenty of works that utilize multi-modal fusion were reviewed; however, they were sourced from one sensor. An in-depth review of the fusion of point clouds and images can be found in [84]. Liang et al. [85] fused BEV and camera images to perform 3D object detections using a continuous fusion layer. Moreover, Liang et al. [41] enhanced the 3D object detection model in [85] by reasoning about 2D and 3D object detections, ground estimation, and depth completion. LaserNet++ [40] is another model that performs 3D object detections; however, by fusing RV and camera features. LaserNet++ reported good performance results, especially for small and distant objects. PointPainting [86] applied sequential fusion for semantic segmentation using the painting technique where the point cloud is augmented with image semantics. Later, the 3D detections are extracted by applying the painted point cloud to a LIDAR detector. Lastly, MVX-Net [87] proposed a 3D object detector using two fusion techniques (point-wise and voxel-wise) to fuse point clouds and camera images.

These works used multi-modal fusion to perform just perception using 3D object proposals. An enhancement model to MultiXNet [64] is proposed by Fadadu et al. [38]. The model in [38] is end-to-end and specializes in joined detection and motion prediction, exploiting multiple LIDAR

representations, camera images, and HD maps as input. On top of LIDAR-based BEV representation, an RV representation is used. With BEVs, the 3D point clouds are discretized into voxels and thus lose fine-grained information, whereas, in RV, the point clouds are not discretized but include the raw LIDAR points. According to the authors, RV representation assists in accurately detecting small objects like pedestrians and bikes. Additionally, RV images can effectively be fused with camera images. However, the learning representation problem becomes complex as the objects in RV are different from those in BEV representation. Hence, the authors propose a method to efficiently integrate RV and BEV images originating from LIDAR, camera, and HD maps. An overall BEV image is generated by fusing all four input representations and is then fed into MultiXNet [64] backbone. In order to generate the overall BEV image, two paths take place. One path is for combining HD maps with the aggregated LIDAR sweeps in BEV form. The other path is for combining the RV image with the camera image. Later, the outcomes from both paths are integrated to form the overall BEV image. Historical information is incorporated only in the LIDAR-based BEV representation.

Lastly, MVFuseNet [88] implements perception and motion forecasting by fusing sequential LIDAR data in both BEV and RV forms, in addition to HD map features. Unlike [38], MVFuseNet performs spatio-temporal fusion of both BEV and RV features for multiple frames. MVFuseNet reports improved performance over [38] in perception and motion prediction across all object categories. However, again, 3D object-level predictions were computed in both [38, 88]. It is evident that no work has yet been conducted that investigates pixel-wise joint perception and motion prediction using multi-modal fusion, which is essential for small and distant objects as they provide fine-grained, pixel level precision. The datasets used to evaluate the experiments in [38, 88] are the public nuScenes and internal private datasets.

In contrast to the reviewed works above, we propose a novel multi-modal fusion network named *LiCaNext*, which fuses features from LIDAR and camera sensors. Three representations are extracted from the LIDAR sensor: BEV, RV, and range residuals. Both the BEV and the range residual images consist of temporal information. Thus, *LiCaNext* generates features that embrace: 1) temporal, depth, and physical object dimensions in BEV form; 2) occlusion and high-resolution point information in RV form; 3) temporal information in RV form holding range data for rich

motion cues; and 4) semantics encompassed in camera images. These rich and integral features enable improved accuracy for real-time pixel-wise joint perception and motion prediction when inserted into MotionNet backbone, especially for small and distant objects.

2.2 Reinforcement Learning

Reinforcement learning (RL) [16, 89] is a subcategory of machine learning [90] and is composed of a group of algorithms that tells the agent how to behave in different situations. The RL algorithms learn a mapping from states to actions allowing agents to maximize the reward by interacting with the environment through trial-and-error. Theoretically, RL has four essential elements: agent, environment, action, and reward. Those four elements need to be defined first to model any RL model. For example, to model an autonomous driving task, the environment would be defined as the town the agent (autonomous vehicle) navigates in. The reward is positive if the agent drives efficiently and safely; negative if a collision occurs or if there are abrupt changes in acceleration.

The paradigm of RL can be modeled as a Markov decision process (MDP) where at each timestep t, an agent observes state s_t, and takes action a_t according to policy $\pi(a_t|s_t)$. Based on the executed action, the environment subsequently evolves into the next state s_{t+1} according to the transition function $\mathcal{P}(s_{t+1}|s_t,a_t)$. In addition, the agent collects a reward signal r_t from the reward function $r(s_t,a_t)$. This process continues until the agent reaches a terminal state and then restarts. An agent-environment interaction formulated in a MDP is shown in Figure 2.1. In MDP, the next state depends on the current state and the action taken, and each performed action results in a reward. To sum up, a MDP is a mathematical framework defined by the 5-tuple $\mathcal{E} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0,1]$ is a state-transition probability function, $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function, and $\gamma \in [0,1]$ is the discount factor indicating the importance of future rewards on the current state.

The agent's objective is to learn an optimal policy π^* that maps states to actions and maximizes the return G. A policy $\pi(a_t|s_t)$ is a characterization of agent's behavior, where it is the probability that action $a_t \in \mathcal{A}$ is executed if the agent is in state $s_t \in \mathcal{S}$. The return G, defined in (2.1), is the discounted sum of rewards starting at time step $t = 0$ until the terminal state of the episode $t = T$.

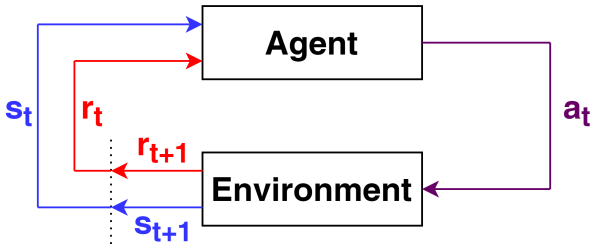


Figure 2.1 The principle of reinforcement learning interaction.

When $\gamma = 0$, it means that the agent is focused on maximizing immediate rewards $r(s_t, a_t)$, whereas when $\gamma = 1$, it means the agent is focused on maximizing future rewards.

$$G = \sum_{t=0}^{T} \gamma^t r(s_t, a_t) \tag{2.1}$$

In order for the agent to achieve an optimal policy π^*, the value of a state and/or the value for the action taken at a particular state needs to be evaluated. This concept defines two forms for the value function, where the agent needs to evaluate through interacting with the environment (i.e., trial-and-error). One form of the value function is the state value function $\hat{v}_\pi(s)$ which is an estimate of how good for an agent to be in state s. Second, the state-action value function $\hat{q}_\pi(s,a)$ estimates how good for an agent to perform action a and be in state s. The evaluation of how good is measured by computing the return G.

A state value function $\hat{v}_\pi(s)$ is the expected return starting in state s and acting according to policy π until the terminal state (end of episode). Similarly, the state-action value function $\hat{q}_\pi(s,a)$, under a policy π, is the expected return starting in state s and performing action a at any time step until the end of the episode. $\hat{v}_\pi(s)$ and $\hat{q}_\pi(s,a)$ are defined in (2.2) and (2.3), respectively.

$$\hat{v}_\pi(s) = \mathop{\mathbb{E}}_{\pi}\left[G|s_t = s\right] \tag{2.2}$$

$$\hat{q}_\pi(s,a) = \mathop{\mathbb{E}}_{\pi}\left[G|s_t = s, a_t = a\right] \tag{2.3}$$

Following the definition of the value function, the next step is to find the optimal policy π^*

that maximizes the expected return G. It is clear now that value functions are defined in terms of policies, where the value of a state s under a given policy π is the expected return, starting from state s and following policy π thereafter. Consequently, if the expected return of π' is greater than or equal than π then policy $\pi' \geq \pi$. In another words, if and only if the value function $\hat{v}_{\pi'}(s) \geq \hat{v}_{\pi}(s)$ then $\pi' \geq \pi$. As a result, finding the optimal value function leads to the optimal policy. The Bellman optimality equation is used to give the optimal value function. The optimal state value function $\hat{v}^*(s)$ in (2.4) and state-action value function $\hat{q}^*(s,a)$ in (2.5) give the maximum expected return achievable under any policy π for any state and state-action pair, respectively.

$$\hat{v}^*(s) = \max_{\pi} v_{\pi}(s), \quad \forall \quad s \in S \tag{2.4}$$

$$\hat{q}^*(s,a) = \max_{\pi} q_{\pi}(s,a), \quad \forall \quad s \in S,\ a \in A \tag{2.5}$$

Once we obtain $\hat{q}^*(s,a)$, known as Q-function, we can apply an RL algorithm to get the optimal policy that finds the action that maximizes $\hat{q}^*(s,a)$ for each state-action pair. Q-learning is a common value-based RL technique used to solve for the optimal policy by learning optimal Q-values for each state-action pair [91]. Watkins and Dayan proved the convergence of Q-learning [92]. Q-learning iteratively updates the state-action pair values until the Q-function converges to the Bellman optimal Q-function $\hat{q}^*(s,a)$, as defined in (2.6).

$$\hat{q}^*(s,a) = \mathbb{E}\left[r_t + \gamma \max_{a' \in A} \hat{q}^*(s',a')\right] \tag{2.6}$$

The state-action pair values are updated using an update rule in (2.7), and each value is stored in a Q-table as shown in Figure 2.2. The rows of the Q-table represent the states, columns are actions, and each cell holds the Q-value. Initially, cells can be filled with random Q-values or can even be zeros. In the Q-function's update equation, $\alpha \in [0,1]$ is the learning rate that controls the amount of updates for Q-values in the Q-table. Basically, the updated Q-value is the weighted sum of the old and new learned Q-value. The old Q-value is updated with a rate of $(1-\alpha)$ and the new

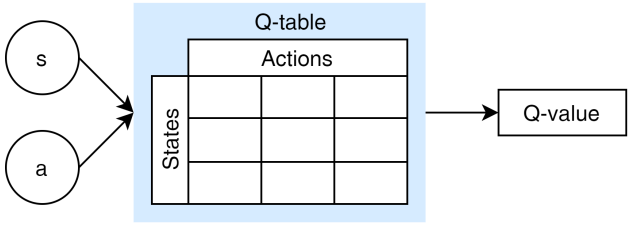


Figure 2.2 Q-learning scheme.

learned value with rate α.

$$\hat{q}(s,a) \leftarrow \hat{q}(s,a) + \alpha\left[r_t + \gamma \max_{a' \in A} \hat{q}(s',a') - \hat{q}(s,a)\right] \tag{2.7}$$

There exist several other techniques to reach the optimal policy. As discussed earlier, value-based functions are one of the methods where the optimal policy is achieved through optimization of the value function. Another technique is policy-based functions, which finds the optimal policy by directly optimizing policies rather than finding the optimal value function. These two techniques propose actions and evaluate them through rewards. Nevertheless, the difference relies on where the optimization is done. With value-based functions, the focus is on finding the optimal return, and then the policy is extracted. Whereas in policy-based functions, policies are directly optimized. Another technique for finding the optimal policy is called the actor-critic method, and it is a combination of both value-based and policy-based functions. The "actor" is responsible for selecting actions, and the "critic" evaluates the decisions made by the "actor".

A major challenge in RL is to balance exploitation and exploration. Exploitation is taking actions that are deemed to be rewarding based on the current policy. This reward is considered immediate; however, to maximize the cumulative reward, new actions need to be explored, leading to higher future rewards. Actions of this kind may not be highly rewarding in the current time step but could lead to a higher return on the long run. One of the methodologies used to handle this trade-off is $\varepsilon - greedy$, where $0 < \varepsilon < 1$. An action is exploited according to the agent's current policy with probability ε, and a random action is selected (exploration) with a probability of $(1-\varepsilon)$. To begin with, an agent will have no information about the model and so needs to do more exploration; hence ε will be closer to 1. As training progresses, the agent builds enough

knowledge about the model, so more exploitation than exploration must occur; hence ε is now closer to 0. A model is defined as the agent's view of the environment. When an agent gets a partial view of the environment through observations, the policy is formulated as a partially observable Markov decision process (POMDP).

A model-based RL means that the agent knows how the environment works and has information on the next state in response to an action. While in model-free RL, the agent does not know the environment and has to construct an internal model for itself. Training model-free is harder for the agent than model-based as the agent needs to build a representation of the environment.

The state space and action space representations are defined based on the problem being tackled. For example, in autonomous driving, the state space based on observations may include the direction, orientation, velocity, distance to lanes, or even raw images from sensors. On the other hand, action space may consist of acceleration and steering. Moreover, both state and action spaces could be discrete or continuous. Defining a proper reward function is crucial as it is the only way to control the agent's behavior. Reward function can be designed to include various parameters like collision, distance to lane center, longitudinal and lateral speed, steering angle, and angular velocity. If the state and action spaces are finite, then the RL task that satisfies the Markovian property is called finite MDP.

2.3 Deep Reinforcement Learning

The tabular approach for storing values for each state-action pair has limitations under complex high-dimensional environments. This approach is only convenient for solving problems with small environments having finite state spaces. Building a table is affordable for such small environments as it is easy to traverse and iterate over all the values in the table for updates. On the other hand, with complex and more sophisticated autonomous driving problems, the tabular approach fails due to memory and computational constraints. Furthermore, a challenge like autonomous driving needs to operate in real-time. It is not feasible to traverse and update values in extensive tables quickly. Here is where the advantages of deep reinforcement learning (DRL) [17] come into play. In particular, DRL algorithms powered by deep learning have bloomed after the astounding advances of

deep learning in various fields, including object detection [76] and segmentation [93]. The success of deep learning falls back on its proficiency in function approximation and the sophisticated properties of its convolutional networks. Most deep learning methods use neural network architectures and thus sometimes are referred to as deep neural networks (DNNs). DNN is a feed-forward neural network with two or more hidden layers. DNNs provide function approximations to RL agents, allowing agents to generalize knowledge and apply it to unseen data.

The involvement of DNN in Q-learning to boost its performance is now discussed. The combination of DNN with Q-learning results in Deep Q-learning. The DNN used to approximate the Q-function is called Deep Q-Network (DQN) [25, 26]. The DNNs replace Q-tables, and Q-values are estimated using function approximators generated by training DNNs. Conceptually, states observed from the environment are fed into the DNN, and the outputs are predictions of all state-action pairs (Q-values). For the DNN to approximate the optimal Q-function, it must train on states as inputs and their corresponding Q-values as outputs. The loss defined in (2.8) is the difference between the target Q-values from the Bellman Q-function (2.6) and the estimated Q-values from the DNN. The objective of the DNN is to minimize that loss to reach the optimal Q-function. The weights in the DNN are updated using stochastic gradient descent (SGD) [94]. The performance of DQN has surpassed human-like performance, for example, in Atari games [25]. In an environment like autonomous vehicles, the inputs to DNN are states represented as images originating from a sensor such as a camera. The network is often referred to as a policy network because its objective is to find the optimal Q-function, and the network weights in DNNs are randomly initialized.

$$\begin{aligned} loss &= \hat{q}^*(s,a) - \hat{q}(s,a) \\ &= \mathbb{E}\left[r_t + \gamma \max_{a' \in A} \hat{q}^*(s',a')\right] - \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r(s_t,a_t)\right] \end{aligned} \tag{2.8}$$

A technique called experience replay is used to enhance sampling efficiency. Experience collected from the environment is stored in a replay buffer rather than training on it directly. The model trains on randomly selected batches of experience from the replay buffer. Each single experience from the replay buffer consists of the following tuple $\langle s_t, a_t, r_t, s_{t+1} \rangle$. The tuple contains

the state s_t of the environment at time t with the action taken a_t, the received reward r_t, and the resulting new next state s_{t+1}. The key advantage of using a replay buffer is that it breaks the correlation between sequential experience samples in training. Moreover, replay buffer allows higher data efficiency as each sample $\langle s_t, a_t, r_t, s_{t+1} \rangle$ in the replay buffer can be sampled more than once during training. If the replay buffer has a maximum capacity of $buffer_{size}$, then it can not store more than $buffer_{size}$ samples of experience. Thus, old samples get cleared out when the replay buffer reaches its maximum capacity to allow newer samples to be trained on.

Double DQN [27] is an enhanced version of DQN. In DQN, one network $q_\theta(s,a)$ is used for both selecting and evaluating the Q-values (2.9). This results in overoptimistic value estimates. To overcome this issue, Double DQN proposes the use of two networks of the same model but with two different sets of weights (2.10). One network is used for evaluating, online network $q_\theta(s,a)$, and the other is for selecting, target network $q_{\hat{\theta}}(s,a)$, the Q-values. The target network for selecting Q-values is the exact copy of the online network but from the last episode. Hence, in the current episode, the online network is used for evaluating Q-values, and when it comes to selecting Q-values, the target network is used. Double DQN proved its effectiveness in giving more accurate value estimates leading to higher rewards.

$$y_t^{\hat{q}} = r_t + \gamma \hat{q}(s_{t+1}, \underset{a}{\operatorname{argmax}}\, \hat{q}(s_{t+1}, a;\, \theta_t);\, \theta_t) \tag{2.9}$$

$$y_t^{double\,\hat{q}} = r_t + \gamma \hat{q}(s_{t+1}, \underset{a}{\operatorname{argmax}}\, \hat{q}(s_{t+1}, a;\, \hat{\theta}_t);\, \theta_t) \tag{2.10}$$

The Dueling DQN, also known as DDQN, [95] architecture is another variant of DQN that leads to better performance measures. DDQN has two streams; this differs from DQN and Double DQN, which have only one stream. The two streams in Dueling DQN are state value $\hat{v}(s)$ and the advantages for each action $Adv(s,a)$ (2.11), while the other two architectures have only $\hat{v}(s)$. The two streams in DDQN are then combined with an aggregation layer to form the estimate of state-action value function $\hat{q}(s,a)$. The advantage is a quantity obtained by subtracting the state value from the state-action value. It defines how much better is an action compared to the rest of the available actions. Decoupling the state value and the actions in the DDQN architecture allows

for better estimates of the value function. In other words, the network gets to learn which states are valuable without knowing the effect of any action on that state. This way, if the state value is bad, then the action values are worthless.

$$Adv_{\pi}(s,a) = \hat{q}_{\pi}(s,a) - \hat{v}_{\pi}(s) \tag{2.11}$$

There exist many more variants of DQN, including Noisy DQN [96], DQN with Prioritized Experienced Replay [97], Deep Recurrent Q-Network (DRQN) [98]. Other types of prevalent DRL algorithms include deep deterministic policy gradient (DDPG) [29], Twin Delayed Deep Deterministic Policy Gradient (TD3) [30], proximal policy optimization (PPO) [28], and soft actor-critic (SAC) [31]. Due to the enormous number of works done in the area of DRL, this study does not provide an exhaustive list for all DRL algorithms.

Indeed, DRL has proved its power in developing autonomous agents in broad range of fields including audio-based applications [99], multiple agent systems [100], mobile and wireless networking [101], connected autonomous vehicles in smart cities [102], and optimal control [103]. In most works, results achieved have registered outcomes that exceed human-like results. The main success behind DRL is its ability to learn low-dimensional feature representations from complex high-dimensional problems. The competence of DRL in solving high-dimensional complex autonomous driving problems is presented in [33, 14].

Next, a detailed review of prominent works that applied DRL algorithms on autonomous driving tasks is provided. Existing literature contains plenty of information in the DRL domain; however, only the most relevant works to our problem are addressed. The majority of the DRL-based autonomous driving works target solving single scenarios such as lane keeping, lane change, parking, intersection navigation, merging, and other simple tasks like roundabout navigation. Nevertheless, designing a policy for multi-scenario cases like urban autonomous driving remains under-explored.

Even though plenty of works have been conducted on imitation learning (IL), a different family of learning-based algorithms for learning autonomous driving policies, the majority of recent

works leverage DRL algorithms. IL-based approaches learn driving policies by mimicking human driving behavior [104, 105, 106, 107, 108, 109, 110]. The major drawbacks of IL are the need for a large training dataset composed of expert driving demonstrations, its performance is restricted to the expert driving experience, and the dataset does not cover extreme scenarios. Alternatively, the DRL paradigm learns its policy through balancing self-exploration and reinforcement in the environment; thus, overcoming IL constraints.

2.3.1 Lane Keeping and Lane Change Assistance Systems

Dong et al. [111] proposed a system that enables connected autonomous vehicles (CAVs) to make collaborative lane change decisions using DQN in combination with graphical convolution network (GCN) [112]. In order to enhance CAV's operation, the algorithm proposed in [111] was installed on centralized architectures like roadside units (RSUs) and cloud platforms. The number of vehicles being controlled is dynamic as vehicles are continuously entering and leaving the proximity of the agent. Typically, optimization-based controllers cannot handle problems with dynamicity in their input and output. In [111], the dynamicity challenge is solved by using a novel DQN fusion method. Generally, in an MDP problem, the agent interacts with only the environment, as shown in Figure 2.1. Whereas in [111], the CAVs (agents) can interact with each other in addition to the environment, as presented in Figure 2.3. According to the authors, local and global information is needed to enable CAVs to make collaborative lane change decisions. Local information is sourced from the vehicles' sensors and used to perform lane changes successfully. On the other hand, global information is sourced from the connected infrastructures and are used to avoid traffic jam by making proactive lane changes. Hence, the need for both local and global information is crucial in such problems. In [111], the fusion of both kinds of knowledge is achieved by GCN. After fusion, DQN is applied to the centralized infrastructure. The involvement of DQN results in safe and collaborative lane changing decisions and adds robustness to the proposed system, where consistent decisions can be made even under varying traffic loads. Experiments conducted involve human-driven vehicles and CAVs. An improvement to the architecture proposed in [111] was addressed in [113] where the DQN algorithm was replaced with DDPG. Additionally, the work in [113] targeted resolving highway bottleneck congestion. The outcomes of [113] proved its efficacy in lowering congestion from bottleneck scenarios. The simulator used in both works is

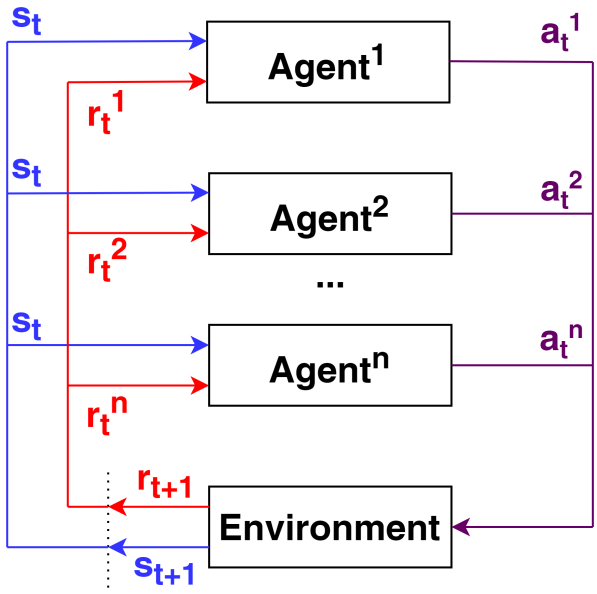


Figure 2.3 Markov game scheme.

Simulation of Urban MObility (SUMO) [114].

Huegle et al. [115] proposed a lane change model using DRL for autonomous vehicles. The authors used Deep Sets to handle varying input processing in the lane change problem. The variation in input size comes from the variable number of vehicles surrounding the agent at any instant in time. The integration of Deep Sets with DQN to solve the lane maneuvering problem was named DeepSet-Q. Additionally, the authors proposed replacing Deep Sets with convolutional neural networks (CNNs) and recurrent neural networks (RNNs) for performance comparison with Deep Sets. Results showed that DeepSet-Q outperformed the rest of the models and even had better generalization capability. The simulator used in this paper is SUMO.

Sallab et al. [116] proposed lane-keeping assist algorithms with two categories, one that operates on discrete actions and the other for continuous actions. A DQN was applied for discrete actions and a variant of actor-critic for continuous actions. An open-source simulator The Open Racing Car Simulator (TORCS), was used to conduct experiments. Three sensors were used to provide information as input to the proposed model, and sensor fusion technique was used to integrate and extract features from all sensors. The sensors utilized are camera, LIDAR, and RADAR.

In another paper, DDPG was used to construct a lane following model in rural areas [117]. The only source of input used in the model was a camera. First, training was conducted on a 3D driving simulator, using Unreal Engine 4 to tune the hyperparameters. Next, the tuned hyperparameters were transferred onto training an autonomous vehicle in the real-world with one lane path and no nearby vehicles. The proposed network was trained in the real-world, where there was an onboard driver that gave the car a negative reward whenever it drove out of the lane. Furthermore, a positive reward was given internally based on the distance traveled autonomously on the lane. Due to the transfer of the tuned hyperparameters, training a policy to drive in the real-world was quickly accomplished. A variational autoencoder (VAE) was used in [117] to encode the camera images before passing them into the training agent. Encoding input images significantly enhanced training efficiency.

Lane change is explored by Alizadeh et al. [118] using DQN in a dynamic and uncertain highway environment. The observations include the relative position and velocities of all actors within the agent's proximity. Each agent in the simulation environment is configured to encompass a lateral controller, adaptive cruise controller, and a safe lane change MOBIL algorithm [119]. The simulation environment can generate traffic scenarios where multiple cars execute lane and speed change decisions. Experiments demonstrated that training a DRL agent in a noisy environment with dynamic actors results in more robust and reliable performance.

Ye et al. [120] successfully learned a smooth, safe, and efficient driving policy to perform lane-change decisions using PPO in scenarios that involve 5 surrounding vehicles. The state space comprises 21 low-dimensional state variables, and 5 variables represent each vehicle. The high-level PPO policy is responsible for generating lane-change decisions, at each time step, based on the current driving situations. At the same time, a low-level controller is used to regulate the longitudinal and lateral movements to generate a corresponding control command that executes the decision. A pre-defined model executes the lower-level controller. All experiments were conducted on SUMO simulator, showing a 99% success rate.

In addition, Chen et al. [121] develops a DDPG-based lane change policy that considers interactions with surrounding vehicles. Experiments demonstrated the success of the learned policy in

performing lane change in dense traffic using only a single camera input image. An attention-based mechanism with hierarchical action space has been adopted and proven in experiments to boost the DDPG performance. The attention mechanism automatically focuses on areas that may affect the driving behavior, such as feasible paths and surrounding vehicles.

2.3.2 Parking Assistance Systems

The authors in [122] proposed a system using PPO to obtain a driving policy within a short period of training time that is capable of exploring a parking lot successfully while avoiding obstacles. The proposed approach maneuvers autonomously and avoid obstacles by performing an evasive maneuver or a stop. A simulated environment was created for data generation, and performance has been evaluated both in simulation and real-world.

Zhang et al. [123] proposed an end-to-end system that tackles automatic perpendicular parking. The DRL algorithm used to handle this task is DDPG. Experiments have shown outstanding results where intelligent human-like parking was accomplished. The simulation platform was coded using PreScan, MATLAB/Simulink, and Python. After training in simulation, experiments were conducted in the real-world. The vehicle had four fisheye cameras spawned on it to detect parking slots.

2.3.3 Intersection Assistance Systems

Tran et al. [124] simulates a traffic environment that includes autonomous and human-driven vehicles to solve the problem of non-signaled intersections. The DRL algorithm used in this paper is PPO. Results demonstrated an enhanced training policy, mobility, and energy efficiency performance. It was also reported that as the number of deployed autonomous vehicles goes up, the performance does not degrade but increases. SUMO was the traffic simulator embraced to conduct the experiments. Observations into the network include the speed and position of the agent and the speed and distance of the preceding and leading autonomous vehicles.

Chen et al. [125] proposed the use of conditional DQN to extract a driving policy that is trained end-to-end. In addition, the concept of defuzzification was used to improve stability in

path following and making smooth turns at intersections. The authors claim that the association between steering angle and acceleration in one output of a Q-network is a limitation and leads to low prediction values. Hence, fuzzy logic was added to handle multiple controlling parameters. Through defuzzification, the parameters get converted to precise values. The agent's global path is fed into the network to take part in the decision process. Images from the camera deployed on the agent are encoded by a network containing a CNN and long short-term memory (LSTM) [126] to extract spatio-temporal features. After encoding the images, the outcomes are later fed into the proposed system. The simulator used in [125] to conduct experiments is CAR Learning to Act (CARLA) [127].

Li et al. [128] proposed a system that considers an agent's safety and efficiency while driving through intersections. The authors used an end-to-end approach to train their network that is based on DQN. Camera images were used as input to the DRL network. Moreover, the distance between the agent and nearby vehicles was included in the network's input enhance performance. Experiments were conducted on CARLA open-source simulator.

Isele et al. [129] proposed a model that tackles the task of maneuvering partially occluded intersections. The methodology used to handle this task is DQN. The authors investigated some exploratory actions to understand the scene better when faced with partially occluded intersections. One of the exploratory actions discussed was that an agent could do a creeping movement at an occluded intersection, i.e., moving a little forward, to gather more information about the scene and clear some of the occlusion. The input to the DQN is in BEV form, and the experiments were run using the SUMO simulator.

In [130], a system was proposed that focuses on four-way intersections in an urban environment. The goal is to train an agent to perform safe maneuvers while following traffic rules under dense traffic. The system proposed a variant of DQN to achieve the task, where a thresholded lexicographic Q-learning framework was adopted. The simulator used to conduct experiments is SUMO. Results showed that the proposed variant of DQN resulted in improved efficiency compared to the baseline, and the trained agent was capable of performing maneuvers safely and efficiently under heavy traffic.

Qiao et al. [131] proposed a curriculum-based DRL method to speed up training a driving policy for stop-sign intersections in an urban environment. On top of speeding up the training process, curriculum learning helps in improving the agent's performance. Curriculum learning involves two levels: the higher level of learning involves generating curriculum based on total rewards, while the lower level, the DRL algorithm, is applied to train a policy. Two DRL algorithms were implemented, DQN for discrete action space, while DDPG is used for continuous action space. Experiments were conducted on simple scenarios only.

Bouton et al. [132] investigates the navigation of urban intersections using the DQN algorithm. Model-checker is exploited to guarantee safety, while belief updates ensure system robustness against perception errors. The location of vehicles behind obstacles is estimated using the learned belief updater, which is composed of an ensemble of RNNs. The state of a traffic participant constitutes position, heading, and longitudinal velocity. However, static objects are represented by their positions and sizes.

Lastly, DDQN was used in [133] to explore the task of navigating unsignalized intersections crowded with pedestrians. A belief update is employed to obtain the perceived state of the environment and the future trajectories of pedestrians given the noisy observations. CARLA simulator was used to conduct experiments. Results show that the proposed approach learns a policy that does not collide with pedestrians while successfully navigating an intersection.

2.3.4 Merge Assistance Systems

In [134], a solution was proposed to allow autonomous vehicles to safely and smoothly merge into dense traffic by interacting with surrounding vehicles. The authors used PPO to approach this problem. The authors state that training a driving policy to predict acceleration and steering angle directly leads to jerkiness and discomfort for passengers. However, if the policy was taught to predict the time derivative of jerkiness and the steering rate, this would lead to smoother signals. The proposed method does not rely on other vehicles' motion predictions but only on interactions with neighboring vehicles to accomplish a merging maneuver. Results verify that the proposed approach successfully merges in dense traffic, and it outperforms traditional rule-based models

and model-predictive control-based models. The system was simulated through an open-source simulator called Automotive Driving Models.

Bouton et al. [135] developed a system that drives on dense highway traffic and aims to merge by making negotiations with different types of surrounding vehicles. Willingness for nearby vehicles to communicate ranges from collaborative (willing to give space) to aggressive (will not permit passing them). Game theory, specifically level-k reasoning, was used in cooperation with a variant of DQN to achieve this task. The level-k method exposes the trained agent to various scenarios, enabling learning of a robust driving policy. The system can generalize to unseen data. The input to the network includes features related to the agent and the surrounding vehicles. The surrounding vehicles' features are defined relative to the agent, and the agent's state is defined by the longitudinal and lateral positions and longitudinal and lateral velocities. The proposed approach was verified to learn more efficient policies than traditional training methods. The simulation environment is implemented using the Julia language [136].

Bouton et al. [137] proposed lane merging that interacts with traffic participants in a crowded scene using DQN. The inputs used to represent a vehicle's state are its distance to the merge point, longitudinal velocity, acceleration, and cooperation level. Simulated experiments reveal that the DQN agent can benefit from reasoning about the interaction with other vehicles.

2.3.5 Other Assistance Systems

Szoke et al. [138] proposed a policy-based RL method to learn a safe driving policy that drives on a dense highway in the shortest amount of time. The state of the environment is represented by a vector with the agent's properties, lane ID, and surrounding vehicles' properties. SUMO simulator was used in [138] to conduct and evaluate experiments. Experiments demonstrate the capability of the learned agent to control a vehicle in a crowded and constantly changing highway environment without collision or termination events most of the time.

Balaji et al. [139] developed a model-free method that depends on PPO to navigate a race track autonomously, relying on a single camera. The method was trained in simulation and was tested, with no further adjustments, in the real-world with a 1/18th scale car. Gazebo was used for

simulation. No other vehicles were deployed in the environment along with the agent. The learned driving policy achieved robust path planning.

In [140], a variant of DQN was used to develop a driving policy for navigating a kart racing game field. Only screen pixels are used for input, and a speed control signal is generated to control the agent. The agent is trained and evaluated on the Mario Kart 64 environment [141]. The system proved that it could extract robust features from the input pixels to drive autonomously in the specified environment.

Chen et al. [142] built a driving policy that enters and exits a roundabout safely without colliding with nearby vehicles. The driving policy is learned from low-dimensional visual encodings. The encodings are captured from a semantic map in BEV representation constituting road geometry, routing, historical detected objects, and historical ego state information. The authors evaluated the proposed approach on three DRL algorithms (DDQN, TD3, and SAC) in a roundabout scenario under heavy traffic conditions. Experiments showed superior performance for SAC over the other two algorithms, where the learned policy completed the assigned task more frequently than the others. CARLA simulator was used to conduct experiments.

In [143], a DQN-based method was proposed to achieve optimal decisions for energy management systems in autonomous vehicles. The work focused on efficient fuel consumption and ways to reduce it to address the problem of air pollution and greenhouse emissions. The authors showed that their proposed system gives a fuel saving of 16.3%. Moreover, DDQN was also applied and presented even higher savings than what was accomplished by DQN.

2.3.6 Urban Autonomous Driving

As mentioned earlier, few works exist that address urban autonomous driving using DRL algorithms. Agarwal et al. [22] achieved an urban driving policy using the PPO algorithm. Low-dimensional inputs are employed that enfold a latent encoding of several BEV semantically segmented images, trajectory waypoints, kinematic features, and traffic light states. The benefits of combining low-dimensional inputs, especially the encodings, are presented in [22], demonstrating an enhanced performance for the urban driving policy. CARLA simulator was used for experimen-

tal evaluation.

Ahmed et al. [23] used a direct perception method to train a DDPG-based policy in an urban environment. The prediction model takes in sequential camera images to generate affordance predictions. These predictions are then concatenated with state observations and vehicle measurements to form the input to the DDPG training agent. Reported results show that using predictions as input to the DRL algorithm performs better than training a DRL algorithm merely on raw camera image data. Experiments were simulated on CARLA.

Chen et al. [24] proposed using sequential latent maximum entropy reinforcement learning (MaxEnt RL) for learning a driving policy in an urban environment. The authors feed the system with a combination of input images, including LIDAR data in BEV form, camera image, and semantic map in BEV form. An encoding for each input is extracted in the environmental model before being fed into the MaxEnt RL model. Experiments simulated in CARLA showed superior performance of the proposed approach compared to DQN, DDPG, TD3, and SAC.

2.4 Discussion

In this chapter, a detailed review was given in Section 2.1 of works that applied perception and motion prediction for autonomous vehicles using different sensors (LIDAR, camera, and combination of both). Additionally, a comprehensive review was presented in Section 2.3 of works that applied DRL algorithms to handle autonomous driving tasks. Table 2.1 provides a summary of the reviewed networks for joint perception and motion prediction, whereas Table 2.2 provides a summary of the reviewed works that used DRL algorithms.

From the works conducted on perception and motion prediction, it is clear that many ideas exist; however, works that used multi-modal fusion surpassed networks that relied on a single input representation. Moreover, works that performed pixel-wise predictions reported positive reflections on the accuracy of small objects. A weakness with object level predictions is that it is difficult to detect objects that were not trained on previously. Thus, this Ph.D. thesis proposes a novel real-time multi-modal fusion named *LiCaNext* to generate perception and motion prediction at a pixel level. *LiCaNext* produces highly accurate predictions, especially for small and distant

objects. *LiCaNext* depends on four input representations: BEV, RV, camera, and range residual images. All of BEV, RV, and range residual images are extracted from a LIDAR, and both BEV and range residual images incorporate temporal information. The multi-modal *LiCaNext* input features generate a rich and complementary set of features that enhance accuracy. These features embrace:

- Temporal, depth, and physical object dimensions in BEV form.
- Occlusion and high-resolution point information in RV form.
- Temporal information holding range data in RV form for rich motion cues.
- Rich semantics in RGB camera images.

Furthermore, it is evident from the surveyed works in section 2.3 that DRL algorithms are capable of attaining outstanding performance in autonomous driving and can overcome human-level performance. Therefore, it is safe to deduce that DRL algorithms are a valid candidate for handling the intelligence required to enable autonomous driving. Even though most surveyed works targeted only simple tasks such as lane change, lane merge, and intersection navigation; however, fewer works exist on urban autonomous driving. The above review shows that training DRL algorithms on encoded input features offer an advantage over training directly on raw input data. This assists in simplifying and enhancing the training capability of the DRL algorithm leading to improved performance. Additionally, going a step backward and replacing the input sensor data with predictions generates more compact features of lower dimensionality. Then encoding such rich and compact information to be fed into the DRL algorithm would push performance even further. No work has been investigated yet that uses LIDAR-based multi-modal fusion to extract pixel-wise joint perception and motion prediction as input to train a sequential latent MaxEnt RL algorithm. To that end, this Ph.D. proposes using a novel real-time multi-modal fusion network named *LiCaNext* to extract joint perception and motion prediction. *LiCaNext* predictions obtain high accuracy for vehicles, especially for small and distant objects like pedestrians and bikes. These accurate predictions are further encoded with another simple BEV image to form a comprehensive and compact latent representation of the surrounding scene. Finally, the latent encodings are utilized by the MaxEnt RL to train a complex driving policy that ensures safety, efficiency, and comfort. Due to the limited computation, the camera module of *LiCaNext* is removed when

Table 2.1 Summary on the reviewed works related to joint perception and motion prediction.

Objective/Task	Sensor/Input	Dataset
FaF - performs predictions for vehicles [37]	LIDAR sweeps (BEV form)	Private Dataset (PD)
IntentNet - Incorporates other vehicles' intentions [59]	LIDAR sweeps (BEV), HD maps (BEV)	PD
In [36], vehicle detections are performed, and space-time cost volume is predicted for evaluating the sampled trajectories using the motion planner	LIDAR sweeps (BEV), HD maps (BEV)	PD
PnPNet - Includes a tracking module on top of detection and trajectory prediction for vehicles [35]	LIDAR sweeps (BEV), HD maps (BEV)	PD, nuScenes
SpAGNN - Considers social interactions between vehicles using GNNs [60]	LIDAR sweeps (BEV), HD maps (BEV)	PD, nuScenes
LaserFlow - Performs predictions for vehicles, pedestrians, and bikes [62]	LIDAR sweeps (RV)	PD, nuScenes
RV-FuseNet - Uses incremental fusion for vehicles [63]	LIDAR sweeps (RV)	PD, nuScenes
MultiXNet - Performs predictions for vehicles, pedestrians, and bikes [64]	LIDAR sweeps (BEV), HD maps (BEV)	PD, nuScenes
MotionNet - Performs pixel-wise predictions for background, vehicles, pedestrians, bikes, and others [34]	LIDAR sweeps (BEV)	nuScenes
Enhancement to MultiXNet, where input is a combination of four images fused in BEV [38]	LIDAR sweeps (BEV & RV), HD maps (BEV), camera	PD, nuScenes
MVFuseNet - Enhancement to [38] using multi-view temporal fusion for vehicles, pedestrians, and bikes [88]	LIDAR sweeps (BEV & RV), HD maps (BEV)	PD, nuScenes

integrated with the sequential latent MaxEnt RL algorithm.

Further information on the proposed approach is provided in the following chapters. The explanation of the multi-modal fusion network *LiCaNext* is done over several chapters. Chapter 3 introduces a part of the proposed *LiCaNext* fusion network where only BEV and RV images are fused. Next, Chapter 4 explains a bigger portion of the fusion network that additionally fuses camera images. This network is called LiCaNet. In Chapter 5, the entire proposed multi-modal fusion network *LiCaNext* is discussed, where in addition to the fusion of BEV, RV, and camera images, range residual images are also fused. *LiCaNext* is the *next* version of LiCaNet. Chapter 6 explains the fusion of *LiCaNext* outcomes with the sequential latent MaxEnt RL algorithm to learn a policy that drives in an urban environment while ensuring safety, efficiency, and comfort.

Table 2.2 Summary on the reviewed works related to DRL algorithms.

Objective/Task	Method	Sensor/Input	Simulator
Lane change with CAVs using GCN [111]	DQN	Vector	SUMO
Lane change with CAVs using GCN [113]	DDPG	Vector	SUMO
Lane change using Deep Sets, CNNs, RNNs [115]	DQN	Vector	SUMO
Lane-keeping assistance [116]	DQN, Actor-Critic	LIDAR, Camera, RADAR	TORCS
Lane following [117]	DDPG	Camera	Simulation, Real-world
Lane change using lateral controller, adaptive cruise controller, and MOBIL algorithm [118]	DQN	Vector	-
Lane change with a low-level controller [120]	PPO	Vector	SUMO
Lane change using an attention-based mechanism [121]	DDPG	Camera	TORCS
Exploring a parking lot [122]	PPO	Vector	Simulation, Real-world
Perpendicular parking [123]	DDPG	Multiple Cameras	PreScan, MATLAB/Simulink, Python
Non-signaled intersections [124]	PPO	Vector	SUMO
Maneuvering intersections using fuzzy logic [125]	DQN	Camera	CARLA
Maneuvering intersections [128]	DQN	Camera	CARLA
Maneuvering partially occluded intersections [129]	DQN	BEV form	SUMO
Maneuvering 4-way intersections [130]	DQN, DQN variants	Vector	SUMO
Maneuvering stop-sign intersections using curriculum based [131]	DQN, DDPG	Vector	-
Maneuvering urban intersections exploiting model-checker and belief updates [132]	DQN	Vector	-
Maneuvering unsignalized 4-way intersections crowded with pedestrians exploiting belief update [133]	DDQN	Vector	CARLA
Merge into dense traffic [134]	PPO	Vector	Automotive Driving Models
Merge into dense highway using game theory [135]	DQN	Vector	Julia language
Lane Merging with vehicles interacting [137]	DQN	Vector	-
Drive on a highway in the shortest amount of time [138]	Policy-based	Vector	SUMO
Navigate a race track [139]	PPO	Camera	Gazebo, real-world
Navigate a track in a kart racing game [140]	DQN variant	Screen pixels	Mario Kart 64
Entering and exiting a roundabout [142]	DDQN, TD3, SAC	BEV form	CARLA
Management of fuel consumption for autonomous vehicles [143]	DQN, DDQN	Vector	-
Urban autonomous driving [22]	PPO	Vector, BEV form	CARLA
Urban autonomous driving [23]	DDPG	Vector, Camera	CARLA
Urban autonomous driving [24]	DQN, DDPG, TD3, SAC	BEV form, Camera	CARLA

CHAPTER 3
END-TO-END MULTI-VIEW FUSION FOR ENHANCED PERCEPTION AND MOTION PREDICTION

This chapter explains the proposed end-to-end fusion of LIDAR-based bird's-eye view (BEV) and range view (RV) features to generate accurate pixel-wise perception and motion prediction in real-time. This chapter begins with a brief introduction and then presents a concise review of similar works. Next, the proposed network architecture that fuses BEV and RV features is illustrated. Subsequently, results are demonstrated, showing the superiority of the proposed fusion approach over the baseline MotionNet and other state-of-the-art methods. Finally, a summary is given to conclude this chapter.

In the next chapter (Chapter 4), LiCaNet is introduced, which builds on the network presented in this chapter to incorporate camera images onto the fusion of BEV and RV features. Subsequently, Chapter 5 introduces *LiCaNext*, the *next* version of LiCaNet. The proposed *LiCaNext* fuses multi-modal features including BEV, RV, camera, and range residual images to enhance performance even further. Chapter 6 presents the integration of *LiCaNext* with sequential latent MaxEnt RL to learn a driving policy that achieves the objectives of safety, efficiency, and comfort. Chapter 7 concludes this Ph.D. thesis.

3.1 Introduction

Autonomous vehicles rely heavily on perception and motion prediction to maneuver safely and efficiently on roads. The LIDAR sensor is now one of the essential sensors used in autonomous vehicles to achieve high-end perception and motion prediction. LIDAR produces massive 3D point cloud datasets that accurately capture depth and intensity information of its environment. These measurements are used for reliable and precise perception [144] and motion prediction [34]. However, processing raw points directly is challenging and inefficient due to the irregularity and sparsity of point clouds. Also, intensive computation would be required to process the colossal number of points. These limitations triggered the idea to process LIDAR data in other views to

avoid the constraints and offer greater performance. Two prevalent LIDAR views are BEV [65] and RV [145, 67] representations.

Even though BEV representation is the most common approach used in perception and motion prediction [34, 64, 146], yet it has some drawbacks [62]. The generation of BEV requires the LIDAR points to be quantized into voxels of predetermined dimensions. This process introduces quantization error, leading to a loss in fine-grained information. Furthermore, distant objects are represented by few points as the density of point clouds is inversely proportional to distance. Likewise, small objects such as pedestrians and bikes are also constituted by few points. Thus, small and distant objects suffer the most from quantization error. The ability to detect distant and small objects is pivotal for the success of autonomous vehicles, specifically when trying to plan and predict motion. It is worth mentioning that the irregularity in point clouds results in numerous empty voxels, inducing redundant computation. On the opposite end, two key advantages exist for BEV representations: 1) object dimensions are range-invariant, offering the training model prior information about object shapes, thus simplifying the learning process; and 2) easy to fuse with historical data of the same form.

In contrast, RV is generated directly using the native LIDAR points, ensuing a dense representation. In recent years, RV representation has received considerable attention in the area of joint perception and motion prediction [62, 63]. RV has proved its competence in this domain, particularly in detecting small and distant objects. RV is computationally more efficient than BEV due to its compact representation. Another essential characteristic of RV is its ability to preserve occlusion information. Unlike BEV, the physical dimensions of objects in RV representation are not maintained as they vary with distance.

Various works addressed the issue of perception and motion prediction - the majority targeting this problem in a modular fashion. In modular approach, each component is designed and trained independently, and at the last stage, all components are aggregated. Only few works exist that formulate this issue in an end-to-end approach, treating the whole model as one complete problem [34, 38, 64, 147]. End-to-end is preferred over modular approaches as they are less prone to the exacerbation of errors and incur lower latency due to feature sharing. Almost all end-to-end works

use a single LIDAR view as input [34, 64, 62, 63, 148], and only some considers multi-view fusion [38, 149].

Lately, works in the context of autonomous vehicles have reported an exceptional gain in accuracy using multi-view fusion of features [145, 38, 69]. As no work has been conducted yet on the use of multi-view LIDAR fusion to jointly perceive and predict motion at a pixel level, this chapter introduces a network that extracts and projects RV features onto BEV to generate rich and complementary features. These features are then fed into a backbone network to enhance perception and motion prediction in an end-to-end fashion. The backbone network used is MotionNet [34], which is a state-of-the-art model that relies on BEV images to perform pixel-wise perception and motion prediction. Experiments show that the fusion of BEV and RV features achieves higher accuracy in perception and motion prediction than the baseline and other state-of-the-art methods. The proposed approach has proved its proficiency in detecting small and distant objects.

3.2 Review

The task of end-to-end perception and motion prediction aims to detect and categorize objects as well as forecast their future positions based on historical information. Cascas et al. proposed IntentNet [59], an advancement of FaF [37], which reasons about other vehicles intentions, predicts longer-term trajectories, and generates better accuracy. Zeng et al. [36] proposed a model that performs vehicle detections and predicts space-time cost volume used for evaluating sampled trajectories. PnPNet [35] offers tracking on top of detection and motion prediction. SpAGNN [60] considers social interactions between vehicles using graph neural networks (GNNs). MultiXNet [64] is a buildup of IntentNet [59] where the classification set is expanded to include pedestrians and bikes. The input used in [59] relies merely on BEV representation sourced from LIDAR data, while [37, 36, 35, 60, 64] incorporates HD maps with the BEV representation. A follow-up on MultiXNet is [38], where BEV, RV, RGB, and HD maps are fused to form the input to MultiXNet backbone. LaserFlow [62] and RV-FuseNet [63] depend on RV form to represent their input, which is eventually used to perform object detection and motion prediction. All of these works are bounding box dependent and generate 3D object detections.

One major weakness of bounding box-based systems is their difficulty detecting objects not presented in the training set. This constraint can be evaded through pixel-based detections. A bounding box-free system that perceives and predicts motion in real-time is MotionNet [34]. MotionNet is a novel model that generates predictions at a pixel level while depending only on LIDAR data in BEV representation as an input source. Experiments conducted in [34] confirm that MotionNet outperforms other baseline models in perception, motion prediction, and inference speed. Consequently, this chapter aims to enhance pixel-wise perception and motion prediction performance by fusing BEV and RV representations. The objective includes enhancing detection accuracy for small and distant objects.

3.3 Proposed Methodology

The proposed network focuses on fusing multi-view features sourced from one LIDAR sensor into a backbone network to enhance perception and motion prediction. This section describes the BEV and RV input representations used in the fusion process. Next, the network used to fuse BEV and RV features is discussed. The projection algorithm from RV onto BEV features is also demonstrated. Lastly, the backbone network is deeply explained.

3.3.1 LIDAR Representation

A LIDAR operates by scanning its entire field-of-view (FOV) using laser beams. The LIDAR measures the time difference between firing a focused laser beam and detecting its reflection. This collected data is used to compute the distance to objects, which can be further used to compute the xyz-coordinates of objects. Provided by nuScenes dataset [66], the LIDAR data is used to extract the BEV and RV forms.

a) Bird's-Eye View: Bird's-eye view (BEV) is formed by projecting the 3D LIDAR points into 2D images of dimensions $l_{bev} \times w_{bev} \times h_{bev}$ meters, and grid cell resolution of $\Delta\hat{l}_{bev} \times \Delta\hat{w}_{bev} \times \Delta\hat{h}_{bev}$ meters in the xyz-axis. The 2D grid images represent the top-down view of the point cloud, where $l_{bev}w_{bev}h_{bev}$ denotes the region-of-interest in the xyz-direction. l_{bev} and w_{bev} are set to each cover a range of 64m in length and width, respectively. The covered range in length is 32m long from

the front- and back-side of the vehicle ($l_{bev} \in [-32, 32]$). Similarly, the width range is divided equally between the left- and right-side of the vehicle ($w_{bev} \in [-32, 32]$). Height dimension h_{bev} covers a total range of 5 meters ($h_{bev} \in [-3, 2]$). The resolution of each 2D grid cell is set to $\Delta\hat{l}_{bev} = 0.25$, $\Delta\hat{w}_{bev} = 0.25$ and $\Delta\hat{h}_{bev} = 0.4$. Discretizing all 3D points into evenly spaced cells results in a BEV image of dimensions $256 \times 256 \times 13$. In other words, the entire height dimension of the point cloud is discretized into 13 image channels ($h_{bev}\Delta\hat{h}_{bev}^{-1}$), each with a size of 256×256 ($l_{bev}\Delta\hat{l}_{bev}^{-1} \times w_{bev}\Delta\hat{w}_{bev}^{-1}$). Upon discretization, a grid cell is considered occupied if at least one LIDAR point is mapped to it and is labeled with a value of 1; otherwise, -1 is assigned.

The procedure outlined above is used to compute one BEV image from a single LIDAR sweep. On the other hand, the BEV input of this work demands a sequence of 5 BEV images. As a result, a sequence of 4 historical LIDAR sweeps is incorporated in addition to the current sweep to provide the capacity to generate motion predictions. Every 5 consecutive (1 current and 4 previous) sweeps are defined as a clip. The historical sweeps are exploited in BEV form because of their simplicity in stacking sequential images. So, on top of computing the BEV image of the current sweep, all 4 historical sweeps need to be converted into BEV form by the same discretization process explained above. Before discretizing the 4 historical sweeps, they must be synchronized with the current sweep's coordinate system. The synchronization process is done through coordinate transformation. This step is necessary to compensate for the autonomous vehicle's motion across time. After the synchronization and discretization steps, all 4 historical BEV images are stacked on top of the current BEV image. Thus, the overall BEV input dimensions become $256 \times 256 \times 13 \times 5$. Figure 3.1 illustrates a sample of sequential raw LIDAR sweeps and the resulting sequential BEV input after synchronization and discretization.

b) Range View: Range view (RV) representation is created by projecting each point $\hat{p}_i = (\hat{x}, \hat{y}, \hat{z})$ in the LIDAR point cloud to a pixel in the projected RV image. A spherical projection is used to speed up computation. The transformation used for projecting to RV image is defined in (3.1).

$$\begin{pmatrix} u_{RV} \\ v_{RV} \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\left[1 - \arctan(\hat{y}, \hat{x})\,\pi^{-1}\right] w_{RV} \\ \left[1 - \left(\arcsin(\hat{z} \cdot range^{-1}) + f_{up}\right) f^{-1}\right] l_{RV} \end{pmatrix}, \tag{3.1}$$

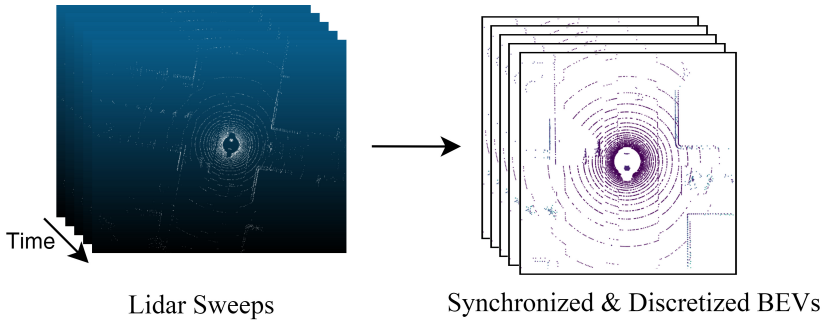


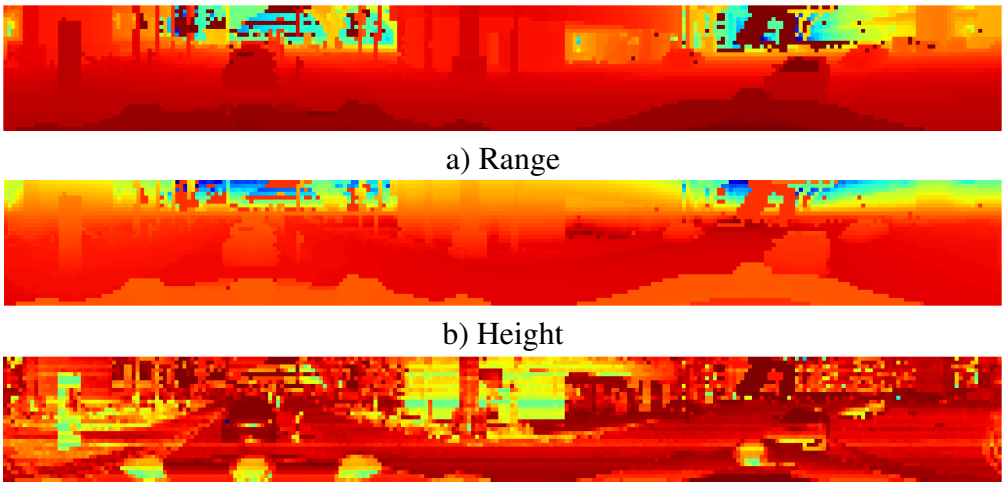
Figure 3.1 BEV features.

where (u_{RV}, v_{RV}) represents the angular coordinates denoting the pixels in the RV image. w_{RV} and l_{RV} indicates the width and length of the RV image, respectively. The vertical FOV of the LIDAR sensor is defined as $f = f_{up} + f_{down}$, and $range = \sqrt{\hat{x}^2 + \hat{y}^2 + \hat{z}^2}$ is the range of point $\hat{p}_i$.

The projected RV image dimensions are designed to be $2048 \times 32 \times 4$. The length value usually reflects the number of LIDAR beams. The LIDAR sensor used in nuScenes dataset is Velodyne HDL32E and has 32 beams. In contrast, the width and channel values are determined by the designer. For each (u_{RV}, v_{RV}) pixel, the 4 channels are constructed by projecting the *range*, height, intensity of $\hat{p}_i$, and the last channel is a binary value indicating if the pixel is occupied by at least one $\hat{p}_i$. If no $\hat{p}_i$ is projected to a (u_{RV}, v_{RV}) then all 4 channels are filled with -1. Unlike the input BEV representation, the RV image only constitutes the current LIDAR sweep with no historical information. In BEV form, past sweeps are simply stacked together. The LiDAR sweep used to generate the RV image is the same as the one used to compute the current BEV image. A sample range, height and intensity of the current sweep in RV form are illustrated in Figure 3.2a, 3.2b, and 3.2c, respectively.

3.3.2 Multi-View Fusion Architecture

As illustrated in Figure 3.3, features from the BEV and RV representations need to be extracted and projected before being concatenated and fed into the MotionNet backbone (discussed in Section 3.3.4). The input BEV representation comprises a sequence of historical sweeps, while the RV input contains no history information and is formed using only the current sweep. The BEV and RV representations are applied independently to two 3×3 convolution layers. The resulting



a) Range

b) Height

c) Intensity

Figure 3.2 RV features.

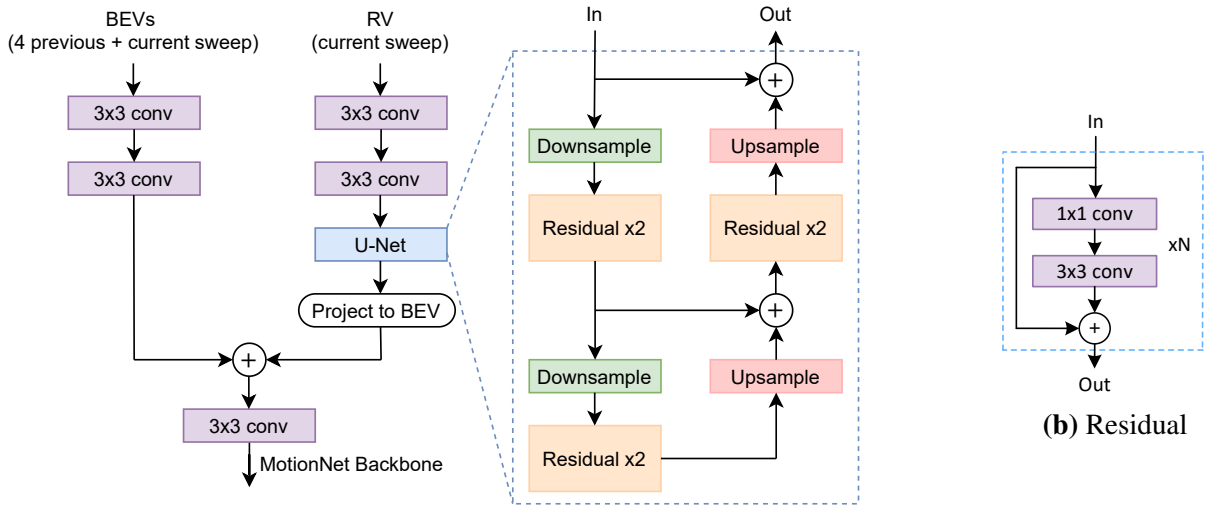
RV features are then applied to a lightweight U-Net architecture [150]. A U-Net shape is leveraged due to its ability to extract and combine multi-scale features. In the downsampling and upsampling layers, a factor of 2 is applied merely on features width while leaving the length dimensions untouched. This is because the horizontal resolution of the RV representation is comparatively bigger than the vertical resolution. Residual blocks in the U-Net design are of depth 2 and embrace skip connections. The resulting RV features are then projected using Algorithm 1 onto the BEV features before being concatenated and finally fed into the backbone network.

3.3.3 Projection

Projection of RV features onto BEV is accomplished using the point painting technique [86]. Algorithm 1 describes the general idea behind the projection technique. Basically, for each LIDAR point mapped to a BEV voxel, its corresponding RV feature is projected onto the same cell position. If more than one RV feature ends up in the same cell, then the average is computed. Empty cells are filled with a value of -1.

3.3.4 Backbone Network

The backbone network used in this work is MotionNet [34]. MotionNet is a novel model that performs pixel-wise joint perception and motion prediction competitively in real-time. MotionNet



(a) Proposed Architecture

(b) Residual

Figure 3.3 The proposed multi-view fusion architecture.

Algorithm 1: Projecting RV features onto BEV

Inputs:

Lidar sweep: $\hat{p} \in \mathbb{R}^{L_n, L_{dim}}$ with L_n points and $L_{dim} = 4$.

RV features: $RV \in \mathbb{R}^{w_{RV}, l_{RV}, ch_{RV}}$ with w_{RV} width, l_{RV} length, and ch_{RV} channels.

BEV dimensions: $bev_{dim} \in \mathbb{R}^2$.

Output:

Projected RV features: $PRV \in \mathbb{R}^{bev_{dim}[0], bev_{dim}[1], ch_{RV}}$.

count = 0 // count $\in \mathbb{R}^{bev_{dim}[0], bev_{dim}[1]}$

for $\hat{p}_i$ *in* $\hat{p}$ **do**

  L_{BEV} = project($\hat{p}_{i_{\hat{x},\hat{y}}}$) // $L_{BEV} \in \mathbb{R}^2$

  if L_{BEV} *falls within* bev_{dim} *range* **then**

    L_{RV} = project($\hat{p}_{i_{\hat{x},\hat{y}}}$) // $L_{RV} \in \mathbb{R}^2$

    tmp = RV[L_{RV}[0], L_{RV}[1], :] // tmp $\in \mathbb{R}^{ch_{RV}}$

    PRV[L_{BEV}[0], L_{BEV}[1], :] += tmp

    count[L_{BEV}[0], L_{BEV}[1]] += 1

  end

  PRV /= count // average (avoid division by 0)

  mask = (count == 0) // mask $\in \mathbb{R}^{bev_{dim}[0], bev_{dim}[1]}$

  PRV[mask, :] = -1 // assign -1 to empty cells

end

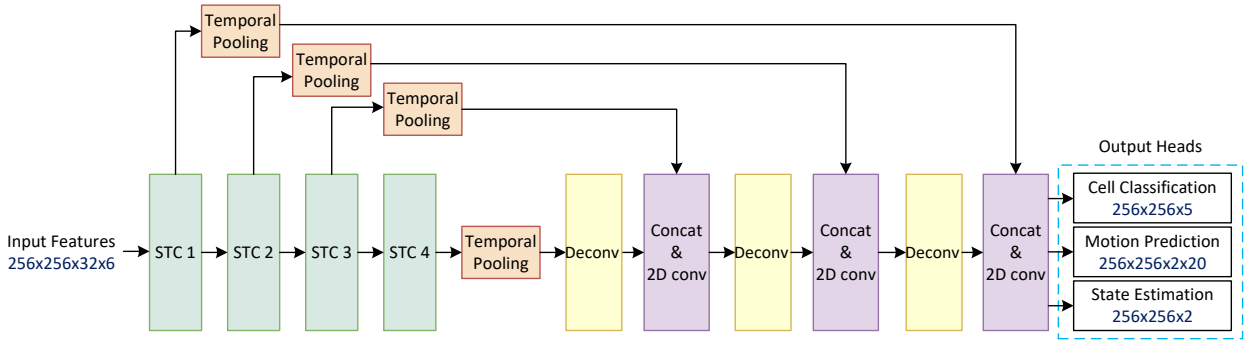


Figure 3.4 MotionNet architecture.

architecture consists of an encoder-decoder named spatio-temporal pyramid network (STPN) and three output heads. A spatio-temporal convolution (STC) block is the main element of STPN, which consists of two 2D convolutions followed by one pseudo-1D convolution. The STPN architecture has STC blocks structured in a hierarchical way to extract features at different scales, leveraging multi-scale spatio-temporal features. The lightweight design of the STC block is the key reason behind MotionNet speed efficiency, enabling it to run in real-time. The decoder part consists of deconvolution, concatenation, and two 2D convolutions layers. The deconvolution layers are used to scale up the input features to allow concatenation with the features coming from the STC blocks. Global temporal pooling is used to assist in fusing multi-stage temporal features while going up the decoder part of the STPN. This design promotes the extraction of local and global spatio-temporal information. Figure 3.4 presents the architecture of MotionNet.

The MotionNet output heads are: 1) cell classification – for perceiving the category of pixels; 2) motion prediction – for predicting pixels motion; 3) state estimation – for predicting whether the pixels are static or dynamic. The three output heads constitute two 2D convolution layers each to acquire BEV pixel-wise predictions. The cell classification head classifies pixels from 5 category groups: background, vehicles, pedestrians, bikes, and others. The *others* category is assigned to detect objects that are not categorized in any of the remaining four groups. Therefore, the output dimension of the cell classification head is $256 \times 256 \times 5$. The motion prediction head predicts pixel positions for a sequence of 20 frames into the future (translating into 1 second); thus, the dimension of its output head is $256 \times 256 \times 2 \times 20$. Lastly, the output dimension of the state estimation head is $256 \times 256 \times 2$ because it predicts whether each pixel in the BEV image is static

or dynamic.

MotionNet loss function, defined in (3.2), consists of six components. Three of which are dedicated for global regularization of network training. These components are linked to the three output heads (cell classification loss $\mathcal{L}_{class}$, motion prediction loss $\mathcal{L}_{motion}$, and state estimation loss $\mathcal{L}_{state}$). Spatial consistency loss $\mathcal{L}_s$, foreground $\mathcal{L}_{ft}$ and background $\mathcal{L}_{bt}$ temporal consistency losses are the other three components dedicated for local regularization. $\mathcal{L}_{class}$ and $\mathcal{L}_{state}$ are cross-entropy losses, while $\mathcal{L}_{motion}$ is weighted smooth L_1 loss. In cross-entropy, a distinct weight is assigned to each category to handle the class imbalance issue. On the other hand, the remaining loss components ($\mathcal{L}_s$, $\mathcal{L}_{ft}$, and $\mathcal{L}_{bt}$) are associated with local regularization. The constants b_α, b_β, and b_γ in (3.2) are balancing factors.

$$\mathcal{L} = \mathcal{L}_{class} + \mathcal{L}_{motion} + \mathcal{L}_{state} + b_\alpha \mathcal{L}_s + b_\beta \mathcal{L}_{ft} + b_\gamma \mathcal{L}_{bt}\,, \tag{3.2}$$

$\mathcal{L}_s$ loss component, defined in (3.3), limits the predicted motion of all pixels relating to an object obj_e in one frame. So, the overall motion of obj_e should be reflected by all pixels belonging to the same object.

$$\mathcal{L}_s = \sum_e \sum_{(i,j),(i',j') \in obj_e} \left\| PM_{i,j}^{(\tau)} - PM_{i',j'}^{(\tau)} \right\|, \tag{3.3}$$

where $PM_{i,j}^{(\tau)} \in \mathbb{R}^2$ is the predicted motion at pixel (i, j) in time τ. Comparing all $PM_{i,j}^{(\tau)}$ and $PM_{i',j'}^{(\tau)}$ is computationally expensive, hence only adjacent pixels are considered. $||\cdot||$ is the smooth L_1 loss.

Unlike $\mathcal{L}_s$, which restricts motion spatially, $\mathcal{L}_{ft}$ in (3.4) constrains the predicted motion temporally for foreground objects between adjacent frames. This is achieved by assuming that no sudden changes in motion will occur between two consecutive frames.

$$\mathcal{L}_{ft} = \sum_e \left\| PM_{obj_e}^{(\tau)} - PM_{obj_e}^{(\tau+\Delta t)} \right\|, \tag{3.4}$$

where $PM_{obj_e}^{(\tau)} \in \mathbb{R}^2$ denotes the overall motion of object obj_e, computed as follows: $PM_{obj_e}^{(\tau)} =$

$\sum_{(i,j)\in obj_e} PM_{i,j}^{(\tau)} / obj_{e_{pixels}}$, where $obj_{e_{pixels}}$ is the number of pixels representing obj_e.

Lastly, $\mathcal{L}_{bt}$ defined in (3.5) confines the temporal loss for static background pixels.

$$\mathcal{L}_{bt} = \sum_{(i,j)\in PM^{(\tau)} \bigcap T(\tilde{PM}^{(\tau-\Delta t)})} \left\| PM_{i,j}^{(\tau)} - Trans_{i,j}(\tilde{PM}^{(\tau-\Delta t)}) \right\|, \tag{3.5}$$

where $PM^{(\tau)}$ and $\tilde{PM}^{(\tau)}$ are motion predictions with current time being t and $t+\Delta t$, respectively. A transformation $Trans \in SE(3)$ is necessary to align $\tilde{PM}^{(\tau-\Delta t)}$ with $PM^{(\tau)}$. After applying $Trans$, the static background pixels in $Trans(\tilde{PM}^{(\tau-\Delta t)})$ and $PM^{(\tau)}$ will partially overlap.

3.4 Experiments and Results

3.4.1 Dataset

The dataset used to conduct our experiments is nuScenes [66]. It consists of 850 annotated scenes, with each having a continuous sequence of sweeps. Our training set consists of 500 scenes (17,065 sequences), while our validation and test set has 100 (1,719 sequences) and 250 scenes (4,309 sequences), respectively. The LIDAR sensor used in the nuScenes dataset is Velodyne HDL32E, consisting of 32 beams, and operates at 20Hz. The horizontal FOV of the LIDAR is 360°, while its vertical FOV ranges from $-30.67°$ to $10.67°$.

3.4.2 Training Setup

Each scene in the dataset is divided into several clips, where each clip consists of 5 consecutive (1 current and 4 previous) sweeps. The current sweeps are sampled at 2Hz for training and 1Hz for testing. The current sweeps for testing are sampled at a lower frequency to reduce the similarity between the clips. Additionally, the period between all sweeps in a clip is 0.2s. The initial learning rate is set at 1.6×10^{-3} and it decays every 10 epochs to end at 0.8×10^{-3}. Due to the limited hardware, all experiments are trained with a batch size of 4.

3.4.3 Evaluation Metrics

The metrics used in Table 3.1 to evaluate the success of the experiments are categorized into two groups: displacement error and classification accuracy. The displacement error measures the correctness of motion prediction, while the classification accuracy accesses the perception level. Each of these two metric groups is further divided into subgroups. The displacement error is divided into three-speed subgroups: static, slow, and fast. Mean and median errors are computed for each speed subgroup using L_2. The static subgroup is defined for pixels with no motion, while the slow and fast subgroups are defined for pixels having speeds of $(0m/s, 5m/s]$ and $(5m/s, 20m/s]$, respectively. As mentioned earlier, motion is predicted in a sequence of 20 frames, translating into 1s into the future. Results in Table 3.1 are extracted from the last predicted frame. The speed is computed using the displacement error and LIDAR frequency.

Pixels are categorized into five object classes. In addition to the classification accuracy of each object class, the mean classification accuracy (MCA) and overall accuracy (OA) are also computed under the classification group. MCA in (3.6) denotes the average classification accuracy of the five category classes, while OA in (3.7) is the average classification accuracy over all pixels. Furthermore, the inference time for each experiment is measured.

$$MCA = \frac{1}{Nclasses} \sum_{j=1}^{Nclasses} \frac{(Total\ \#of\ correct\ predictions)_{Class_j}}{(Total\ \#of\ ground\ truths)_{Class_j}}, \tag{3.6}$$

$$OA = \frac{1}{Kimages} \sum_{k=1}^{Kimages} \frac{(\#of\ correct\ predictions)_{image_k}}{Npixels_{image_k}}, \tag{3.7}$$

where $Class_j$ denotes the category class j, while $Nclasses$ is the total number of category classes; $Npixels_{image_k}$ refers to the number of pixels in image k ($image_k$), and $Kimages$ is the total number of images in a dataset. The total number of predictions or ground truths for $Class_j$ are measured using all images in a dataset $Kimages$. Only non-empty pixels are considered for evaluation.

3.4.4 Experimental Setup

MotionNet confirms that using bigger BEV dimensions than $256 \times 256 \times 13$ accumulates additional computational cost without promising any performance improvements. Similarly, using 5 frames to reflect the temporal informal manifests a good trade-off between efficiency and accuracy. Thus, our BEV input dimension is $256 \times 256 \times 13 \times 5$. In this chapter, experiments are conducted to measure the effect of fusing RV with BEV features. Several experiments are conducted to investigate the effect of RV width dimensions on perception and motion prediction accuracy performance. The three RV width dimensions experimented on are: 2048, 1024, and 512. The length of the RV images is fixed at 32, relating to the number of laser beams in the LIDAR, and the depth is set to be 4, representing range, height, intensity, and flag information. The maximum RV width dimension experimented on is 2048 due to the limited computational power. Furthermore, a comparison is given between our proposed approach and other state-of-the-art approaches.

3.4.5 Results

Table 3.1 shows the results concerning perception and motion prediction for MotionNet and our proposed extension. As evident from Table 3.1, our proposed approach resulted in higher classification accuracy and lower displacement error than MotionNet, meaning enhanced perception and motion prediction. Based on the attained results, the fusion of BEV and RV features results in performance gain, regardless of the RV width dimension. However, the conducted experiments show that the highest classification accuracy and motion prediction is registered for RV images represented with bigger width dimensions. In our case, the biggest RV width dimension experimented on is 2048. An average improvement of 3.8% is obtained across all categories and 0.6% in the overall pixel accuracy. Also, the classification accuracy for smaller objects is more apparent with our proposed approach. A maximum increase of 7.6% and 3.0% is achieved for pedestrians and bikes, respectively, while only 1.5% is registered for vehicles. These results confirm that incorporating RV images leads to enhancement in perception. In terms of displacement errors, not only did our model ($2048 \times 32 \times 4$) acquire inferior mean and median errors in most speed groups compared to MotionNet but also reaped the least among the other experiments of smaller RV dimensions. Even though the displacement error for the slow group in the experiment with RV width

Table 3.1 Comparison in terms of perception and motion prediction between MotionNet and our proposed extension that includes multi-view fusion. Different width dimensions for RV representation are considered. The length and depth of RV images are fixed at 32 and 4, respectively.

Method	Static		Slow ($\leq$ 5m/s)		Fast ($>$ 5m/s)		Classification Accuracy (%)							Time
	Mean	Median	Mean	Median	Mean	Median	Bg	Vehicle	Ped.	Bike	Others	MCA	OA	(ms)
MotionNet [34]	0.0236	0	0.2534	**0.0959**	1.0778	0.7346	97.5	91.2	74.9	22.1	65.9	70.3	96.3	**20.5**
Ours (2048)	**0.0215**	0	0.2510	0.0961	**1.0200**	**0.7032**	97.9	92.7	82.5	25.1	72.1	**74.1**	**96.9**	33.4
Ours (1024)	0.0227	0	0.2497	0.0967	1.0360	0.7056	97.9	92.5	82.2	23.4	70.9	73.4	96.8	28.3
Ours (512)	0.0237	0	**0.2466**	0.0969	1.0554	0.7371	97.9	92.4	81.4	23.0	70.4	73.0	96.8	25.7

dimensions of 2048 is not minimum, the results are still competitive. Reduction in displacement error indicates advancement in motion prediction. Hence, the results demonstrated in Table 3.1 conclude that fusing BEV and RV features secure enhanced joint perception and motion prediction compared to MotionNet.

Table 3.2 further examines the classification accuracy, but this time based on distance from the LIDAR sensor. The detections are divided into 3 ranges: short ($\leq$10m), medium (11m-20m), and far ($>$20m). Naturally, the detection rate decreases with increasing distance, and this aligns with the findings in Table 3.2, where the detection accuracy decreases with a farther distance from the sensor. However, it is clear that the drop in detection accuracy is even less with our proposed approach. For example, the accuracy drop for vehicles between the short- and far-range is 12.7% for MotionNet, and 8.6% for our multi-view fusion model (2048).

Furthermore, our proposed model, with RV width of 2048, delivered substantial detection accuracy for distant objects. As an example, the vehicles category attained a gain of 4.1% in the far-range. Smaller objects obtained even better gain with farther distance. A rise of 7.1% is realized in the far-range for pedestrians and a pinnacle of 4.7% for bikes. These results show that multi-view fusion enhances perception and motion prediction and, more specifically, improves performance for small and distant objects, which is crucial for the safety and reliability of autonomous driving.

Although fusing BEV and RV features increased the latency over MotionNet; however, our proposed solution can still be deployed in real-time applications like autonomous driving as the recorded inference speed is within the real-time requirement (50ms). Finally, Figure 3.5 demon-

Table 3.2 Evaluation of perception based on different distance ranges from the LIDAR sensor: Short (S) - (0m, 10m], Medium (M) - (10m-20m], and Far (F) - (20m, 30m].

Method	Classification Accuracy (%)														
	Background			Vehicle			Pedestrian			Bike			Others		
	S	M	F	S	M	F	S	M	F	S	M	F	S	M	F
MotionNet [34]	98.3	97.3	95.8	94.5	91.0	81.8	77.4	74.5	73.3	29.1	19.5	17.0	76.1	62.6	55.2
Ours (2048)	**98.5**	**97.7**	**96.7**	94.5	**93.2**	**85.9**	**86.2**	**81.7**	80.4	28.8	**24.0**	**21.7**	**80.2**	**70.2**	**62.4**
Ours (1024)	98.5	97.6	96.5	**94.7**	92.6	85.8	85.9	81.1	**80.6**	**31.3**	18.8	21.7	79.5	68.9	60.3
Ours (512)	98.5	97.7	96.7	94.7	92.5	85.2	85.0	80.3	80.0	29.4	22.2	18.9	78.8	68.3	60.1

Table 3.3 Performance comparison with other state-of-the-art methods.

Method	Static		Slow		Fast		Class. Acc. (%)	
	Mean	Median	Mean	Median	Mean	Median	MCA	OA
FlowNet3D (pretrained) [51]	2.0514	0	2.2058	0.3172	9.1923	8.4923	-	-
FlowNet3D [51]	0.0410	0	0.8183	0.1782	8.5261	8.0230	-	-
HPLFlowNet (pretrained) [52]	2.2165	1.4925	1.5477	1.1269	5.9841	4.8553	-	-
HPLFlowNet [52]	**0.0041**	0.0002	0.4458	**0.0960**	4.3206	2.4881	-	-
PointRCNN [46]	0.0204	0	0.5514	0.1627	3.9888	1.6252	55.4	96.0
LSTM-Encoder-Decoder [53]	0.0358	0	0.3551	0.1044	1.5885	1.0003	69.6	92.8
Ours (2048)	0.0215	**0**	**0.2510**	0.0961	**1.0200**	**0.7032**	**74.1**	**96.9**

strates qualitative comparisons between MotionNet and our proposed approach (RV width of 2048). Three examples are presented, with each column denoting a scene. The first row shows the ground truth, second row has MotionNet predictions, and the last contains our proposed model's predictions. For easy comparisons, ground truths are also available in the second and third rows. The presented examples expose how our proposed approach generates more accurate detections and motion predictions compared to MotionNet. It is clear from the illustrated examples how our model perceives the surrounding environment more accurately and predicts motion closer to the ground truth compared to MotionNet.

Finally, further experiments are provided to prove the effectiveness of our proposed approach (RV width of 2048) against other state-of-the-art methods. Performance comparison for our proposed approach is performed against FlowNet3D [51], HPLFlowNet [52], PointRCNN [46], and LSTM-Encoder-Decoder [53]. In addition to the finetuned FlowNet3D and HPLFlowNet, results for their pretrained models are included. The two scene flow datasets used for the pretrained models are FlyingThings3D and KITTI Scene, while finetuned on nuScenes. FlowNet3D and

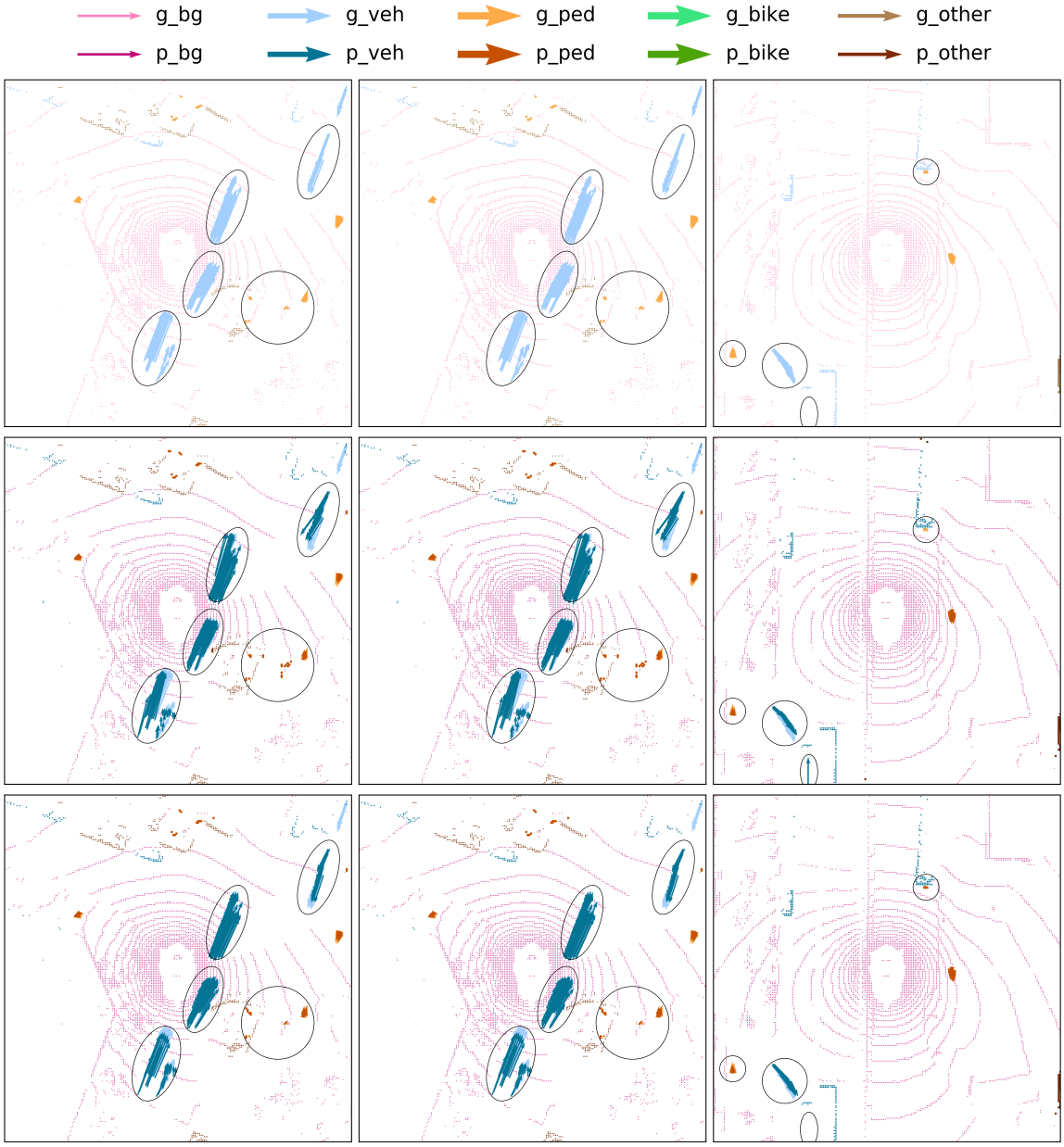


Figure 3.5 Qualitative analysis comparing MotionNet (second row) with our proposed multi-view fusion technique (third row). The ground truth images are available for reference (first row). Images in the second and the third row also incorporate the ground truth for easier comparison. Color codes are defined at the top of the figure where the prefixes $g\_$ and $p\_$ denote ground truth and prediction, respectively.

HPLFlowNet models estimate the scene flow between two point clouds, while PointRCNN predicts directly from a point cloud. LSTM-Encoder-Decoder estimates the multi-step 2D grid map representation of the point cloud. Table 3.3 reveals that our proposed approach exceeds the state-of-the-art in joint perception and motion prediction. Even though the mean error in the static and the median in the slow groups are not the lowest; nonetheless, the overall performance of our proposed approach largely outperforms the other state-of-the-art methods, especially in the fast group. All these comparisons collectively show the potential of our proposed approach in joint perception and motion prediction.

3.5 Summary

A multi-view LIDAR fusion model is proposed for joint perception and motion prediction in autonomous vehicles. RV features are extracted and projected onto BEV features to form a multi-view fusion input. Those rich features are then fed into MotionNet backbone to generate accurate pixel-wise joint perception and motion prediction in real-time. A LIDAR sensor was used to generate both views. We evaluated our work on nuScenes dataset and compared it with MotionNet and other state-of-the-arts. Experiments showed that the fusion of multi-view LIDAR images substantially increased accuracy for both perception and motion prediction, especially for distant and small objects. Lastly, it can be noted from the achieved results that the accuracy of bikes are the lowest, and this is because they are underrepresented in the nuScenes dataset. One solution to overcome this is to train a separate small network focused solely on bikes; afterward, essential features characterizing bikes are fused into the primary model to strengthen bike detection. Bike representation in a dataset can also be expanded using augmentation techniques. Finally, a confusion matrix can be leveraged to visually examine the network classification performance and analyze under which category bikes are misclassified.

CHAPTER 4
LICANET: FURTHER ENHANCEMENT OF JOINT PERCEPTION AND MOTION PREDICTION BASED ON MULTI-MODAL FUSION

This chapter expands on the network proposed in Chapter 3 by incorporating a camera image into the fusion of LIDAR sourced sequential bird's-eye view (BEV) and range view (RV) images. The network proposed in this chapter is named LIDAR Camera Network (LiCaNet) and achieves accurate pixel-wise joint perception and motion prediction in real-time. This chapter begins with a brief introduction about the advantages of fusing multi-modal features sourced from LIDAR and camera sensors, and the best representations for LIDAR raw data. Second, a concise review of works exploiting LIDAR and camera-based multi-modal features is provided. LiCaNet architecture is extensively discussed next. Subsequently, experimental results are presented to validate the outstanding performance of LiCaNet. Finally, the chapter is concluded with a summary.

In the following chapter, *LiCaNext* - the *next* version of LiCaNet, is introduced. *LiCaNext* combines sequential range residual images into the fusion of BEV, RV, and camera images. Chapter 6 proposes the utilization of multi-modal fusion to generate highly accurate predictions, extracted from *LiCaNext*, to train a sequential latent MaxEnt RL algorithm for learning a driving policy that can achieve the objectives of safety, efficiency, and comfort in an urban environment. Lastly, this Ph.D. thesis is concluded in Chapter 7.

4.1 Introduction

The field of autonomous driving had secured incremental progress over the past few years, especially around 2014, when deep learning blossomed. At that time, researchers started regaining hopes that the impediments in autonomous driving could be resolved with the help of innovative deep learning. As mentioned in Chapter 1, an autonomous vehicle consists of several pipelines ranging from perception, motion prediction, planning to control [10, 11, 12]. The two pivotal pipelines in autonomous vehicles are perception and motion prediction, as they allow the vehicle to observe the environment and forecast the dynamics in its surroundings. All subsequent

pipelines rely on the accuracy of both perception and motion prediction. Without these pipelines, an autonomous vehicle cannot operate safely and reliably. Moreover, fundamental to perception and motion prediction pipelines are their input data provided by sensors.

In general, an autonomous vehicle is equipped with a suite of different sensors [151]. Although each sensor has advantages and disadvantages, combining data features from several sensors provides complementary information by reaping the benefits of all employed sensors and mitigating the inherent challenges of individual sensors. This chapter focuses on fusing LIDAR and camera features; thus, first, a brief comparison between LIDAR and camera sensors is given, then how multi-modal fusion leads to performance advancement is presented. A LIDAR sensor is designed to capture precise depth and physical information of the surrounding environment. LIDAR can operate under hazardous weather conditions; however, they are expensive and require more space to implement on vehicles. On the other hand, a camera sensor acquires color information offering rich semantic images. Similar to our vision, cameras can quickly identify the type of object based on colors. Unlike LIDAR, a camera is ineffective at defining object ranges, capturing images under poor illumination scenarios as well as severe weather conditions.

Fusing precise range and geometric measurements from a LIDAR and rich semantic information from a camera yields an integral set of features resistant to the limitations manifested by individual sensors. For example, when a camera captures semantic features concerning an object, the existence of LIDAR data complements those features by adding the object's depth and physical dimensions. Additionally, small and distant objects are naturally represented by few LIDAR points, and even a camera captures inadequate semantics for such objects. However, when these features are aggregated, the representation of such objects is strengthened.

LIDAR is the most common sensor employed in autonomous vehicles, and several representations exist in the literature for its data. The prominent LIDAR data representations include point-based form [46, 47], 3D voxelization [48, 49], BEV [152, 146, 38], and RV [68, 67, 63]. In this chapter, on top of multi-sensor fusion, we take advantage of fusing multi-view LIDAR data representation. Undoubtedly, each LIDAR representation has its benefits and drawbacks. However, as discussed in Chapter 3, the two most efficient and effective are BEV and RV forms. These two

representations overcome most of the limitations found in the other LIDAR-based representations, and further, they encompass additional properties that are valuable to the learning model. A brief recap, BEV and RV are compact 2D image projections of the LIDAR point cloud, inexpensive to generate, and efficient to process using 2D convolutions. In addition, BEV simplifies the process of adding historical information and preserves object dimensions making learning easier. Lastly, RV preserves occlusion and high-resolution point information. Accordingly, exploiting both BEV and RV form in a fusion network is computationally inexpensive and offers valuable, constructive features procured from one sensor, enabling a deeper understanding of the scene.

Based on recent works [152, 38, 39, 40, 41, 42], it was proven that the benefits of exploiting multi-modal fusion for perception and motion prediction in autonomous driving are substantial. To the best of our knowledge, no other work explores the multi-modal fusion of historical BEV, RV, and camera features to address the issue of pixel-wise joint perception and motion prediction in real-time. In light of the above observations, this chapter proposes a novel LIDAR Camera Network (LiCaNet), which expands on the fusion network of [152] (Chapter 3) to involve a camera in addition to a LIDAR sensor. Hence, a new camera module is added to extract relevant semantic features from camera images, enabling fusion with BEV and RV features. Figure 4.1 illustrates the architecture of our proposed LiCaNet multi-modal fusion network. LiCaNet aims to generate rich and complementary features constituting: temporal, depth, and object sizes encoded in BEV form; occlusion and high-resolution point information embodied in RV; and semantic information characterized in a camera image. After engendering the multi-modal features, they are fed into a backbone network to attain an enhanced joint perception and motion prediction model, especially for small and distant objects. These overarching set of multi-modal features engendered by LiCaNet feeds the backbone network a vivid and knowledgeable image of the surrounding scene, resulting in improved accuracy. The backbone network used is MotionNet, which is the same as the one used in Chapter 3.

4.2 Review

As presented in the Related Work chapter (Chapter 2), the utilization of multi-modal fusion for perception and motion prediction has recently gained much attention among researchers in

autonomous driving. Plenty of works exist that use multi-modal fusion sourced from LIDAR and camera sensors to just perform perception using 3D object proposals [39, 84, 85, 41, 40, 86, 87]. However, Fadadu et al. [38] proposed a multi-modal fusion model (BEV, RV, camera, and HD maps) for joint perception and motion prediction. LIDAR and camera sensors are used in [38] to perform 3D object level predictions. It is evident that no work has yet been conducted that investigates pixel-wise joint perception and motion prediction using multi-modal fusion to generate fine-grained, pixel level precision, which is essential for predicting small and distant objects.

To that end, this chapter proposes LiCaNet, a multi-modal fusion network to generate accurate pixel-wise perception and motion prediction. LiCaNet fuses camera features with LIDAR-based historical BEV and RV features. LiCaNet engenders multi-modal features that embrace: 1) temporal, depth, and physical object dimensions in BEV form; 2) occlusion and high-resolution point information in RV form; and 3) semantics in camera images. These rich and integral features enhance the accuracy of perception and motion prediction, especially for small and distant objects.

4.3 Proposed Methodology

The overview of our proposed LiCaNet model is shown in Figure 4.1. LiCaNet is designed to incorporate features from LIDAR and camera sensors to produce complementary multi-modal features. The LIDAR data is represented in sequential BEV and RV images. LiCaNet consists of three modules: BEV, RV, and camera. The BEV and RV modules represent the network presented in [152] (Chapter 3), and the camera module is the proposed expansion resulting in LiCaNet. The camera module accepts camera images and extracts relevant features to be fused with the outcomes of the BEV and RV modules for further performance enhancement. The multi-modal features generated by LiCaNet are then used as input to MotionNet backbone network for pixel-wise joint perception and motion prediction. Similar to [152], nuScenes dataset [66] is used to evaluate LiCaNet, as it consists of large amounts of high-quality camera data designed for autonomous driving in addition to LIDAR point clouds. The methodologies used to analyze LiCaNet predictions are classification accuracy for perception and displacement error for motion prediction. LiCaNet performance outperforms [152] and MotionNet. In Chapter 3, it was proven that [152] outperforms state-of-the-art methods; thus, as LiCaNet extends [152] and outperforms it then by default it is

better than the state-of-the-art. Furthermore, LiCaNet operates in real-time, making it suitable for autonomous driving.

4.3.1 LIDAR Input Representation

The two LIDAR representations used in LiCaNet are BEV and RV. The formulation of both LIDAR input representations are computed as explained in Section 3.3.1. In this chapter, the RV image is designed to have fixed dimensions of $1024 \times 32 \times 4$. Although in the previous chapter, it was proved that the wider the RV image, the better the performance; however, in this chapter, we select the RV width dimension to be 1024 instead of 2048 because this is the maximum our limited hardware can handle to load and train LiCaNet.

4.3.2 Camera Input Representation

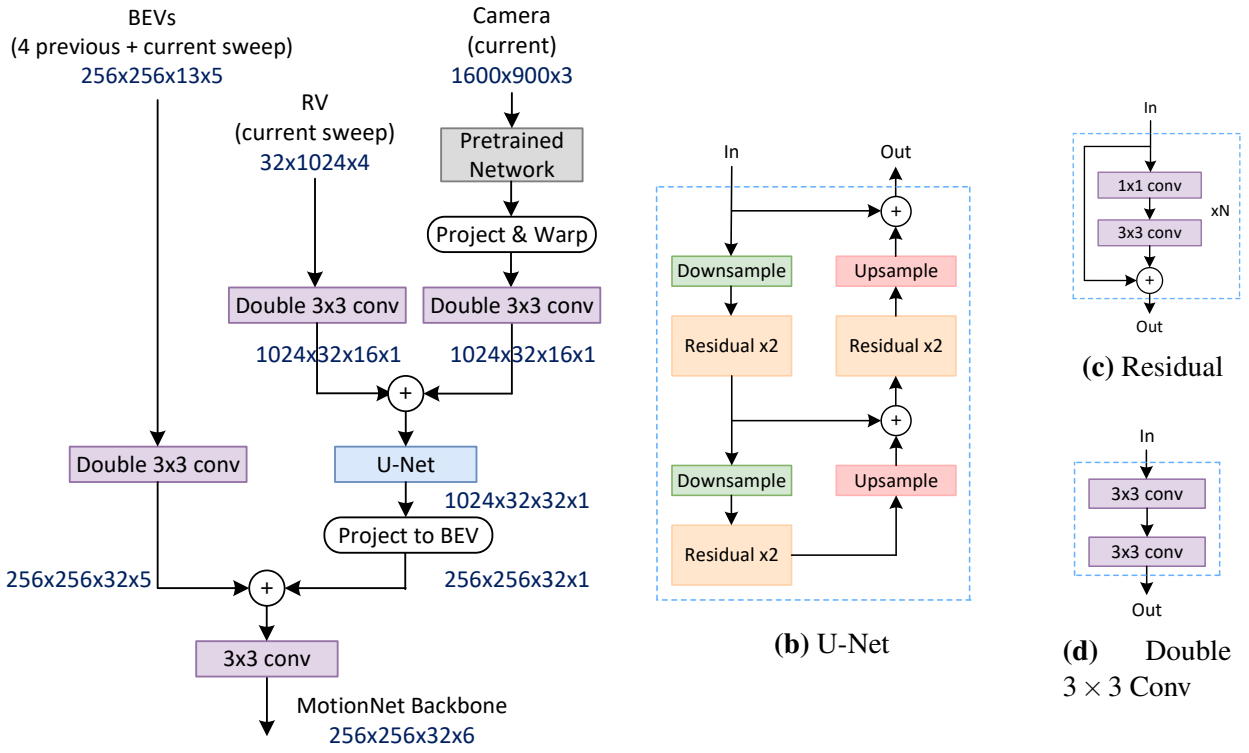
A camera sensor captures information with color encodings, offering semantically rich images. From the nuScenes dataset, the front camera images are used as input to the LiCaNet camera module. The RGB camera images are of dimensions $1600 \times 900 \times 3$. Figure 4.2b shows a sample camera image captured at the exact timestamp as the LIDAR sweep that is used to compute the current BEV and RV images.

4.3.3 LiCaNet Architecture

As aforementioned, the significance of fusing multi-modal features is to extract complementary information that contributes to producing improved perception and motion prediction. A LIDAR is adopted in LiCaNet primarily due to its capability in capturing precise depth information. Moreover, a front camera is employed for its dense semantic features. A comparison between the advantages and disadvantages of the selected LIDAR-based representations (BEV and RV) is given in Section 3.1. Consequently, concatenating features from BEV, RV, and camera representations produce complementary features that leverage all representations' benefits and mitigate the drawbacks of individual representations. Therefore, the performance advancements reported in this chapter are attributed to the fusion of camera semantics into BEV and RV features.

The architecture of LiCaNet is depicted in Figure 4.1. The proposed fusion scheme consists of three key modules: BEV, RV, and camera. A sample of input data to the LiCaNet modules is illustrated in Figure 4.2. Starting with the BEV module, once all LIDAR sweeps are transformed into BEV representation through synchronization and discretization, the aggregated BEVs are sent down a two 3×3 convolution layers, named *Double* 3×3 *conv*. Concurrently, the RV image and camera features representing only the current timestamp are also passed down a *Double* 3×3 *conv* independently. However, before applying the RGB camera image directly to *Double* 3×3 *conv*, the RGB image is first passed to a small pretrained network for high-level feature extraction. Then the high-level features are projected and warped into RV representation. Next, the resulting features from the RV and camera modules are concatenated and applied to a U-Net [150]. U-Net is an encoder-decoder network with a strong representation ability mainly because of the skip connections that combine shallow features from the encoder path with deep features from the decoding path at their respective stages. The features resulting from the U-Net are in RV form and thus need to be projected to BEV representation to complete the fusion process. The subsequent step is to concatenate the projected features, in BEV representation from the RV and camera modules, with the features from the BEV module. The last step in the fusion process is to feed the resulting multi-modal features into a single 3×3 convolution layer. This proposed LiCaNet fusion process generates rich complementary features that enable us to achieve enhanced performance. Finally, the generated multi-modal features are then applied to MotionNet backbone network to perform accurate pixel-wise joint perception and motion prediction in real-time. The backbone network is extensively explained in Section 3.3.4.

Directly feeding the fusion network with raw RGB features causes the learning network to discard most features as they do not comprise valuable high-level information. Thus, the camera image is first passed to a pretrained network to extract high-level features, which are later fed into the fusion network. Now, to use the extracted high-level camera features in our fusion process, we need a mapping from the LIDAR points to the camera image. This mapping permits the retrieval of the high-level camera features corresponding to LIDAR points residing in the camera's FOV. In summary, once a mapping from LIDAR points to camera features is computed, then it is possible to warp and project features from the camera image into RV image. The mapping of each LIDAR



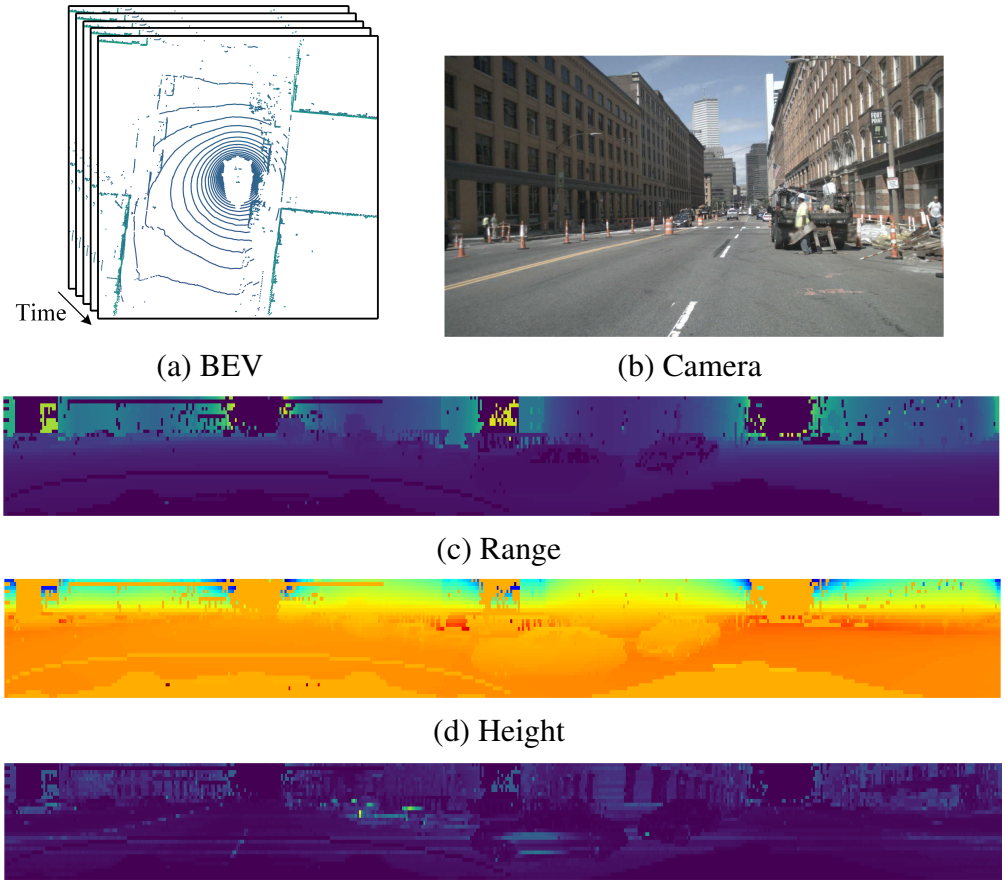
(a) LiCaNet Fusion Scheme

(b) U-Net

(c) Residual

(d) Double 3×3 Conv

Figure 4.1 LiCaNet architecture. The input is composed of 5 sequential BEV images, an RV, and a camera image. LiCaNet is composed of three modules. The BEV and RV modules consist of double 3×3 convolution layers (d). The camera module consists of a pretrained network, projecting and warping the camera features into RV form and double 3×3 convolution layers. The RV features from both the RV and camera modules are concatenated and fed into U-Net (b). The U-Net consists of residual blocks (c) and upsample and downsample blocks of scale 2. The output of the U-Net, in RV form, is projected onto BEV and concatenated with the features from the BEV module to be finally fed into a single 3×3 convolution layer. Finally, the LiCaNet output is fed into MotionNet backbone for joint perception and motion prediction.



(a) BEV (b) Camera

(c) Range

(d) Height

(e) Intensity

Figure 4.2 A sample of LiCaNet input features. (a) illustrates the historical BEV images; (b) the camera image; (c), (d), and (e) represent three channels of the RV image.

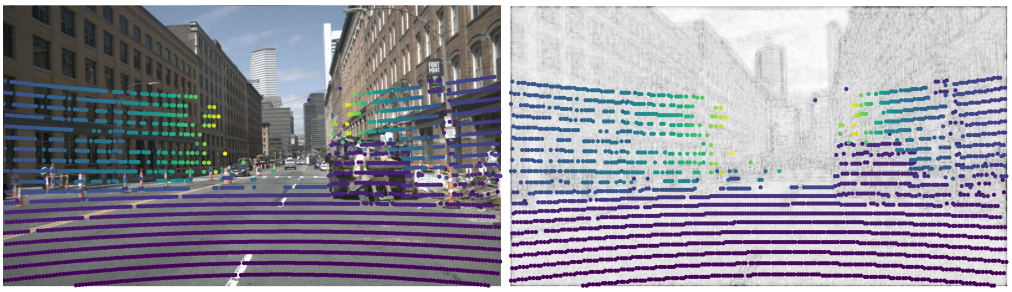


Figure 4.3 Examples of LIDAR points' range values projected onto the camera image. The left image represents the LIDAR points projected on the raw camera image. The right image shows the LIDAR points projected on the outcome of the lightweight pretrained network (in grey scale).

point p_i onto a camera image is achieved by the transformation $Trans_{C \leftarrow L}$ defined in (4.1).

$$Trans_{C \leftarrow L} = \mathcal{K} \left(Trans_{C \leftarrow veh_{t_C}} * Trans_{veh_{t_C} \leftarrow veh_{t_L}} * Trans_{veh_{t_L} \leftarrow L} \right), \tag{4.1}$$

where subscripts C, L, and veh stands for camera, LIDAR, and vehicle, respectively. $\mathcal{K}$ is the intrinsic calibration matrix of the camera. In nuScenes dataset, LIDAR and camera sensors have different operational frequencies and so before transforming LIDAR points to the camera's coordinate system they need to be mapped to the vehicle's coordinate system to compensate for the time-shift between the two sensors. $Trans_{veh_{t_L} \leftarrow L}$ transforms LIDAR points to the vehicle's frame at LIDAR capture time t_L, $Trans_{veh_{t_C} \leftarrow veh_{t_L}}$ transforms the points from vehicle's frame at LIDAR capture time t_L to camera capture time t_C. Last, $Trans_{C \leftarrow veh_{t_C}}$ transforms the points from vehicle's frame at t_C to the camera's coordinate system.

The complete mapping equation that maps LIDAR points $\hat{p}$ onto the camera coordinate system is defined in (4.2).

$$[u_C \; v_C \; 1]^T = Trans_{C \leftarrow L} \left(\hat{p} \right), \tag{4.2}$$

where (u_C, v_C) are the mapped points from the LIDAR's coordinate system onto the camera. An example of LIDAR points' range values being mapped using (4.2) and projected onto camera coordinate system is shown on the left image of Figure 4.3. The projection algorithm is explained later.

Ultimately, the extracted high-level camera features need to be fused with RV features. Thus,

using the mapping computed in (4.2) between the LIDAR points and the camera features, it becomes possible to project the camera features into RV representation. Up to this point, we assumed that the features extracted from the pretrained network have the same dimensions as the original camera image. Unfortunately, this is not always the case, so to resolve this issue, we need to update the mapping between LIDAR points and camera image pixels with a scale factor as expressed in (4.3).

$$\begin{aligned} u_{C_{scaled}} &= u_C * scale_l^{-1} \quad \text{and} \quad scale_l = (l_C / l_{hlcf}), \\ v_{C_{scaled}} &= v_C * scale_w^{-1} \quad \text{and} \quad scale_w = (w_C / w_{hlcf}), \end{aligned} \tag{4.3}$$

where $(u_{C_{scaled}}, v_{C_{scaled}})$ are the scaled mapped points on the camera image. (l_C, w_C) represents the length and width of the original camera image, while (l_{hlcf}, w_{hlcf}) denotes the resolution of the high-level camera features resulting from the pretrained network. The right image of Figure 4.3 illustrates the projected LIDAR points' range values on the high-level features resulting from the pretrained network, using mapping in (4.2) and scaling in (4.3). The lightweight pretrained network used to extract high-level features from camera images is described in Section 4.4.4.

The projection of features from one form to another is accomplished using the painting technique [86]. Briefly, the algorithm takes the mean of all features from the source representation that corresponds to a LIDAR point $\hat{p}_i$ and projects them to the cell position in the target representation where $\hat{p}_i$ is linked. If no features are projected into a cell in the target representation, the cell value remains -1. The same algorithm is repeated for all points that have a mapping between the source and target forms. Figure 4.4 demonstrates an illustrative example of the adopted projection algorithm. Firstly, the mapping for each LIDAR point $\hat{p}_i$ is computed in both the source and target forms (denoted by blue and orange arrows). Next, all features in the source form where $\hat{p}_i$ is linked to (denoted by $s1_{\hat{p}_i}$, $s2_{\hat{p}_i}$, and $s3_{\hat{p}_i}$) is averaged. Eventually, the averaged features are positioned in the target's cell where $\hat{p}_i$ is linked (orange arrow). Figure 4.5 displays features from the camera image projected into RV form. The number of LIDAR points mapped between the camera and RV forms determine the resolution of the target RV image.

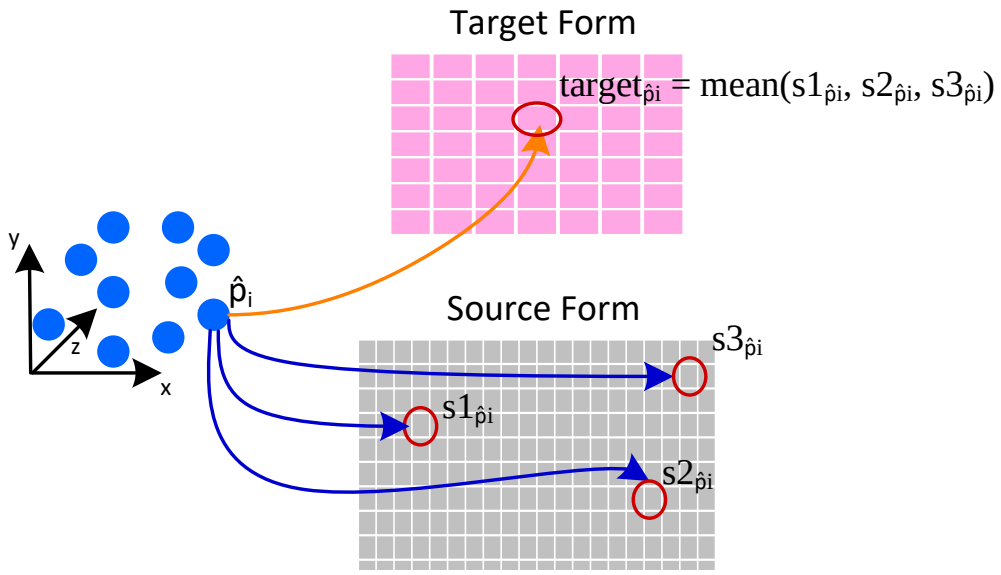


Figure 4.4 Illustration of the projection algorithm from source to target form. $target_{\hat{p}_i}$ is the average of all features that LIDAR point $\hat{p}_i$ is mapped to in the source form ($s1_{\hat{p}_i}$, $s2_{\hat{p}_i}$, and $s3_{\hat{p}_i}$).

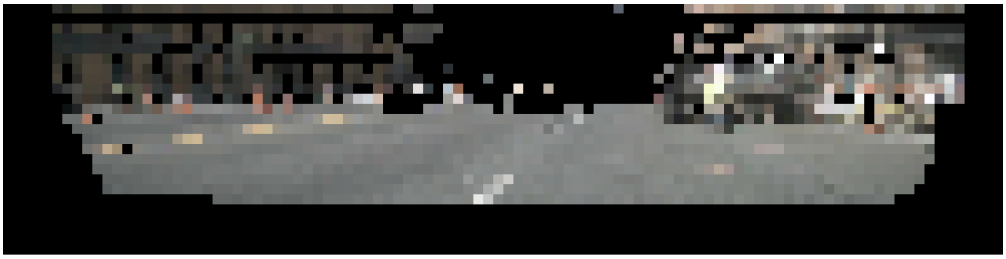


Figure 4.5 Example of the front camera image projected into RV representation. The RV image has been cropped to present only the area that contains the projected camera features. The rest of the RV image is empty because the camera FOV is 70°, while the horizontal LIDAR FOV is 360°.

4.4 Experiments and Results

This section begins with a brief description of the dataset used. Second, the set of experiments performed to validate the significance of multi-modal fusion for joint perception and motion prediction are defined. Details about the training setup are illustrated next. Extensive quantitative and qualitative results are provided to confirm the effectiveness of LiCaNet.

4.4.1 Dataset

This chapter conducts experiments on the same dataset used in the previous chapter. A detailed description of the nuScenes dataset can be found in Section 3.4.1. As LiCaNet involves camera images, the characteristics of those images are defined in this section. Images from only the front camera sensor are used in LiCaNet. The camera sensor captures images at 12Hz with an opening angle of 70°.

4.4.2 Training Setup

For fair comparisons, the setup used for conducting LiCaNet experiments is the same as the setup described in Section 3.4.2. Briefly, the learning rate is initialized at 1.6×10^{-3} and terminated at 0.8×10^{-3}, with a decay factor of 0.5 every 10 epochs. The time span between consecutive LIDAR sweeps used to construct the historical BEVs is 0.2s. The current sweep is sampled at 2Hz for training and 1Hz for testing. The batch size used to train LiCaNet is 4.

4.4.3 Evaluation Metrics

The exact evaluation metrics presented in Section 3.4.3 are used for evaluating LiCaNet. The metrics used to evaluate our model's perception are overall accuracy (OA) and mean category accuracy (MCA). OA is the average classification accuracy over all pixels, whereas MCA is the average accuracy over the category classes. L_2 displacement error at 1s time horizon is used to evaluate motion prediction. The displacement error is evaluated on pixels based on three-speed groups: static ($0m/s$), slow ($\leq 5m/s$), and fast ($> 5m/s$). The speed is computed using the displacement error and LIDAR frequency. The mean and median error is reported for each speed category.

4.4.4 Experimental Setup

Similar to Chapter 3, the BEV images of LiCaNet are set to have dimensions of $256 \times 256 \times 13 \times 5$. In terms of the RV dimensions, it was proven in Section 3.4.5 that using RV images with a width dimension of 2048 for fusion leads to better joint perception and motion prediction compared to smaller RV dimensions. However, in this chapter, the RV width dimension is selected as 1024 rather than 2048 due to our limited computational power. Thus, the overall dimensions of the LiCaNet RV images are set to be $1024 \times 32 \times 4$. The length of the RV images is attributed to the number of laser beams of the LIDAR sensor used in the nuScenes dataset. Moreover, the 4 RV channels constitute range, height, intensity, and a binary flag indicating the validity of cells. Finally, the dimensions of the RGB camera images are $1600 \times 900 \times 3$. The dimensions of all input representations are illustrated in Figure 4.1.

The experiments conducted in this chapter include the baselines MotionNet and its successor - the multi-view fusion network [152] (presented in Chapter 3). Since our earlier work [152] has shown its success over state-of-the-art models; thus, in this chapter, we build on that analysis and limit the comparison of LiCaNet to MotionNet and its successor [152]. As explained, LiCaNet experiments are conducted with an RV width dimension of 1024; thus, the experiment from the previous chapter used to act as baseline must have the exact RV dimensions. Additionally, the *Double* 3×3 *conv* block of the RV module in the previous chapter, uses 32 channels to encode features. In order to load and train LiCaNet on our machine, we narrow down the depth of the RV module's *Double* 3×3 *conv* block to 16 channels. Thus, in order to compare LiCaNet with the matching experiment of the previous chapter, we retrain the fusion of historical BEV and RV images with a depth configuration of 16 for the RV module's *Double* 3×3 *conv* block. It is worth mentioning that LiCaNet minus the camera module matches the fusion network of the previous chapter. Thus, we name this experiment LiCaNet (LIDAR only). The comparison with these two experiments is essential to verify that LiCaNet (fusion of BEV, RV, and camera) outperforms both our earlier work (fusion of BEV and RV) and MotionNet (BEV only).

The next set of experiments analyzes the performance of LiCaNet under various lightweight pretrained networks. These experiments are used to monitor whether the performance gain of Li-

Table 4.1 Comparison of perception and motion prediction results between MotionNet, multi-view LIDAR-based fusion, and the proposed LiCaNet model.

Method	Static		Slow ($\leq$ 5m/s)		Fast ($>$ 5m/s)		Classification Accuracy (%)							Time
	Mean	Median	Mean	Median	Mean	Median	Bg	Vehicle	Ped.	Bike	Others	MCA	OA	(ms)
MotionNet [34]	0.0236	0	0.2534	**0.0959**	1.0778	0.7346	97.5	91.2	74.9	22.1	65.9	70.3	96.3	**20.5**
LIDAR fusion [152]	0.0227	0	0.2497	0.0967	1.0360	0.7056	97.9	92.5	82.2	23.4	70.9	73.4	96.8	28.3
LiCaNet (LIDAR only)	0.0230	0	0.2531	0.0964	1.0547	0.7305	97.6	92.0	80.6	22.6	69.2	72.4	96.6	28.1
LiCaNet (MobileNetv2_6)	0.0224	0	**0.2504**	0.0964	**1.0432**	0.7304	97.9	92.8	82.7	23.0	70.7	73.4	96.8	30.7
LiCaNet (VGG16_6)	0.0224	0	0.2527	0.0969	1.0456	0.7300	97.8	92.4	84.0	23.2	71.9	**73.9**	**96.9**	31.0
LiCaNet (ResNet50_11)	0.0223	0	0.2530	0.0963	1.0479	0.7295	97.8	92.6	81.9	23.4	70.4	73.2	96.8	32.9
LiCaNet (ResNeXt50_11)	**0.0220**	0	0.2529	0.0963	1.0461	**0.7289**	98.0	92.7	81.5	23.6	70.6	73.3	96.9	33.2

CaNet is consistent across all pretrained networks. All selected lightweight networks dedicated to extracting high-level features from camera images are pretrained on ImageNet. The four pretrained networks investigated are MobileNetv2, VGG16, ResNet50, and ResNeXt50. These pretrained networks are chosen as they have shown challenging performance in image classification. As the proposed camera module requires only a lightweight pretrained network, only 6 convolution layers are employed from MobileNetv2 and VGG16; and 11 layers from ResNet50 and ResNeXt50.

4.4.5 Quantitative Results

Table 4.1 unveils the perception and motion prediction results of our conducted experiments. It is evident from the collected results in Table 4.1 that all LiCaNet experiments, with the different pretrained networks, compare favorably to MotionNet and LiCaNet (LIDAR only) - a narrowed version of the multi-view LIDAR-based fusion network [152]. For fair comparisons, the second experiment in Table 4.1 is excluded from our analysis as the depth of its RV module is wider than LiCaNet. Nevertheless, it was included to show that its narrower version has inferior performance in both perception and motion prediction. According to Table 4.1, the use of 6 convolution layers from pretrained VGG16 resulted in the best perception. Even though VGG16 did not attain the lowest displacement errors; however, VGG16 still achieved competitive motion predictions compared to the other pretrained networks. Comparing perception accuracy of LiCaNet (VGG16_6) with LiCaNet (LIDAR only) experiments, it is clear that the addition of the camera module achieved a substantial increase of 1.5% and 0.3% in MCA and OA, respectively. Moreover, a total gain of 3.6% in MCA and 0.6% in OA is registered relative to MotionNet.

In addition, examining the classification accuracy per category indicates that of all the selected pretrained networks, VGG16_6 is considered the best at detecting small objects. With only 6 convolution layers, VGG16_6 secured the highest detection accuracy for pedestrians and a competitive accuracy for bikes. In comparison, ResNeXt50_11 used 11 convolution layers to procure the maximum accuracy for bikes (23.6%), which is only 0.4% higher than what VGG16_6 accomplished. Also, the use of ResNeXt50_11 did not perform as well as VGG16_6 in detecting the remaining categories. Overall, VGG16_6 outperformed the other pretrained networks in perception. Further investigation into the classification accuracy results reveals that smaller objects have the highest perceptual gain compared to the other categories. LiCaNet (VGG16_6) resulted in a jump of 0.4% for vehicles, but a rise of 3.4% and 0.6% is procured for pedestrians and bikes, respectively, compared to LiCaNet (LIDAR only) experiment. The presented results in Table 4.1 confirm that LiCaNet experiments, for all different pretrained networks, achieved an enhancement in perception accuracy and prominent decrease in displacement error (i.e., increase in motion prediction) compared to [152] and MotionNet. Thus, the exploitation of camera images in the fusion process assists in achieving an enhanced perception and motion prediction model, with the highest advancement dedicated to small objects.

Table 4.2 further investigates the success of LiCaNet experiments by restricting the perception accuracy to within the camera FOV. Furthermore, the results are measured based on the distance from the camera sensor. For each object category, perception is measured within the camera FOV at three distance ranges: short-range (S) defined from 0m-10m, medium-range (M) from 11m-20m, and far-range (F) from 21m-30m. Considering the perception accuracy between LiCaNet, LiCaNet (LIDAR only), and MotionNet experiments, it is obvious that within the camera FOV, the accuracy is higher for LiCaNet experiments, with the highest rise recorded for small and distant objects.

To begin with, comparing the gain of vehicles between LiCaNet (VGG16_6) and LiCaNet (LIDAR only), a rise of 0.7% and 1.4% is attained for short- and far-range, respectively. Similarly, for pedestrians, an increase of 1.7% and 1.9% is procured in the same range groups, respectively. This shows that the proposed LiCaNet can gather higher accuracy for distant objects. Moreover, smaller objects collected even greater gain in perception within the camera FOV. For example, in the far-

Table 4.2 Evaluation of classification accuracy within the camera FOV based on distance ranges from the camera sensor. The three distance groups are short-range (S) defined from 0m-10m, medium-range (M) from 11m-20m, and far-range (F) from 21m-30m.

Method	Classification Accuracy in Camera FOV (%)														
	Background			Vehicle			Pedestrian			Bike			Others		
	S	M	F	S	M	F	S	M	F	S	M	F	S	M	F
MotionNet [34]	98.5	97.0	94.8	95.7	93.1	83.3	83.9	81.4	76.0	20.0	16.2	33.2	81.2	71.6	57.7
LIDAR fusion [152]	98.7	97.5	95.7	96.0	93.9	87.0	89.4	81.1	83.5	30.5	21.3	31.3	84.9	76.2	63.3
LiCaNet (LIDAR only)	98.6	97.1	95.2	94.5	93.3	86.1	88.6	77.0	82.1	26.2	20.1	31.0	83.4	75.2	61.7
LiCaNet (MobileNetv2_6)	98.8	97.5	95.6	95.2	94.6	87.7	90.0	86.6	**84.1**	28.6	21.9	**38.2**	83.3	75.6	66.4
LiCaNet (VGG16_6)	98.7	97.2	95.3	**95.2**	**94.6**	87.5	**90.3**	**86.7**	84.0	**35.2**	**23.6**	35.0	85.5	**79.5**	**67.7**
LiCaNet (ResNet50_11)	98.8	97.5	95.7	94.3	87.8	87.6	89.3	78.5	83.4	26.9	22.3	34.0	86.5	75.7	66.0
LiCaNet (ResNeXt50_11)	**98.8**	**97.5**	**96.0**	94.8	94.4	**87.9**	89.0	77.9	82.6	21.0	22.7	34.6	**87.4**	75.8	65.4

range 1.4% improvement is registered for vehicles, while 1.9% for pedestrians and 4.0% for bikes. This proves the potential of LiCaNet in detecting small and distant objects. Furthermore, a natural characteristic in any model is that the detection accuracy decreases with a farther distance from the sensor; nevertheless, the drop with LiCaNet is lower. The accuracy drop between the short- and far-range of vehicles, pedestrians, and bikes is 8.4%, 6.5%, and 4.8% for LiCaNet (LIDAR only); whereas, for LiCaNet (VGG16_6) the drop is merely 7.7%, 6.3%, and 0.2%, respectively. To summarize, results in Table 4.2 confirm that the perception accuracy of LiCaNet within camera FOV is substantially better than our earlier work, especially for small and distant objects. Although most of our analysis is limited between LiCaNet (VGG16_6) and LiCaNet (LIDAR only), comparing LiCaNet with MotionNet leads to even more significant gains.

Lastly, the inference time is naturally compromised when a fusion network is expanded by adding a camera module. Thus, a trade-off should be made between accuracy and inference time. Although our proposed LiCaNet model involves a camera module, all experiments with the different pretrained networks did not exceed the real-time requirement. Therefore, a safe conclusion can be drawn that LiCaNet is suitable for autonomous driving applications. According to the results in Table 4.1, incorporating a camera module onto our multi-view fusion network (presented in Chapter 3) results in a minimum increase of 2.6ms. This is recorded for MobileNetv2, where 6 convolution layers are utilized. VGG16_6 contains more parameters than MobileNetv2 and thus is heavier than MobileNetv2_6, explaining the reason behind the additional 0.3ms in inference time. Furthermore, the use of 11 convolution layers will undoubtedly consume additional time

compared to 6 layers. As ResNeXt50 has a denser architecture than ResNet50, its inference time is the maximum (33.2ms) compared to the rest of the used pretrained networks.

4.4.6 Qualitative Results

Qualitative results for LiCaNet are displayed in Figure 4.6. Five scenes are presented to visually compare LiCaNet (LIDAR only) predictions with LiCaNet (VGG16_6). Each scene is displayed in a column; the first row displays the ground truth, the second row shows LiCaNet (LIDAR only) predictions, and the last row depicts LiCaNet (VGG16_6) predictions. The ground truth is also included in the second and third rows for easier visual comparison. Arrows represent motion predictions. The presented examples demonstrate many prediction differences between the experiments; yet, only the most apparent ones are labeled with circles to simplify the comparison process for the reader. It is obvious that the overlap between LiCaNet (VGG16_6) predictions and the ground truth is higher compared to LiCaNet (LIDAR only) and the ground truth, indicating better accuracy attained by LiCaNet (VGG16_6). Indeed, these examples vividly illustrate that our proposed LiCaNet model has enhanced perception and motion prediction than LiCaNet (LIDAR only). To that end, in addition to quantitative analysis, it is now qualitatively proven that incorporating camera images in the fusion network enhances performance.

Furthermore, Figure 4.7 provides visual comparison on perception and motion prediction confined to the camera FOV (70°). Examining the overlap between the predictions and the ground truth, it is evident that the accuracy level is higher within the camera FOV for LiCaNet (VGG16_6), especially for small and distant objects. Thus, this comparison further validates the positive effect of adding semantic camera features to the fusion network.

4.5 Summary

This chapter presented a new method, LiCaNet, that fuses multi-modal features into a backbone network to perform accurate pixel-wise joint perception and motion prediction for autonomous driving. LIDAR and camera sensors are used to extract rich and complementary multi-modal features. The LIDAR data is represented in sequential BEV and RV forms. The predictions are attained in real-time, making LiCaNet suitable for real-world autonomous driving applications. The

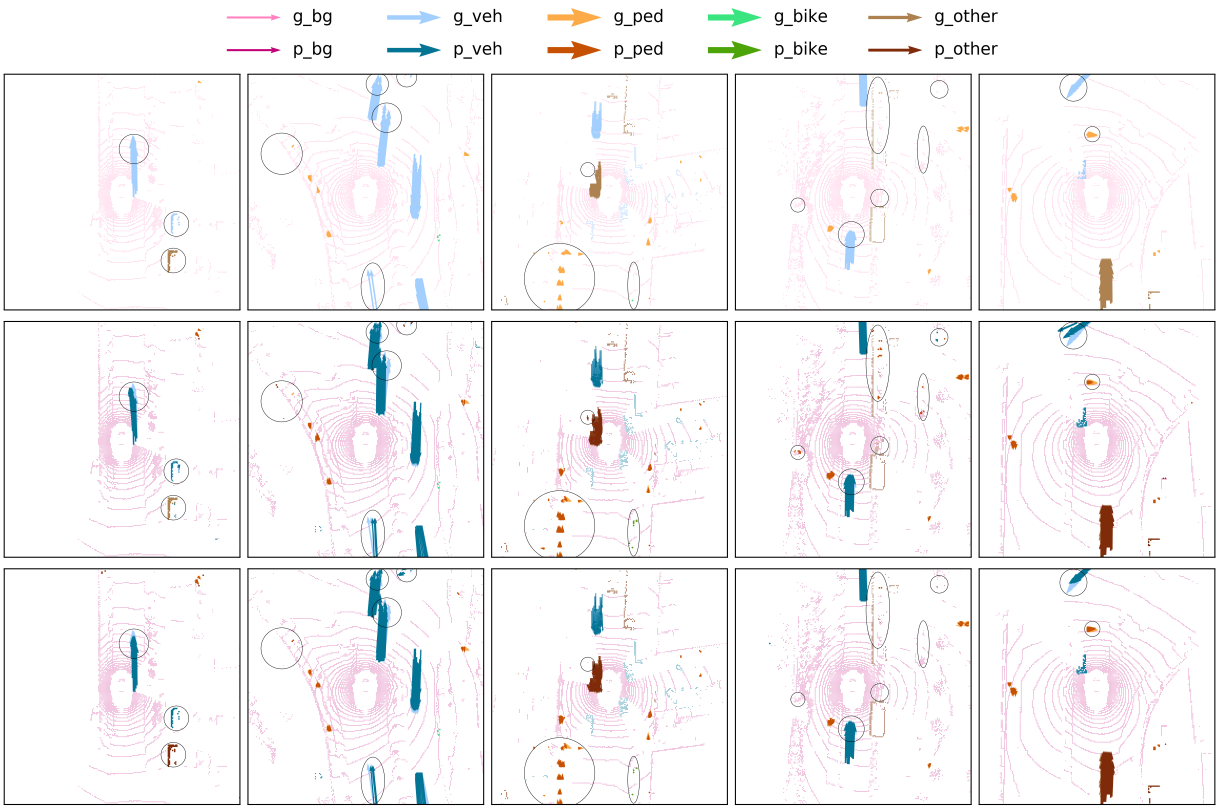


Figure 4.6 Qualitative comparison of perception and motion prediction. Top row: ground truth. Middle row: LiCaNet (LIDAR only). Bottom row: LiCaNet (VGG16_6). Ground truth is also present in the second and third row for easier visual comparison. Color codes are presented at the top of the figure. $g\_$ and $p\_$ denotes ground truth and prediction colors, respectively.

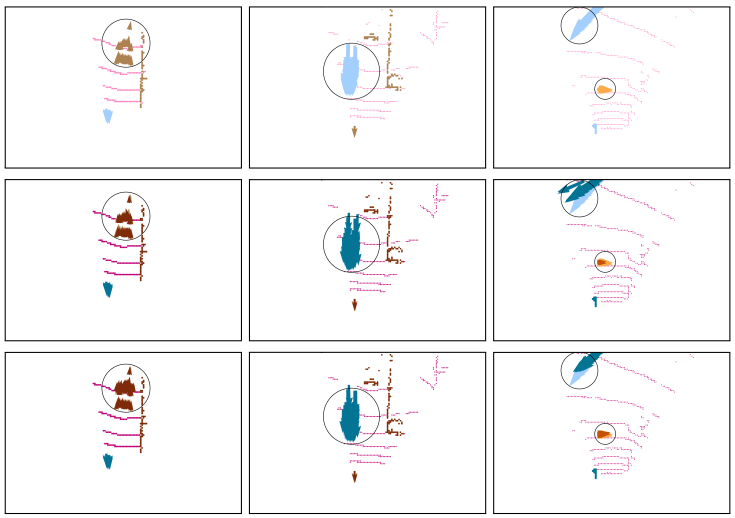


Figure 4.7 Examples of perception and motion prediction within camera range. Ground truths are presented in all rows. The second and third rows include the outcomes of LiCaNet (LIDAR only) and LiCaNet (VGG16_6), respectively.

experimental evaluation confirms that the involvement of camera information results in enhanced performance for joint perception and motion prediction. In addition, most accuracy improvement is registered within the camera field-of-view region, with the highest recorded for small and distant objects. Overall, LiCaNet outperforms its multi-view LIDAR-based predecessor fusion network and MotionNet.

CHAPTER 5
LICANEXT: INCORPORATING SEQUENTIAL RANGE RESIDUALS FOR ADDITIONAL ADVANCEMENT IN JOINT PERCEPTION AND MOTION PREDICTION

This chapter introduces *LiCaNext*, a buildup of LiCaNet, to capture additional accuracy advancement in joint perception and motion prediction while maintaining real-time requirements. In contrast to LiCaNet (Chapter 4), which fuses sequential bird's-eye view (BEV), range view (RV), and camera images, *LiCaNext* introduces sequential range residual images into the multi-modal fusion network to further improve performance, with minimal increase in inference time. Employing sequential range residual images has a substantial direct impact on motion prediction and positively influences perception. This chapter begins with a brief introduction to the advantages of range residual images and the benefits gained from fusing them with BEV, RV, and camera features. Second, a condensed review is given of works that addressed motion segmentation and works that targeted joint perception and motion prediction. Third, the proposed *LiCaNext* architecture is explained. Experimental evaluation is provided next, verifying the significant performance gain resulting from incorporating sequential range residuals, with monotonic progress for a larger number of exploited residuals. Last, a summary concludes this chapter.

In the following chapter (Chapter 6), multi-modal feature fusion is exploited to generate accurate perception and motion prediction using *LiCaNext*; then, these predictions are used to train a sequential latent maximum entropy reinforcement learning (MaxEnt RL) model. This learns a driving policy in urban environment that achieves the objectives of safety, efficiency, and comfort. Finally, Chapter 7 concludes this Ph.D. thesis.

5.1 Introduction

Recently, researchers invested an ample amount of time and momentum towards improving the safety of autonomous driving by enhancing the accuracy and reliability of its primary components [153, 152, 146, 34, 38, 62]. As mentioned in the previous chapters, the current main research

focuses on exploiting multi-modal fusion to develop superior perception and motion prediction components. The multi-modal fusion technique leverages data extracted from a diverse range of representations from sensors deployed on an autonomous vehicle to 1) better infer the current state of its surroundings and 2) accurately predict the dynamicity of this state in the near future. The utilization of multi-modal fusion in perception and motion prediction proved its competence in enhancing performance [153, 146, 38]. The prominence of multi-modal fusion stems from the fact that the generated features are rich and complete. These features constitute the complementary properties of all fused data representations and alleviate the inherent constraints of individual representations.

The work presented in this chapter focuses on developing a joint perception and motion prediction model for autonomous driving by combining data from LIDAR and camera sensors. The two most common representations of LIDAR data are BEV [146, 34] and RV [152, 38]. The previous chapter proposed LiCaNet [153], which tackled the same problem and produced challenging results. LiCaNet formulates its LIDAR data in BEV and RV representations. The BEV input is composed of a historical sequence of LIDAR data, while the RV and the camera input images represent only the current frame. This chapter proposes *LiCaNext*, the *next* version of LiCaNet, which expands on the multi-modal fusion network by incorporating sequential range residual images [93].

Range residuals are computed using the frame differencing technique [154], a pixel level comparison between the current and previous frames. Figure 5.1 illustrates an example of sequential range residual images. A residual image captures rich temporal information of objects, which is vital for identifying motion in a scene, leading to foreground-background differentiation. Object motion segmentation is identified as one of the fundamental requirements for real-world applications, including visual surveillance [155], traffic control [156], autonomous driving [146], and much more. An active research area in scene understanding for autonomous driving is moving object segmentation (MOS) [93, 157, 42, 158, 159]. MOS is a class-agnostic approach that detects and localizes motion in scenes. Furthermore, MOS can detect unseen objects (e.g., rare animals or construction vehicles) since it relies on motion cues rather than semantics. Typically, the ability to detect the dynamicity of the surrounding environment is pivotal for the safety of autonomous driving because it enables the prediction of objects' future states and path-planning. The terms

motion detection and motion segmentation are used interchangeably in this work.

A common challenge in motion detection is the rapid variation in illumination, such as the sudden appearance of clouds in the sky. In reality, under sharp changes of lightning conditions, static pixels are expressed by different intensities from the rest of the background pixels; thus, incorrectly classifying them as foreground. Fortunately, our computed range residuals are invariant to changes in illumination because range images only contain distance values to objects, and intensity values are excluded. Consequently, motion detection performance remains unaffected under adverse illumination situations if temporal information is extracted from range residual images. Accordingly, extracting rich temporal information from range images and integrating them into the multi-modal fusion network permits us to obtain an effective and efficient motion detection invariant to changes in intensities.

The RV input image of our previous multi-modal fusion network (LiCaNet) lacks temporal information. It merely consists of spatial information, including the range, intensity, and height of objects in the surrounding environment. After inserting sequential range residual images in RV form, our RV input images now generate features that embrace spatio-temporal information. Even though utilizing sequential range residuals is a modest adjustment to our multi-modal fusion network, the boost in perception and motion prediction accuracy that results from this simple addition is substantial. Many models exist in the literature that offers additional performance advancement, in these two critical components, compared to their predecessors; nonetheless, the extra computation introduced is relatively high compared to the additional accuracy attained. LiCaNet model, an enhancement to MotionNet [34], achieved an increase of 2.1% in perception and a mean error drop of 23.1mm in motion prediction for the fast category (pixels having speed $> 5m/s$); with a rise of 7.6ms in inference time. On the other hand, our proposed *LiCaNext* obtains an increase in inference time of only 1.6ms while achieving a perception gain of 1.6% and a mean error decrease of 73.7mm in motion prediction for the fast category. Thus, a simple modification applied to a model that leads to a significant progression in accuracy with a limited increase in computation time is considered a significant contribution.

Accordingly, alongside the temporal information fed into the BEV module of our multi-modal

fusion network, the range residuals also hold the dynamicity of the surroundings. This redundant temporal information strengthens the model's confidence in differentiating between foreground-background objects, accurately predicting motion, and enhancing perception. Furthermore, the increase in inference time incurred due to the addition of residual images is minimal. Figure 5.2 demonstrates the methodology of inserting sequential range residuals into our multi-modal fusion network.

Overall, the features generated by our proposed *LiCaNext* combines the 1) physical object dimensions and temporal information represented in BEV images, 2) occlusion information characterized in RV form, 3) motion cues embodied in both range residuals and RV forms, and 4) rich semantics of the surrounding environment signified in a camera image. Finally, these generated features are fed into MotionNet backbone network [34] to perform accurate pixel-wise joint perception and motion prediction in real-time.

5.2 Review

This section briefly reviews works that target motion object segmentation for autonomous driving. In addition, a concise overview of works that address perception and motion prediction is presented. Motion segmentation is defined as detecting motion at a pixel level, while motion prediction is forecasting the future motion of objects. In this work, we perform pixel-wise motion prediction.

5.2.1 Moving Object Segmentation

Early motion detection architectures relied on a geometric understanding of the scene [160]. Such architectures face difficulty adapting to challenging situations as their designs are complex enough and are created exclusively for specific challenging scenarios. Recent advancements in learning-based approaches lead to massive progress in motion detection [161, 162, 163, 93, 157, 42, 164, 165]. SMSnet [161] is a method that leverages a convolutional neural network (CNN) and depends on two sequential camera images to perform pixel-wise category labeling and motion detection. MODNet [162] is another CNN model that fuses motion and appearance cues to perform joint detection and motion segmentation. One camera sensor is used in [161, 162] to perform

motion segmentation on vehicles only. Unfortunately, it is unsafe to depend merely on a camera sensor for performing MOS because the semantics of the image degrades sharply under low-quality illumination conditions. Conversely, Dewan et al. [163] proposed a LIDAR-based method that depends on two sequential scans to perform a pointwise segmentation classifier distinguishing foreground objects from static background. The model in [163] makes use of up-convolutional networks to accomplish the desired task. Moreover, Chen et al. [93] proposed real-time class-agnostic motion segmentation using sequential LIDAR scans. The input to the CNN network in [93] is a combination of RV image, representing the current LIDAR scan, and range residuals computed from historical LIDAR frames.

Recent works are targeting the fusion of multi-modal data to attain more robust motion segmentation. RST-MODNet [157] is a CNN architecture that leverages the fusion of sequential camera and optical flow images to achieve real-time motion detection. FuseMODNet [42] is another real-time CNN architecture; however, it depends on combining LIDAR and camera images to capture motion information. Additionally, Mohamed et al. [164] developed a real-time CNN architecture, for instance-level class-agnostic motion segmentation with a camera and optical flow images as input. Unlike the earlier versions, [164] improves the diversity of moving objects by adding four additional classes instead of just vehicles. Lastly, BEV-MODNet [165] is another enhancement that investigated the idea of learning motion detections directly on the BEV space. In [165], a deep network is designed with a two-stream RGB and an optical flow fusion architecture.

5.2.2 Perception and Motion Prediction

Plethora of methodologies has been explored to enhance perception and motion prediction performance, where the majority used single input representation to address this task [62, 59, 63, 88, 35, 37, 60, 166, 147, 34]. Recently, applying multi-modal feature fusion has sparked a lot of interest from the research community in autonomous vehicles. To begin with, Fadadu et al. [38] define a unified architecture that incorporates two views of LIDAR data (BEV and RV), camera, and HD maps for advanced object detection and trajectory prediction. Another fusion method attempts to integrate the outcomes of MotionNet with BEV images to perform efficient and safe autonomous driving in an urban environment by training a reinforcement learning model [146].

Khalil et al. [152] put forward a multi-view LIDAR-based fusion network to enhance pixel-wise joint perception and motion prediction compared to MotionNet baseline. The two input LIDAR representations employed in [152] are BEV and RV images. Lastly, recently proposed LiCaNet [153] extends [152] with camera image fusion. LiCaNet records excellent performance for both perception and motion prediction compared to its predecessor.

In comparison to the multi-modal fusion networks mentioned above, this chapter proposes *LiCaNext*, a buildup on LiCaNet model where the multi-modal fusion network is expanded to involve range residuals in RV representation. In addition to the temporal information provided by the historical sequence of BEV images, employing motion cues encoded in sequential range residuals reinforces the richness and completeness of the generated features. Therefore, our multi-modal fusion network is fed with redundant temporal information in RV form, allowing the exploitation of spatio-temporal information from both BEV and RV representations.

5.3 Proposed Methodology

5.3.1 Input Representation

All input representations in *LiCaNext* are the same as LiCaNet, except for the newly incorporated range residuals.

a) Bird's-Eye View, Range View, and Camera: The BEV and RV in *LiCaNext* are formulated the same way as in LiCaNet (Chapter 4). The BEV dimensions are $256 \times 256 \times 13 \times 5$, while the RV dimensions are $1024 \times 32 \times 4$. Moreover, the used camera images have the same dimensions as in LiCaNet ($1600 \times 900 \times 3$).

b) Range Residuals: As aforementioned, the main idea of this work is to append sequential range residuals into the multi-modal fusion network to push further the performance of the joint perception and motion prediction model. The range residuals in RV form are computed by subtracting range images of both current and previous frames. Following [93], the first step towards

computing a range residual image is to extract the range images of the two frames separately. The second step is to compute the residual values by subtracting the two range images from each other at a pixel level. Only valid pixels in both range images are considered for computing the residual values of the corresponding pixels. The residual values of all other pixels are set to 0.

However, before computing the range images of the two frames, the previous frame must be synchronized to the current frame. The synchronization process is done by transforming the point cloud of the previous frame into the coordinate system of the current frame. This stage is necessary to counterfeit the ego-motion (motion of the autonomous vehicle). The final step is to normalize the residual images using (5.1).

$$res^0_{n,m} = \frac{\left|\Delta range^0_{n,m}\right|}{range^0_m} \quad \text{and} \quad \Delta range^0_{n,m} = range^0_m - \ range^n_m \tag{5.1}$$

where $\Delta range^0_{n,m}$ is the non-normalized residual image between the current range image ($range^0$) and the previous n^{th} transformed range image ($range^n$). Whereas, $range^0_m$ and $range^n_m$ represent the range value at cell m of the current and the n^{th} frames, respectively. Lastly, $res^0_{n,m}$ signifies the residual value at cell m between the 0^{th} and the n^{th} frames. The dimension of each residual image is the same as the individual channels of the RV image (1024×32). If multiple residual images are to be fused in the multi-modal fusion network, then they are stacked on top of each other. For example, if 4 residuals are fused, then the input representation of the sequential range residual images becomes $1024 \times 32 \times 4$.

Figure 5.1 provides examples of normalized sequential range residual images at different timestamps. The current and the previous frames are sampled using the same configuration adopted for formulating the BEV input. The range image of the current frame and its corresponding labels, both in RV form, are included in Figure 5.1 for reference purposes. The labels are color-coded to easily distinguish and locate the different objects in the range and residual images. The color black is assigned for background, blue for vehicles, red for pedestrians, green for bikes, and brown for all other objects available in the surrounding. The physical appearance of some objects in the label image does not reflect their actual shape due to the scale variance issue in RV images. For instance, some pedestrians are rendered by few dots, whereas, looking at the pedestrian located in

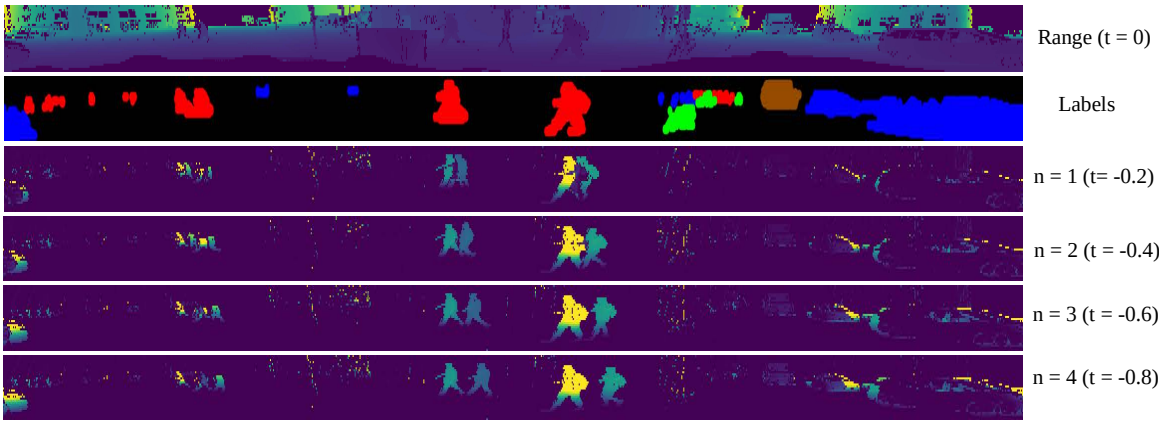


Figure 5.1 An illustrative example of normalized range residual images. The first two images portray the range of the current frame and the labels of all objects in that frame. A maximum of $\hat{N} = 4$ residual images is demonstrated, with n indicating the residual image engendered between the previous n^{th} transformed frame and the current frame.

the middle of the label image, it is evident from its appearance that it is a pedestrian. Typically, only a few LIDAR points represent distant objects. Thus, when transformed into RV form, their appearance is limited to a couple of pixels, so the appearance will not reflect the object's actual shape. The location of objects relative to the LIDAR sensor is visible in Figure 5.2 as it comprises the labels of the current frame in BEV form. Furthermore, Figure 5.2 also includes an RGB image displaying the front-side of the current scene within the camera field-of-view (FOV).

Interpreting the residual images in Figure 5.1, one can undoubtedly observe the dynamicity of some objects due to their high motion. In comparison, other objects have weak displacement representation because of their moderate motion. In contrast, the remaining objects have void motion as they are static. In particular, a vehicle could be parked on the side of the road, waiting for a traffic light, or even pedestrians could be standing still waiting to cross the road.

5.3.2 LiCaNext Architecture

The *LiCaNext* architecture consists of four modules: BEV, RV, residual, and camera. The architecture scheme is presented in Figure 5.2. The flag channel of the RV image is not depicted in the figure. The addition of the residual module into the LiCaNet architecture leads to *LiCaNext*. A total number of 5 sequential frames are used to represent *LiCaNext* input. The current frame is used to construct: one BEV image (13 channels), one RV image (4 channels), and a single

RGB camera image (3 channels). Furthermore, each of the 4 previous frames generates one BEV image (13 channels) and one residual image (1 channel). Therefore, 5 BEV images are stacked to form the BEV module input, 1 RV image represents the RV module input, $\hat{N} = 4$ residual images represent the residual module input. Lastly, one camera image is used as input to the *LiCaNext* camera module. The color-coding in the BEV images is embraced for illustration purposes only, and they are the same as the labels image in Figure 5.1.

The BEV, RV, and residual modules consist of two 3×3 convolution layers. In contrast, the camera module involves a lightweight pretrained network, followed by projecting and warping the features into RV form, and lastly, two 3×3 convolution layers. In order to project and warp camera features onto the RV form to be concatenated with the RV images, the mapping between the LIDAR points and the camera image needs to be computed first. However, due to the different operating frequencies of LIDAR and camera sensors, coordinate transformation must be applied to compute the mapping. The first step is to transform the LIDAR points into the vehicle's coordinate system at LIDAR capture time. Secondly, transform the points that exist in the vehicle's coordinate from LIDAR capture time to camera capture time. Next, transform the resulting points from the vehicle's coordinate system at camera capture time into the camera's coordinate system. Lastly, apply the camera's intrinsic calibration matrix on the transformed points. This transformation process compensates for the time shift between the two sensors and results in the LIDAR points being mapped correctly on the camera image. Suppose the features extracted from the pretrained network have different dimensions than the original camera image. In that case, the mapping needs to be updated using a scale factor calculated between the dimensions of the extracted features and the original camera image. This mapping allows us to warp features from the camera image into RV representation.

The generated features from the RV, residual, and camera modules are then concatenated and sent into a U-Net. The same U-Net used in LiCaNet is adopted for *LiCaNext*. Briefly, the U-Net consists of two scaling layers with a horizontal scaling factor of 2 on each layer. The vertical scale is kept constant due to its small representation compared to the width of the RV image. At each U-Net layer, residual blocks with skip connections are used. The subsequent step is to project the resulting features from the U-Net onto the BEV form to be concatenated with the BEV features

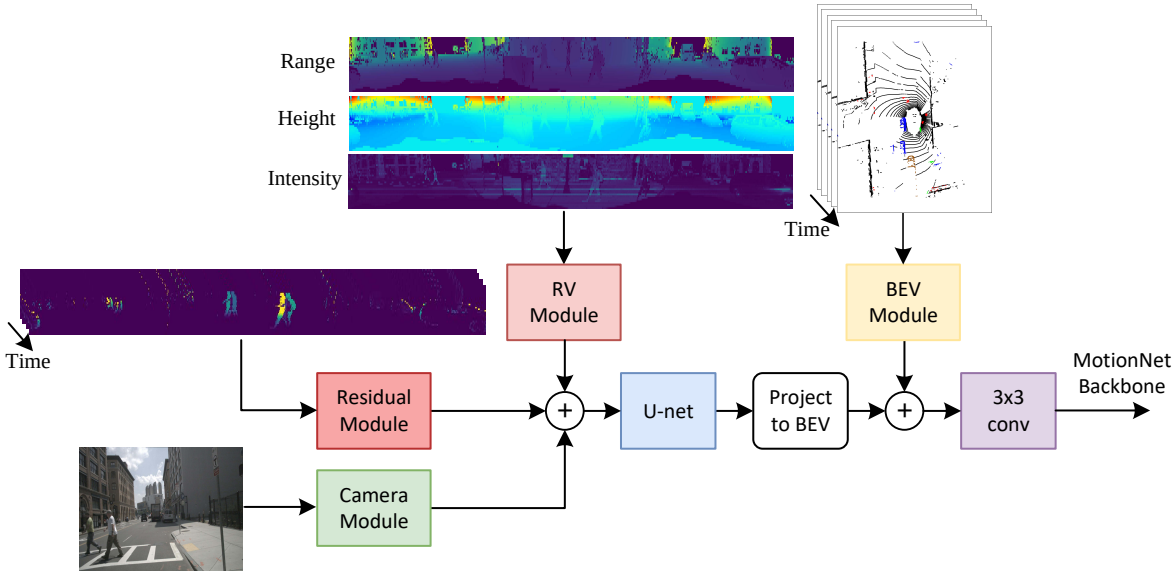


Figure 5.2 *LiCaNext* architecture.

encoded by the BEV module. Projection from one form to another is achieved using the painting approach. The projection from RV to BEV representation is illustrated in Algorithm 1 of Section 3.3.3. Additionally, the same approach is used to project from camera to RV representation. Basically, for each raw LIDAR point mapped to a BEV cell, its corresponding RV feature is projected into the same BEV cell position. If more than one RV feature ends up in the same cell, then the average is computed. Empty cells are filled with a value of -1. Further details on projection is found in Section 4.3.3.

The final step in *LiCaNext* multi-modal fusion network is to inject the concatenated features, in BEV form, into a single 3×3 convolution layer. At this stage, the produced features are rich and comprehensive. They consist of spatio-temporal information sourced from two representations (BEV and RV), physical object dimensions encoded in the input BEV images, occlusion information provided from RV images, and rich semantics signified in a camera image. When these features are inserted into MotionNet backbone network, they yield accurate pixel-wise joint perception and motion prediction in real-time.

5.4 Experiments and Results

5.4.1 Dataset

The experiments in this chapter are conducted on the nuScenes dataset, which is the same dataset used in the previous two chapters. A detailed description of the nuScenes dataset can be found in Sections 3.4.1 and 4.4.1.

5.4.2 Training Setup

Following the same training setup as LiCaNet, the learning rate is initialized at 1.6×10^{-3} and terminated at 0.8×10^{-3}, with a decay factor of 0.5 every 10 epochs. The time span between consecutive LIDAR sweeps used to construct the historical BEVs and the sequential residual range images is 0.2s. The current sweep is sampled at 2Hz for training and 1Hz for testing. The batch size used to train *LiCaNext* is 4.

5.4.3 Evaluation Metrics

Similar to the previous two chapters, motion prediction is evaluated using the mean and median displacement errors for pixels based on three-speed groups. The speed groups are static, slow, and fast. Pixels with speed $\leq 5m/s$ are assigned to the slow speed group, while pixels with predicted motion $> 5m/s$ are assigned to the fast group. If pixels are predicted to have 0 motion, then they are assigned to the static group. Due to the large proportion of staticity in a scene, distinguishing static from dynamic pixels becomes essential to avoid biased displacement errors. The displacement error is measured using L_2 distances between the predicted and the ground-truth displacements. Additionally for perception, the metrics used are classification accuracy per category, mean classification accuracy (MCA), and overall pixel accuracy (OA).

5.4.4 Experimental Setup

Table 5.1 comprises all experiments needed to verify the performance enhancement attained by *LiCaNext* in both perception and motion prediction. The first experiment records the performance

of the original MotionNet model, which acts as the primary baseline. LiCaNet [153] (Chapter 4) proved that engaging RV and camera images into the fusion process outperform the baseline, which depends merely on BEV images as input. Our proposed *LiCaNext* expands on LiCaNet by incorporating residual images pushing the performance even further. Two of the most prominent LiCaNet experiments are included in Table 5.1 to observe this performance advancement. The first LiCaNet experiment uses only a LIDAR sensor in the multi-modal fusion process, i.e., the fusion of BEV and RV images. The second LiCaNet experiment embraces LIDAR and camera sensors, i.e., the fusion of a camera image on top of BEV and RV images. The use of VGG16 in LiCaNet as the lightweight pretrained network for extracting high-level camera features achieved the best overall performance among the other evaluated pretrained networks. Accordingly, for *LiCaNext* experiments, 6 layers of VGG16 are adopted to represent the lightweight pretrained network.

Before evaluating the performance of the entire *LiCaNext* fusion model, the effect of fusing different numbers of range residual images on a multi-modal fusion network that depends merely on a LIDAR sensor is first examined. Therefore, the first series of *LiCaNext* (LIDAR only) experiments will omit the camera module and fuse only BEV, RV, and residual images. The outcome of this evaluation will be compared to LiCaNet (LIDAR only) experiment. Consequently, the number of sequential range residuals that realize the best performance is adopted to evaluate the entire *LiCaNext* model, including the camera module. In all experiments, the RV and range residual images are defined to have width and length dimensions of 1024×32, respectively.

5.4.5 Quantitative Results

Table 5.1 reveals that *LiCaNext* achieves outstanding joint perception and motion prediction results compared to its predecessor LiCaNet. We begin our evaluation by comparing *LiCaNext* (LIDAR only, *res* = 1) experiment to LiCaNet (LIDAR only). In this comparison, the effect of fusing one range residual image with BEV and RV images is investigated. It is worth noting that when the residual module is removed from *LiCaNext*, i.e., no range residual images are fused, its architecture becomes the same as LiCaNet. *LiCaNext* (LIDAR only, *res* = 1) experiment obtains a maximum gain of 1.2% and 0.2% in MCA and OA, respectively. Moreover, a drop in displacement error is recorded for all speed groups, indicating better motion prediction. This evinces that the fu-

Table 5.1 Perception and motion prediction comparison between our proposed *LiCaNext* and LiCaNet models. Performance of the original MotionNet model is also included. Pixels are assigned to static, slow, and fast speed groups if their predicted motion is $0m/s$, $(0m/s, 5m/s]$, and $(5m/s, 20m/s]$, respectively.

Method	Static		Slow		Fast		Classification Accuracy (%)							Time
	Mean	Median	Mean	Median	Mean	Median	Bg	Vehicle	Ped.	Bike	Others	MCA	OA	(ms)
MotionNet [34]	0.0236	0	0.2534	**0.0959**	1.0778	0.7346	97.5	91.2	74.9	22.1	65.9	70.3	96.3	**20.5**
LiCaNet (LIDAR only) [153]	0.0230	0	0.2531	0.0964	1.0547	0.7305	97.6	92.0	80.6	22.6	69.2	72.4	96.6	28.1
LiCaNext (LIDAR only, *res* = 1)	0.0229	0	0.2526	0.0960	1.0325	0.7256	97.9	92.6	82.6	24.1	71.0	73.6	96.8	29.2
LiCaNext (LIDAR only, *res* = 2)	0.0226	0	0.2499	0.0960	1.0231	0.7142	97.9	92.9	82.7	23.9	71.4	73.8	96.8	29.4
LiCaNext (LIDAR only, *res* = 3)	0.0223	0	0.2484	0.0960	1.0075	0.7073	97.9	93.1	82.5	23.7	72.7	74.0	96.8	29.5
LiCaNext (LIDAR only, *res* = 4)	**0.0222**	0	**0.2479**	0.0960	**0.9810**	**0.6866**	98.0	93.0	82.8	25.7	70.3	**74.0**	**96.9**	29.7
LiCaNet (VGG16_6) [153]	0.0224	0	0.2527	0.0969	1.0456	0.7300	97.8	92.4	84.0	23.2	71.9	73.9	96.9	31.0
LiCaNext (VGG16_6, *res* = 4)	**0.0221**	0	**0.2425**	0.0961	**0.9801**	**0.6842**	98.1	93.1	83.9	24.6	72.0	**74.3**	**96.9**	31.9

sion of a single residual image with BEV and RV images significantly advances motion prediction and offers a substantial rise in perception.

Next, the effect of increasing the number of fused residual images on performance is examined. Table 5.1 unveils that an increase in the number of fused residual images establishes a monotonic rise in perception accuracy and a constant reduction in displacement error. The optimal performance is registered for *LiCaNext* (LIDAR only, *res* = 4) experiment, where 4 residual images are exploited in the multi-modal fusion process. A peak gain of 1.6% and 0.3% is established in MCA and OA, respectively. Furthermore, a 73.7mm and 43.9mm decrease in the mean and median errors for the fast-speed group is recorded, and a drop of 5.2mm and 0.4mm for the slow-speed. The mean error for the static group procured a reduction of 0.8mm with no median error. On the other hand, the *LiCaNext* (LIDAR only, *res* = 1) achieves a mean and median drop of 22.2mm and 4.9mm for the fast-speed group, 0.5mm and 0.4mm for slow-speed, and 0.1mm and 0m for the static group. This confirms that increasing the number of sequential residual images in the fusion process lowers the displacement error significantly.

Moreover, during the training stage, the model learns associations between the motion patterns embedded in the residual images and their corresponding objects presented in the other input representations. These learned associations positively influence perception. For instance, vehicles typically have higher speeds and are characterized by more pixels than pedestrians and bikes. So when strong-motion cues sourced from multiple neighboring pixels are fed into the model, this in-

creases the model's confidence in categorizing those pixels as a vehicle. Thus, fusing motion cues in the form of range residuals enhance perception. Lastly, comparing the performance of *LiCaNext* (LIDAR only, *res* = 4) with the original MotionNet results in a significant gain in motion prediction across all speed groups, with a rise of 3.7% and 0.6% in MCA and OA, respectively. Even though the median error in the slow group is not the lowest for *LiCaNext* experiments; however, the obtained error is lower than what LiCaNet achieved.

The reason behind incorporating a maximum of 4 range residual images in *LiCaNext* is to match the number of fused BEV images generated from previous frames. After determining that fusing 4 residuals with BEV and RV images yields the best accuracy in joint perception and motion prediction, the subsequent step is to evaluate the performance of the entire *LiCaNext* model. Usually, the inclusion of camera images in the fusion process boosts performance because of the rich semantic information provided by the RGB images. The performance gain as a result of including a camera sensor is realized in LiCaNet [153], where LiCaNet (VGG16_6) experiment outperformed LiCaNet (LIDAR only). Therefore, integrating a camera image into the fusion of BEV, RV, and residual images should further push performance. According to Table 5.1, *LiCaNext* (VGG16_6, *res* = 4) achieves a perception enhancement of 0.3% in MCA, and the accuracy of OA is maintained, compared to *LiCaNext* (LIDAR only, *res* = 4). A noticeable drop in displacement error is fulfilled in most speed groups, except for the median error of the slow group, which increased by 0.1mm only. This confirms that adding a camera image onto the fusion of BEV, RV, and residual images improves perception and motion prediction even further. Comparing *LiCaNext* (VGG16_6, *res* = 4) to a model that fuses BEV, RV, and camera images (i.e., LiCaNet (VGG16_6)), a 0.4% rise is registered in MCA, while OA accuracy is maintained. In addition, a greater drop in displacement error is recorded in all speed groups. This reveals that incorporating residual images onto a multi-modal fusion network involving a camera module positively affects performance. Ultimately, the motion prediction advancement that *LiCaNext* accomplished compared to MotionNet is even better than what its LIDAR-only version procured. *LiCaNext* obtained an outstanding enhancement of 4.0% in MCA and 0.6% in OA compared to MotionNet.

Table 5.2 presents the effect of exploiting sequential range residuals on small and distant objects. Generally, small objects (e.g., pedestrians and bikes) have lower perception accuracy than

Table 5.2 Evaluating the perception accuracies based on three distance ranges: Short (S) - (0m, 10m], medium (M) - (10m, 20m] and far (F) - (20m, 30m]. The results of the last two experiments are limited to the camera 70° FOV.

Method	Classification Accuracy (%)														
	Background			Vehicle			Pedestrian			Bike			Others		
	S	M	F	S	M	F	S	M	F	S	M	F	S	M	F
MotionNet [34]	98.3	97.3	95.8	94.5	91.0	81.8	77.4	74.5	73.3	29.1	19.5	17.0	76.1	62.6	52.5
LiCaNet (LIDAR only) [153]	98.4	97.5	96.1	94.3	92.1	84.8	83.8	78.7	78.1	30.9	18.0	**19.9**	78.5	65.7	58.5
LiCaNext (LIDAR only, *res* = 1)	98.5	97.7	96.6	94.7	93.0	85.1	86.5	**82.0**	80.0	31.4	21.4	18.3	79.7	69.1	60.4
LiCaNext (LIDAR only, *res* = 2)	98.5	97.6	95.6	95.0	93.1	85.8	86.6	81.6	80.2	31.2	21.1	18.4	80.2	70.3	60.6
LiCaNext (LIDAR only, *res* = 3)	98.5	97.6	96.5	**95.4**	**93.3**	86.1	**86.6**	81.7	80.0	30.9	20.5	19.2	**81.1**	**70.6**	**62.7**
LiCaNext (LIDAR only, *res* = 4)	**98.6**	**97.7**	**96.6**	95.1	93.2	**86.2**	86.5	81.6	**81.0**	**36.5**	**21.2**	18.6	78.6	68.3	60.3
LiCaNet (VGG16_6) [153]	98.7	97.2	95.3	95.2	94.6	87.5	90.3	86.7	84.0	35.2	23.6	35.0	85.5	79.5	67.7
LiCaNext (VGG16_6, *res* = 4)	**98.8**	**97.4**	**95.6**	**95.4**	**94.9**	**87.9**	**90.7**	**87.1**	**84.6**	**35.9**	**24.1**	**35.8**	**85.5**	**80.1**	**68.1**

larger objects (e.g., vehicles) as they are represented by much fewer pixels. Similarly, the accuracy drops naturally with increasing distance from the sensor. This is because the sensor's performance degrades progressively at capturing fine information with a farther distance. However, Table 5.2 unveils that *LiCaNext*, compared to LiCaNet, improves accuracy even further for small and distant objects. The perception accuracies presented are measured according to three distance range groups: short (S), medium (M), and far (F). The short-range is defined to include all pixel detections within the (0m, 10m] range. While medium- and far-ranges are defined for pixels in the (10m, 20m] and (20m, 30m] ranges, respectively. Furthermore, the last two experiments involve a front camera sensor. To evaluate the effectiveness of the multi-modal feature fusion, which includes the camera features, the perception measurements of the last two experiments are restricted to the camera 70° FOV. This is because camera features are incorporated in that area only. Most of the improvements due to the camera features would be recognized in that same region. However, the perception accuracy of the other experiments is measured on the entire LIDAR 360° FOV.

To begin with, *LiCaNext* (LIDAR only, *res* = 4) compared to LiCaNet (LIDAR only) experiment achieves a 0.8% rise in accuracy for the vehicles category in the short-range group. On the other hand, pedestrians and bikes obtain a higher accuracy gain of 2.7% and 5.6% for the same speed group, respectively. This shows that the jump in detection rate for smaller objects is higher than bigger ones, proving that *LiCaNext* attains better accuracy for smaller objects. Next, we demonstrate how the fusion of residual images caused better improvements in the perception

of distant objects. The accuracy gain for vehicles and pedestrians in the far-range is 1.4% and 2.9%, respectively. The bikes category is the only exception where the accuracy dropped by 1.3%; however, the overall accuracy of bikes is still superior to LiCaNet (LIDAR only) by 3.1%. Hence, not only is the detection gain higher for most object classes in the far-range in relation to short-range, but also smaller distant objects (pedestrians) obtained a greater gain than larger ones. This observation confirms that residual images assist in intensifying the detection rate for both small and distant objects.

Furthermore, comparing the drop in accuracy between the far- and short-range in both *LiCaNext* (LIDAR only, *res* = 4) and LiCaNet (LIDAR only), it is clear that the drop in accuracy is lower in most categories. For the background, vehicles, pedestrians, and others, *LiCaNext* (LIDAR only, *res* = 4) achieved a drop of 2%, 8.9%, 5.5%, and 18.3%; whereas, LiCaNet (LIDAR only) achieved a drop of 2.3%, 9.5%, 5.7%, and 20.0%, respectively. This shows that the fusion of residual images reduces the loss resulting from the natural degradation of sensors' performance with farther distances. A final observation is that experiments that fuse range residuals secured more significant gains compared to MotionNet than what LiCaNet accomplished, especially for small and distant objects.

After demonstrating, in Table 5.1, that exploiting residual images in a multi-fusion network involving a camera module enhances performance even further, the effect of this experiment on small and distant objects is further investigated. The following comparisons will only be made between the last two experiments of Table 5.2, as their measurements are restricted to the camera 70° FOV. The last two experiments show that within the camera FOV, the perception accuracy of *LiCaNext* (VGG16_6, *res* = 4) outperforms LiCaNet (VGG16_6). The accuracy improvement attained in the short-range for vehicles is 0.2%, while 0.4% and 0.7% are acquired for pedestrians and bikes. The accuracy improvement for vehicles in the far-range is 0.4%; whereas, for pedestrians and bikes, it is 0.6% and 0.8%, respectively. This shows that the inclusion of residual images within the fusion process of BEV, RV, and camera images results in improved perception for all object categories, and even stronger accuracy is secured for small and distant objects. Moreover, the accuracy drop between the far- and short-range groups is lower in *LiCaNext* (VGG16_6, *res* = 4) compared to LiCaNet (VGG16_6). This indicates that the fusion of residual images onto BEV, RV, and cam-

era images decreases even further the natural effect of performance degradation within the camera FOV at farther distances.

Expanding the LiCaNet (LIDAR only) network to include residual images has incurred an additional inference time of 1.6ms for *LiCaNext* experiments (without the camera module). The involvement of the camera module resulted in an increase of 2.2ms compared to *LiCaNext* (LIDAR only, *res* = 4). Furthermore, the fusion of BEV, RV, residual, and camera images resulted in 0.9ms increase compared to LiCaNet (VGG16_6). The total inference time of the entire *LiCaNext* model is 31.9ms, which is less than the real-time requirement. Overall, the provided results confirm that incorporating residual images has a significant effect on improving performance with a minimal increase in inference time. Thus, *LiCaNext* can be used to perform accurate joint perception and motion prediction for autonomous driving.

5.4.6 Qualitative Results

In this section, qualitative examples are provided to illustrate the enhancements procured by *LiCaNext*. Figure 5.3 compares the outcomes of *LiCaNext* (LIDAR only, *res* = 4) and LiCaNet (LIDAR only) using five different scenes. The most noticeable gains are labeled with circles. Even though there exist other modest improvements that *LiCaNext* achieves, particularly with motion; nevertheless, they are not labeled as they are difficult to see. It is evident from the examples provided that *LiCaNext* results in more accurate perception and motion prediction. For instance, the motion predictions in the first example are more accurate in *LiCaNext* compared to LiCaNet, as the overlap between the motion predictions and the ground truth is higher in *LiCaNext*. The second example clearly shows how *LiCaNext* outperforms LiCaNet in terms of perception. The top right circle shows that LiCaNet mistakenly predicted several pixels as pedestrians, but *LiCaNext* correctly detected those pixels as background. Additionally, the middle right circle in the second example shows that both models mistakenly detected pedestrians even though there were none. The difference here is that *LiCaNext* predicted zero motion for those falsely detected pedestrians, whereas LiCaNet detected motion. Furthermore, the top right circle in the last example shows that *LiCaNext* predicted the vehicle's motion more accurately compared to LiCaNet. In that same example, LiCaNet wrongly predicted several pixels as pedestrians, whereas *LiCaNext* correctly

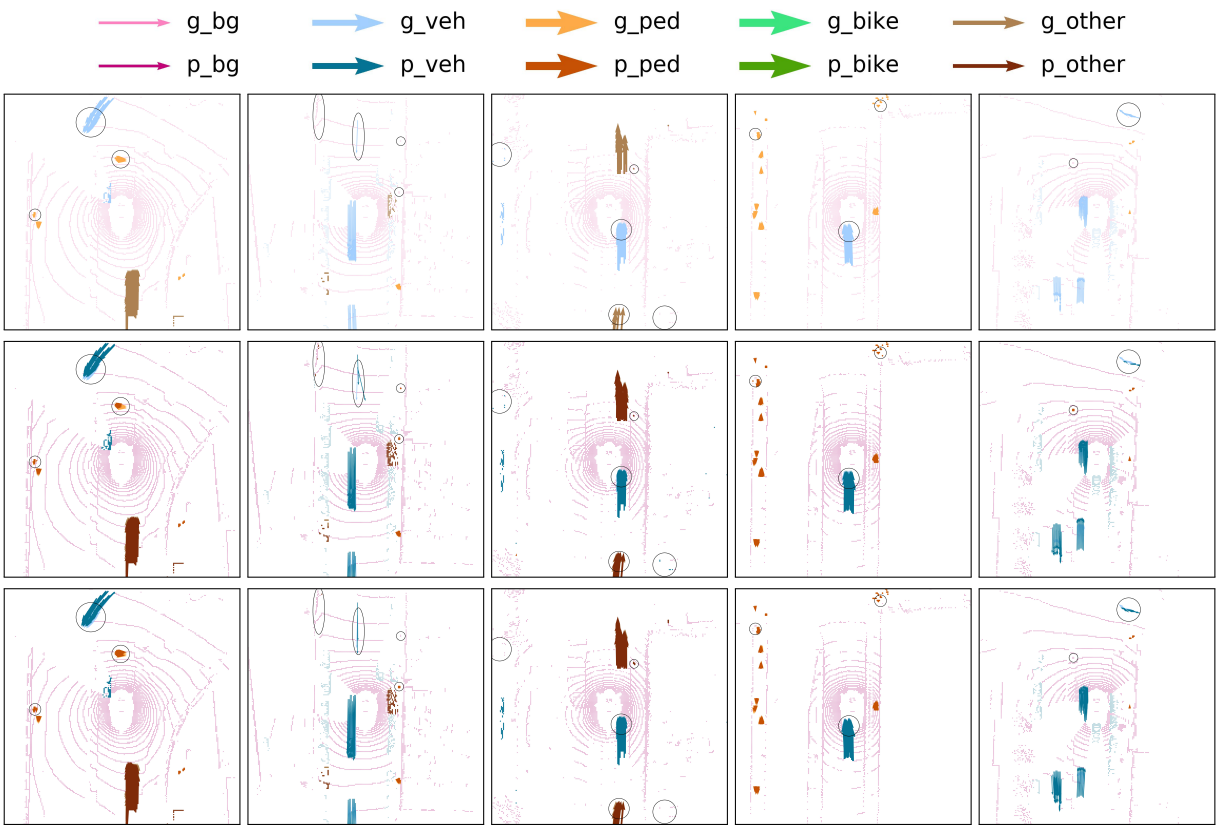


Figure 5.3 Comparison of perception and motion prediction using qualitative examples between *LiCaNext* (LIDAR only, *res* = 4) and LiCaNet (LIDAR only) in the full 360° range. The first row consists of only the ground truth. The second row displays the LiCaNet (LIDAR only) predictions, while the last row exhibits our *LiCaNext* (LIDAR only, *res* = 4) predictions. The ground truth is also present in the last two rows for easier visual comparison. The color codes used are attached at the top of the figure. *g_* denotes ground truth, while *p_* symbolizes prediction colors. *g_bg* and *p_bg*, for instance, represent the ground truth and predictions of the background pixels in the image, respectively.

predicted those as background.

To analyze the effect of fusing residual images in a multi-modal fusion network that involves a camera module, three qualitative examples are provided in Figure 5.4 to compare between *LiCaNext* (VGG16_6, *res* = 4) and LiCaNet (VGG16_6). The predictions displayed are within the vehicle's front view (70° FOV), as the camera features are only incorporated in that region, and most improvements will exist in that same region. Thus, restricting the view to the 70° FOV allows for easy visualization of the achieved enhancements due to adding residual images onto a multi-modal fusion network involving camera features. It is apparent from all three examples that

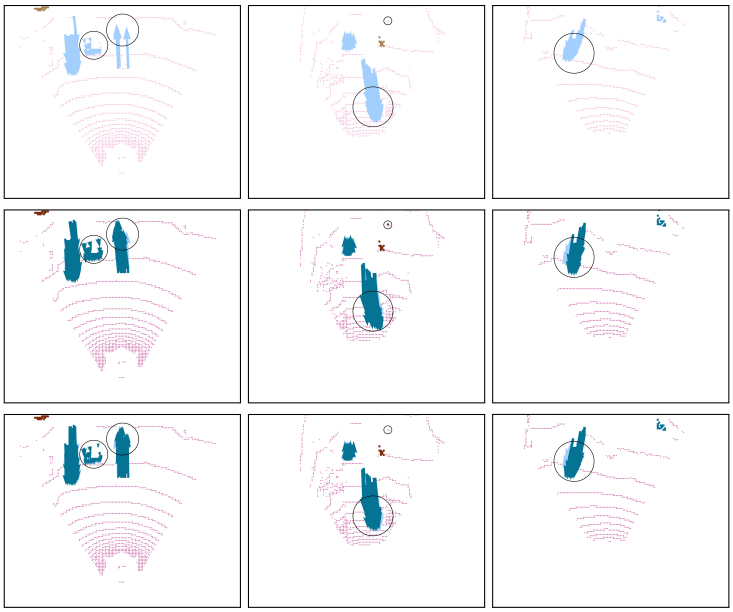


Figure 5.4 Qualitative comparison between the outcomes of *LiCaNext* (VGG16_6, *res* = 4) and LiCaNet (VGG16_6) within the camera 70° FOV. Rows indicate the ground truth, LiCaNet (VGG16_6), and *LiCaNext* (VGG16_6, *res* = 4) outcomes, respectively. The ground truth is also included in the last two rows.

the motion predictions attained by *LiCaNext* are closer to the ground truth compared to LiCaNet. The second example demonstrates how the perception of *LiCaNext* is better than LiCaNet. In that example, LiCaNet mistakenly detected a pixel as *others*; however, *LiCaNext* correctly detected that pixel as background. Moreover, in the third example, the overlap of *LiCaNext* motion predictions and the ground truth is greater than the overlap between the motion predictions obtained by LiCaNet and the ground truth. From the quantitative and qualitative results reported in this chapter, a safe conclusion can be drawn that our proposed *LiCaNext* pushes the accuracy boundaries even further than LiCaNet.

5.5 Summary

In this chapter, an accurate and real-time model, named *LiCaNext*, is proposed to perform pixel-wise joint perception and motion prediction. *LiCaNext* incorporates sequential range residual images in its multi-modal fusion process to significantly exceed the performance of its predecessor LiCaNet. *LiCaNext* performs exceptionally well on small and distant objects, achieving even better predictions than its previous version. Conducted experiments were evaluated on the challenging

nuScenes dataset, confirming the excellent joint perception and motion prediction results.

CHAPTER 6

EXPLOITING MULTI-MODAL FUSION FOR URBAN AUTONOMOUS DRIVING USING LATENT DEEP REINFORCEMENT LEARNING

The primary cause of road accidents is attributed to human driving error. Autonomous driving is a potential solution to eliminate human driving interaction, improving traffic safety and efficiency. In the past few years, researchers verified the great capability of deep reinforcement learning (DRL) in learning complex tasks, where various DRL-based approaches for autonomous driving have been investigated. However, exploiting multi-modal fusion to generate perception and motion prediction and then leveraging these predictions to train a latent DRL has not been targeted yet. To that end, this chapter proposes enhancing urban autonomous driving using *LiCaNext* with latent DRL. A single LIDAR sensor is used to extract bird's-eye view (BEV), range view (RV), and residual input images. These images are passed into *LiCaNext* [167] (Chapter 5), a real-time multi-modal fusion network, to generate accurate joint perception and motion prediction. Next, predictions are fed with another simple BEV image into the latent DRL to learn a complex end-to-end driving policy ensuring safety, efficiency, and comfort. A sequential latent model is deployed to learn a more compact representation from inputs, leading to improved sampling efficiency for reinforcement learning.

This chapter begins with an introduction to the advantages of DRL algorithms and the effect of exploiting multi-modal fusion on performance for urban autonomous driving using the DRL algorithm. Second, this chapter includes a concise review of works that applied DRL algorithms for learning a driving policy. Later, the proposed approach, implementation details, simulation environment, and experimental evaluation are explained in great detail. Lastly, a summary concludes this chapter. The next chapter (Chapter 7) concludes this Ph.D. thesis.

6.1 Introduction

As discussed in Section 1.2, human error is the main reason behind the high fatality rates on roads. A study conducted in [20] demonstrated the direct relationship between road accidents and

human driving. To overcome these catastrophic accidents, incompetent human driving decisions need to be eradicated. This can be achieved by introducing autonomous driving, which eliminates human driving and can also improve traffic safety and efficiency. Despite the intense efforts made by researchers over the last decade to develop urban autonomous driving, considerable challenges persist[33]. This is primarily because of the complexity of the driving environment and its diverse requirements: to get to the destination as fast as possible while avoiding collisions, respecting traffic rules, and ensuring passenger comfort.

Learning-based approaches have seen great success in complex high-dimensional applications that require extensive decision-making. Such environments include video games [168], healthcare [169], computer vision [170], transportation [33], and robotics [171]. On the other hand, traditional non-learning approaches tend to fail in such environments because they are biased towards human heuristics, making them hard to scale. For that reason, non-learning approaches are limited to low-dimensional environments. The two most effective learning-based paradigms adopted for autonomous driving are imitation learning (IL) [105] and reinforcement learning (RL) [33, 14]. Basically, IL is a supervised learning approach that learns a mapping between observations and control commands from expert driving demonstrations. The main drawbacks of IL are 1) it requires a massive amount of expert demonstrations to learn from; 2) is biased towards the expert driving skills and so cannot exceed human-level performance; 3) dataset does not cover dangerous situations. Conversely, an RL agent discovers by trial-and-error how to behave in an environment to maximize cumulative rewards. Therefore, in RL, the learning agent can learn experiences not covered in an IL dataset and achieve super-human performance.

With the striking achievements of deep learning, DRL has emerged from combining RL and deep learning techniques. DRL has witnessed exciting developments and registered remarkable breakthroughs in various domains in the past few years, including autonomous driving [22, 23, 24, 146]. Different types of DRL algorithms exist and are mainly categorized as value-based [25, 26, 27], policy-based [28], and actor-critic-based [29, 30, 31, 32] methods. This chapter adopts sequential latent maximum entropy (MaxEnt) RL [32] algorithm to learn a complex end-to-end driving policy in urban driving. Unlike other DRL algorithms, sequential latent MaxEnt RL performs representation and task learning separately. The high-dimensional observations are first

encoded into a more compact latent space using a learned sequential latent environmental model. Then MaxEnt RL is trained on this concise latent space, accelerating RL training and enhancing sampling efficiency.

This chapter proposes exploiting multi-modal feature fusion to generate accurate joint perception and motion prediction. These predictions with another simple BEV image are encoded using the learned sequential latent environmental model into a compact low-dimensional latent space. Next, MaxEnt RL is trained on this latent space to learn a driving policy that ensures safety, efficiency, and comfort in an urban environment [172]. The perception and motion prediction are generated using *LiCaNext* [167], a novel real-time multi-modal feature fusion network that generates accurate pixel-wise joint perception and motion prediction. *LiCaNext* secured substantial accuracy gain over state-of-the-art networks and has proven its success in detecting small and distant objects. Incorporating *LiCaNext* predictions as part of the input into the sequential latent MaxEnt RL adds valuable information to the learning process leading to an enhanced driving policy.

Typically, an urban environment is highly dynamic. Supplying an agent with information about the nearby objects and their future trajectories simplifies and enhances the agent's perception and motion prediction pipelines due to the relevancy of the inserted data. Additionally, the performance of other pipelines, such as decision and control, will automatically advance because all pipelines are interconnected and learned simultaneously in end-to-end training. Furthermore, an urban environment is crowded with small objects like pedestrians and bikes, which are harder to detect due to their physical structures. Providing the agent with such information deters it from dedicating substantial training time to detect and forecast the motion of small objects. Therefore, feeding our sequential latent MaxEnt RL with accurate perception and motion prediction enhances the driving policy's effectiveness.

The proposed approach relies merely on one LIDAR sensor to extract BEV, RV, and residual representations, forming *LiCaNext* input. Even though, the original *LiCaNext* network involves the fusion of camera images, in this work the camera module of *LiCaNext* is not used due to our limited computational hardware. The *LiCaNext* predictions with another simple BEV image construct the input to the sequential latent MaxEnt RL algorithm. Figure 6.1 shows the overall architecture of our

proposed approach. CARLA driving simulator [127] is used to conduct our experiments. Results reveal that our proposed methodology outperforms state-of-the-art methods and demonstrates its effectiveness against different environments and various weather conditions. Our learned policy navigates successfully in an urban environment while ensuring safety, efficiency, and comfort. This work is the first to investigate the exploitation of multi-modal fusion to procure accurate pixel-wise perception and motion prediction and then leverage these predictions to train a sequential latent MaxEnt RL for learning a complex driving policy.

6.2 Review

6.2.1 Deep Reinforcement Learning

The prevalence of accidents caused by inefficient human driving decisions spurred the interest of researchers to develop autonomous vehicles. As mentioned earlier, two families of learning-based approaches exist for learning autonomous driving policies: imitation learning (IL) and reinforcement learning (RL). Even though plenty of works have been conducted on IL, which learns driving policies by mimicking human driving behavior [105, 106, 107, 108, 109, 110]; however, most current works leverage RL. In particular, deep reinforcement learning (DRL) powered by deep learning has bloomed after the astounding advances of deep learning in various fields, including object detection [76] and segmentation [93]. The success of deep learning falls back on its proficiency in function approximation and the sophisticated properties of its convolutional networks. The competence of DRL in solving high-dimensional complex autonomous driving problems is presented in [33, 14]. Some of the prevalent DRL algorithms include DQN [25, 26], DDQN [27], PPO [28], DDPG [29], TD3 [30], and SAC [31]. The majority of the DRL-based autonomous driving applications target solving single scenarios such as lane keeping [117], lane change [120], lane merging [137], and roundabout navigation [142]. Nevertheless, designing a policy for multi-scenario cases like urban autonomous driving remains underexplored.

Kendall et al. [117] developed the first application of DRL in autonomous driving, where a single camera image is used to learn a lane keeping DDPG-based policy. Ye et al. [120] performs lane change using PPO in scenarios that involve 5 surrounding vehicles, and the state of each vehicle

is represented with 5 low-dimensional variables. Bouton et al. [137] proposes lane merging that interacts with traffic participants in a crowded scene using DQN. The inputs used in [137] are the vehicles' low-dimensional physical and internal states. Lastly, Chen et al. [142] proposes learning a driving policy from low-dimensional visual encodings. The encodings are captured from a semantic map in BEV representation constituting road geometry, routing, historical detected objects, and historical ego state information. The authors evaluated the proposed approach on three DRL algorithms (DDPG, TD3, and SAC) in a roundabout scenario under heavy traffic conditions. Experiments showed superior performance for SAC over the other two algorithms, where the learned policy completed the assigned task more frequently than the others.

Several works exist that explore urban driving using DRL algorithms. Agarwal et al. [22] achieves an urban driving policy using the PPO algorithm. Low-dimensional inputs are employed that enfold a latent encoding of several BEV semantically segmented images, trajectory waypoints, kinematic features, and traffic light states. The benefits of combining low-dimensional inputs, especially the encodings, are presented in [22], demonstrating an enhanced performance for the urban driving policy. Ahmed et al. [23] uses a direct perception method to train a DDPG-based policy in an urban environment. The prediction model takes in sequential camera images to generate affordance predictions. These predictions are then concatenated with state observations and vehicle measurements to form the input to the DDPG training agent. Reported results show that using predictions as input to the DRL algorithm performs better than training a DRL algorithm merely on raw camera image data. Furthermore, Chen et al. [24] proposes using sequential latent maximum entropy (MaxEnt) RL for learning a driving policy in an urban environment. The authors feed the system with a combination of input images, including LIDAR data in BEV form, camera image, and semantic map in BEV form. An encoding for each input is extracted in the environmental model before being fed into the MaxEnt RL model. Results show superior performance over DQN, DDPG, TD3, and SAC.

The preceding papers demonstrate that training DRL algorithms on encoded input features offer an advantage over raw input data training—this aids in simplifying and enhancing the DRL algorithm's training capabilities, resulting in higher performance. Going backward and substituting the incoming sensor data with predictions generates more compact features of lower dimensionality.

However, no work has been done yet that exploits LIDAR-based multi-modal fusion to extract pixel-wise joint perception and motion prediction and then leverage these predictions as part of an input to train a sequential latent MaxEnt RL algorithm. To that end, this chapter proposes using our multi-modal fusion network named *LiCaNext* to extract joint perception and motion prediction. These accurate predictions are further encoded with another simple BEV image to produce a latent representation of the surrounding scene that is both comprehensive and compact. Finally, the latent encodings are utilized by the MaxEnt RL to train a sophisticated driving policy that ensures safety, efficiency, and comfort.

6.3 Proposed Approach

The overview of our proposed approach is presented in Figure 6.1. The designed sequential latent MaxEnt RL algorithm learns a driving policy that navigates in an urban environment while ensuring safety, efficiency, and comfort. The sequential latent MaxEnt RL algorithm consists of environmental and RL models. The sequential latent environmental model learns a low-dimensional latent space from observations. This latent space is then used to train the MaxEnt RL for driving policy learning. The environment and the MaxEnt RL models are trained end-to-end. A detailed explanation of the sequential latent MaxEnt RL architecture and its learning process is provided in Section 6.3.1.

The use of low-dimensional learned latent space to train the MaxEnt RL model reduces complexity and enhances its training process leading to a better policy. This chapter proposes incorporating accurate pixel-wise perception and motion prediction with another simple BEV image into the sequential latent environmental model. These predictions hold rich and complementary information about the objects in the surrounding environments, including their dynamicity. Furthermore, the dimensionality of these predictions is significantly less than the original multi-modal images used to generate them. Thus, using accurate perception and motion prediction with another simple BEV image as input into the sequential latent model leads to a more informative and less complex latent space. This positively influences MaxEnt RL training and pushes performance even further. The exploited multi-modal images are BEV, RV, and range residual images, as illustrated in Figure 6.1.

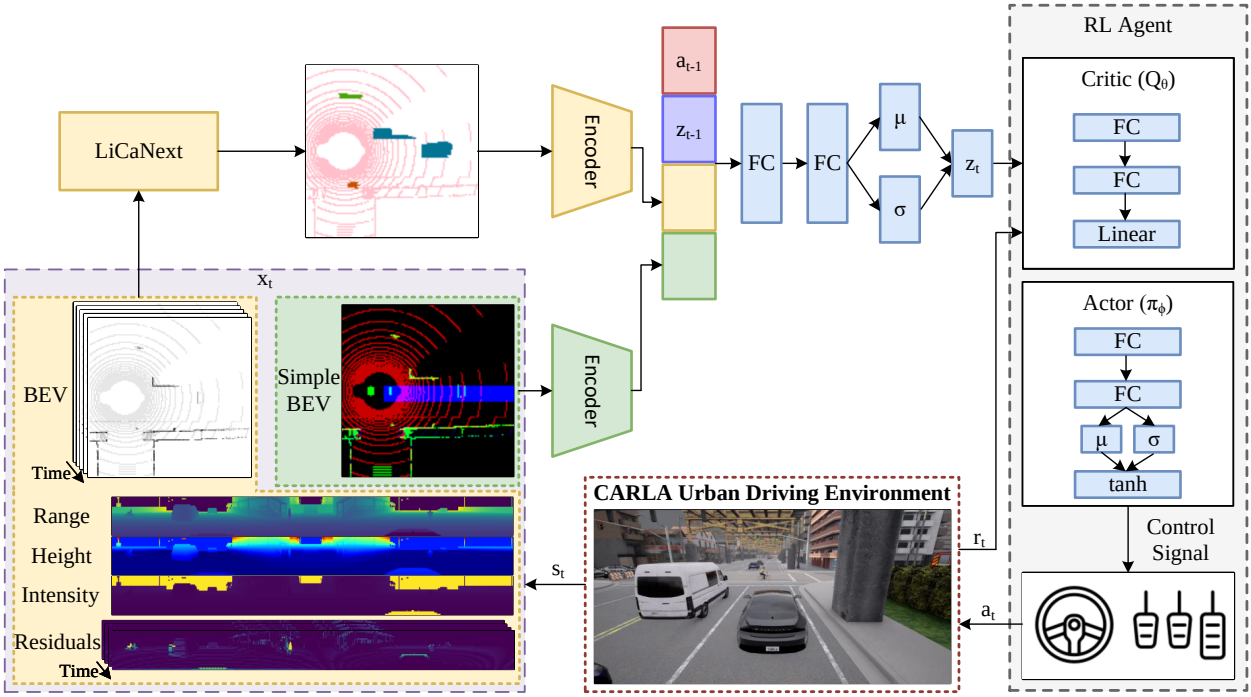


Figure 6.1 An overview of our proposed architecture. In our architecture, four images x_t are extracted from a single LIDAR sensor to represent the current state s_t. Three representations (BEV, RV, and residuals) are first fed into *LiCaNext* to generate accurate perception and motion prediction. These predictions are then fused with a simple BEV image, previous latent state z_{t-1}, and previous action a_{t-1} to obtain the latent state z_t. The RL agent trains on the low-dimensional z_t to learn a driving policy in an urban environment. The RL agent interacts with the environment by selecting an action a_t, the environment responds by sending the agent a reward r_t and evolves into a new state. FC represents a fully connected layer, a Gaussian layer is denoted by μ and σ, and tanh denotes a tanh bijector. The control signal holds throttle and steering values.

A novel real-time multi-modal fusion network named *LiCaNext* is used to extract accurate pixel-wise perception and motion prediction. The previous chapter provides extensive details on the architecture of *LiCaNext*, the formulation of its multi-modal input representations, and the generated perception and motion prediction (Section 5.3). Figure 6.2 illustrates *LiCaNext* architecture minus the camera module. Section 6.4.3 presents addition details on the simple BEV and the *LiCaNext*'s output images.

6.3.1 Sequential Latent MaxEnt RL

DRL algorithms generally attempt to learn representations and the assigned task to maximize the total discounted reward. The process of learning both in one model is inefficient because

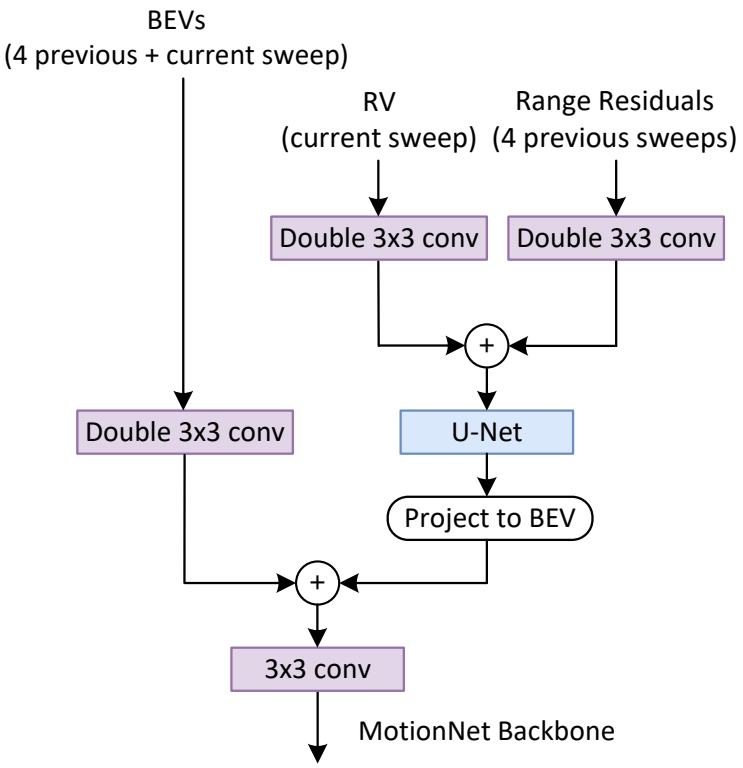


Figure 6.2 *LiCaNext* fusion scheme without the camera module.

instead of the DRL algorithm focusing entirely on learning the specified task, a large portion of its training is dedicated to representation learning, hence acting as a bottleneck. Conversely, the sequential latent MaxEnt RL algorithm performs these two tasks simultaneously but in separate models. A sequential environmental latent model is built just for representation learning, and a MaxEnt RL model is merely for task learning. The RL model trains on the low-dimensional latent space learned from the sequential environmental model. Therefore, learning a latent space allows distinguishing representation learning and reinforcement learning; thus, task learning is no longer learned directly from observations. The two models of our algorithm are constructed and connected using probabilistic graphical models (PGMs). Recently, PGM has been applied extensively in representing and connecting sequential time processes, latent variables, and RL [173, 32, 174, 24, 146]. The key reason behind the success of PGMs is their immense potential in behavior estimation. Extensive background information on RL and its formulation using MDP is presented in Section 2.2.

The architecture of the sequential latent MaxEnt RL algorithm structured in PGM is presented

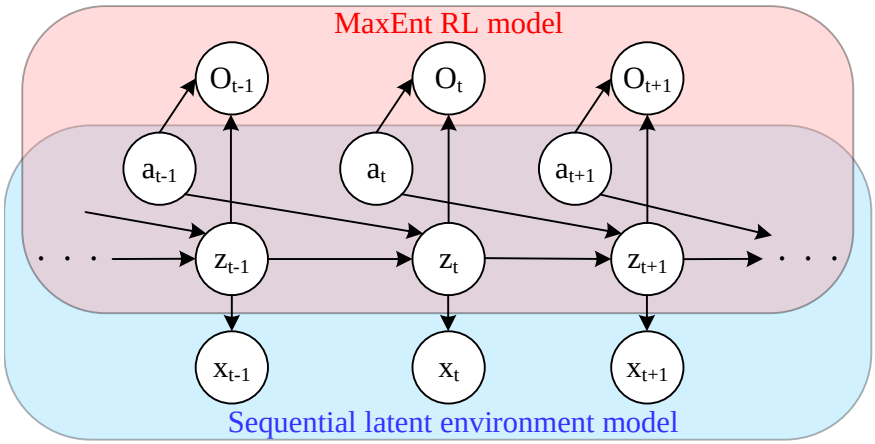


Figure 6.3 A PGM for the sequential latent MaxEnt RL architecture.

in Figure 6.3. Blue represents the sequential latent environmental model, while red denotes the MaxEnt RL model. In Figure 6.3, the high-dimensional partially observable input at timestep t is represented by the variable x_t, the executed action denoted by a_t, and the latent state summarizing the historical information signified by z_t. These variables constituting the environmental PGM model allows it to accurately mimic the real environment where states evolve with time and are partially observable with high dimensionality. In terms of the MaxEnt RL model, a notion of reward is included and is denoted by O_t to represent the optimality indicator. Edges in PGM represent conditional dependence between the terms. A decoding of the latent state z_t into the input x_t is performed by the observation generative model $p(x_t|z_t)$. The state transition model $p(z_{t+1}|z_t,a_t)$ determines the next state z_{t+1} based on the current state z_t and the executed action a_t. The initial latent state z_1 is generated from $p(z_1) = p(z_1|x_1)$. Further information is provided below on $p(O_t|z_t,a_t)$, and the variational approximate posterior $q(z_t|x_t)$ that acts as the encoder from x_t to z_t. In addition, given x_{t+1} a filtering function $q(z_{t+1}|z_t,x_{t+1},a_t)$ can be inferred in a recursive bayesian filter way.

a) Objective of MaxEnt RL as PGM: As our algorithm is based on using MaxEnt RL formulated as an inference problem in PGM, this section provides details showing that the objective functions of MaxEnt RL and its PGM extension match. This would prove that MaxEnt RL can be optimized by learning a PGM. Typically, the objective of MaxEnt RL is to maximize the expected total discounted reward and the entropy regularization term as defined in (6.1). The entropy is

defined by $-\log \pi_\phi(a_t|z_t)$, where ϕ is the learned weights of the deep neural network for policy π_ϕ. The reward discount factor γ is implicitly included in the transition function and thus is not included in all upcoming equations.

$$\phi^* = \underset{\phi}{\text{argmax}} \underset{\substack{z_1 \backsim p(z_1) \\ a_t \backsim \pi_\phi(a_t|z_t) \\ z_{t+1} \backsim p(z_{t+1}|z_t, a_t)}}{\mathbb{E}} \sum_{t=1}^{T} \big(r(z_t, a_t) - \log \pi_\phi(a_t|z_t)\big) \tag{6.1}$$

where T marks the end of a trajectory.

In Figure 6.3, the PGM structured MaxEnt RL model includes an optimality term O_t, which represents optimality of reward given the state and action. This means that maximizing the optimality across all states and actions in the trajectory encourages the agent to take highly rewarding actions, thus attaining an optimal policy. The distribution of O_t being optimal is defined in (6.2).

$$p(O_t = 1|z_t, a_t) = \exp(\, r(z_t, a_t)\,) \tag{6.2}$$

where $O_t = 1$ donates optimality at timestep t. The optimal trajectory distribution conditioned to $O_t = 1$ is presented in (6.3).

$$\begin{aligned} p(O_{1:T}, z_{1:T}, a_{1:T}) &= p(O_{1:T}|z_{1:T}, a_{1:T}) p(z_{1:T}, a_{1:T}) \\ &= p(z_1) \prod_{t=1}^{T-1} p(z_{t+1}|z_t, a_t) \prod_{t=1}^{T} p(O_t = 1|z_t, a_t) \\ &= p(z_1) \prod_{t=1}^{T-1} p(z_{t+1}|z_t, a_t) \prod_{t=1}^{T} \exp(r(z_t, a_t)) \\ &= \left[p(z_1) \prod_{t=1}^{T-1} p(z_{t+1}|z_t, a_t) \right] \exp\left(\sum_{t=1}^{T} r(z_t, a_t) \right) \end{aligned} \tag{6.3}$$

where subscript $1:T$ denotes an entire trajectory. The distribution in (6.3) is the probability of observing a trajectory, where the trajectory with the highest reward gets the highest probability.

Variational inference is used to optimize actions, which approximates $p(O_{1:T}, z_{1:T}, a_{1:T})$ via optimizing the variational distribution $q(z_{1:T}, a_{1:T})$. The distribution $q(z_{1:T}, a_{1:T})$ reflects the tra-

jectory distribution under the current policy $\pi_\phi(a_t|z_t)$, as defined in (6.4).

$$\begin{aligned} q(z_{1:T}, a_{1:T}) &= q(z_1) \prod_{t=1}^{T-1} q(z_{t+1}|z_t, a_t) \prod_{t=1}^{T} q(a_t|z_t) \\ &= p(z_1) \prod_{t=1}^{T-1} p(z_{t+1}|z_t, a_t) \prod_{t=1}^{T} \pi_\phi(a_t|z_t) \end{aligned} \tag{6.4}$$

In variational inference, approximate inference is performed by optimizing the evidence lower bound (ELBO). Here the evidence is O_t and so its log probability for all $t \in 1, ..., T$ is given by:

$$\begin{aligned} \log p(O_{1:T}) &= \log \int \int p(O_{1:T}, z_{1:T}, a_{1:T}) dz_{1:T} da_{1:T} \\ &= \log \int \int p(O_{1:T}, z_{1:T}, a_{1:T}) \times \frac{q(z_{1:T}, a_{1:T})}{q(z_{1:T}, a_{1:T})} dz_{1:T} da_{1:T} \\ &= \log \mathop{\mathbb{E}}_{q(z_{1:T}, a_{1:T})} \left[\frac{p(O_{1:T}, z_{1:T}, a_{1:T})}{q(z_{1:T}, a_{1:T})} \right] \end{aligned} \tag{6.5}$$

Using (6.3), (6.4), and applying Jensen's inequality on (6.5) results in the ELBO as given by:

$$\begin{aligned} \log p(O_{1:T}) &\geq \mathop{\mathbb{E}}_{q(z_{1:T}, a_{1:T})} \left[\log p(O_{1:T}, z_{1:T}, a_{1:T}) - \log q(z_{1:T}, a_{1:T}) \right] \\ &\geq \mathop{\mathbb{E}}_{q(z_{1:T}, a_{1:T})} \sum_{t=1}^{T} \left[r(z_t, a_t) - \log \pi_\phi(a_t|z_t) \right] \end{aligned} \tag{6.6}$$

Finally, comparing (6.1) and (6.6) confirms that the objectives of MaxEnt RL and its PGM extension are equivalent, as optimizing the objective in (6.6) with respect to the policy π_ϕ matches the objective of (6.1). Thus, proving that it is valid to formulate MaxEnt RL in a PGM. The advantages of using MaxEnt RL in our algorithm over standard RL are first as we have just shown that MaxEnt RL can be formulated and learned in a PGM. Second, MaxEnt RL has exploration embedded by default in its policy.

b) Variational Inferencing and Learning for the Joint Models: As aforementioned, the sequential latent MaxEnt RL algorithm consists of the environmental model and the MaxEnt RL

model. In addition, both models structured in PGM are jointly learned. Learning our PGM, presented in Figure 6.3, requires solving the objective of maximizing the log likelihood of the inputs and the optimality variables. The learning objective $p(x_{1:\tau+1}, O_{\tau+1:T}|a_{1:\tau})$, defined in (6.7), includes the likelihood of the past inputs and the optimality of future actions, thus incorporating learning of both the environmental and the MaxEnt RL models.

$$\log \prod_{(x_{1:\tau+1},a_{1:\tau},r_{1:\tau})\in\mathcal{D}} p(x_{1:\tau+1}, O_{\tau+1:T}|a_{1:\tau}) = \sum_{(x_{1:\tau+1},a_{1:\tau},r_{1:\tau})\in\mathcal{D}} \log p(x_{1:\tau+1}, O_{\tau+1:T}|a_{1:\tau}) \tag{6.7}$$

where $\mathcal{D}$ represents a replay buffer that stores trajectories composed of observational inputs, actions and rewards. τ represents the current timestep, while T is the last timestep in a trajectory.

As attempted in Section 6.3.1 to solve for optimality, variational inferencing is used again to solve for $\log p(x_{1:\tau+1}, O_{\tau+1:T}|a_{1:\tau})$. The variational distribution used is $q(z_{1:T}, a_{\tau+1:T}|x_{1:\tau+1}, a_{1:\tau})$. The log likelihood $\log p(x_{1:\tau+1}, O_{\tau+1:T}|a_{1:\tau})$ is defined in (6.8).

$$\begin{aligned}
\log p(x_{1:\tau+1}, O_{\tau+1:T}|a_{1:\tau}) &= \log \int\int p(x_{1:\tau+1}, O_{\tau+1:T}, z_{1:T}, a_{\tau+1:T}|a_{1:\tau}) dz_{1:T} da_{\tau+1:T} \\
&= \log \int\int p(x_{1:\tau+1}, O_{\tau+1:T}, z_{1:T}, a_{\tau+1:T}|a_{1:\tau}) \\
&\qquad \times \frac{q(z_{1:T}, a_{\tau+1:T}|x_{1:\tau+1}, a_{1:\tau})}{q(z_{1:T}, a_{\tau+1:T}|x_{1:\tau+1}, a_{1:\tau})} dz_{1:T} da_{\tau+1:T} \\
&= \log \mathop{\mathbb{E}}_{(z_{1:T},a_{\tau+1:T})\sim q} \left[\frac{p(x_{1:\tau+1}, O_{\tau+1:T}, z_{1:T}, a_{\tau+1:T}|a_{1:\tau})}{q(z_{1:T}, a_{\tau+1:T}|x_{1:\tau+1}, a_{1:\tau})} \right] \\
&\geq \mathop{\mathbb{E}}_{(z_{1:T},a_{\tau+1:T})\sim q} \Big[\log p(x_{1:\tau+1}, O_{\tau+1:T}, z_{1:T}, a_{\tau+1:T}|a_{1:\tau}) \\
&\qquad - \log q(z_{1:T}, a_{\tau+1:T}|x_{1:\tau+1}, a_{1:\tau}) \Big]
\end{aligned} \tag{6.8}$$

The above ELBO is achieved by replacing integration with expectation and applying Jensen's inequality. The variational distribution $q(z_{1:T}, a_{\tau+1:T}|x_{1:\tau+1}, a_{1:\tau})$ is defined in (6.9). The agent is prevented from controlling transitions and from selecting optimistic actions as future dynamics are

embedded in the variational distribution.

$$q(z_{1:T}, a_{\tau+1:T}|x_{1:\tau+1}, a_{1:\tau}) = \prod_{t=0}^{\tau} q(z_{t+1}|z_t, x_{t+1}, a_t) \prod_{t=\tau+1}^{T-1} p(z_{t+1}|z_t, a_t) \prod_{t=\tau+1}^{T} \pi(a_t|z_t) \tag{6.9}$$

Now we extract the joint likelihood from our PGM in Figure 6.3:

$$p(x_{1:\tau+1}, O_{\tau+1:T}, z_{1:T}, a_{\tau+1:T}|a_{1:\tau}) = \prod_{t=1}^{\tau+1} p(x_t|z_t) \prod_{t=0}^{T-1} p(z_{t+1}|z_t, a_t) \prod_{t=\tau+1}^{T} p(O_t|z_t, a_t) \prod_{t=\tau+1}^{T} p(a_t) \tag{6.10}$$

After substituting (6.9) and (6.10) in the ELBO (6.8) and introducing Kullback-Leibler divergence D_{KL}, we get:

$$\begin{aligned} \log p(x_{1:\tau+1}, O_{\tau+1:T}|a_{1:\tau}) = & \mathop{\mathbb{E}}_{z_{1:\tau+1}\sim q} \left[\sum_{t=0}^{\tau} \log p(x_{t+1}|z_{t+1}) - D_{KL}\big(q(z_{t+1}|z_t, x_{t+1}, a_t) \,||\, p(z_{t+1}|z_t, a_t)\big) \right] \\ & + \mathop{\mathbb{E}}_{(z_{\tau+1:T}, a_{\tau+1:T})\sim q} \left[\sum_{t=\tau+1}^{T} \Big(r(z_t, a_t) - \log \pi(a_t|z_t) + \log p(a_t) \Big) \right] \end{aligned} \tag{6.11}$$

The ELBO presented in (6.11) encompasses the learning of the environmental model and the MaxEnt RL model. The part of (6.11) enclosed in blue brackets is related to the environmental model, while the part within red brackets corresponds to the MaxEnt RL model. Therefore, to learn both environmental and MaxEnt RL models, the ELBO objectives in (6.11) need to be optimized.

Learning the environmental model leads to maximizing the likelihood of observations. Five neural networks (parameterized with ψ) exist in the environmental model: $p_\psi(z_1)$, $p_\psi(z_{t+1}|z_t, a_t)$, $p_\psi(x_t|z_t)$, $q_\psi(z_1|x_1)$, and $q_\psi(z_{t+1}|z_t, x_{t+1}, a_t)$. The resulting objective function $J_M(\psi)$, with ψ dependent terms, is as follows:

$$J_M(\psi) = \mathop{\mathbb{E}}_{z_{1:\tau+1}\backsim q_\psi} \left[\sum_{t=0}^{\tau} -\log p_\psi(x_{t+1}|z_{t+1}) + D_{KL}\big(q_\psi(z_{t+1}|z_t, x_{t+1}, a_t) \,||\, p_\psi(z_{t+1}|z_t, a_t)\big) \right] \tag{6.12}$$

The objective of the driving policy model enclosed in the red brackets in (6.11) is simply a standard MaxEnt RL problem. The term $\log p(a_t)$ is removed from the driving policy objective function as a uniform action prior is assumed, thus acting as a constant. Soft actor-critic (SAC) is used to solve the MaxEnt RL optimization problem through message passing of soft Q-values that are reliant on latent states z. Minimizing the Bellman residual in (6.13) optimizes parameters θ of the soft Q-function which approximates the messages.

$$J_Q(\theta) = \mathop{\mathbb{E}}_{z_{1:\tau+1} \backsim q_\psi} \left[\frac{1}{2} \Big(Q_\theta(z_\tau, a_\tau) - \hat{Q}_\theta(z_\tau, a_\tau) \Big)^2 \right] \tag{6.13}$$

where $\hat{Q}_\theta$ with delayed target network parameters $\bar{\theta}$ is defined as:

$$\hat{Q}_\theta(z_\tau, a_\tau) = r_\tau + \gamma \mathop{\mathbb{E}}_{a_{\tau+1} \backsim \pi_\phi} \Big[Q_{\bar{\theta}}(z_{\tau+1}, a_{\tau+1}) - \log \pi_\phi(a_{\tau+1} | z_{\tau+1}) \Big] \tag{6.14}$$

Lastly, the policy loss function is given by the following equation:

$$J_\pi(\phi) = \mathop{\mathbb{E}}_{\substack{z_{1:\tau+1} \backsim q_\psi \\ a_{\tau+1} \backsim \pi_\phi}} \Big[\log \pi_\phi(a_{\tau+1} | z_{\tau+1}) - Q_\theta(z_{\tau+1}, a_{\tau+1}) \Big] \tag{6.15}$$

Further details on the neural network architectures of the environmental and MaxEnt RL model is provided in Section 6.4.3. To conclude, the loss functions for optimizing the joint models are $J_M(\psi)$, $J_Q(\theta)$ and $J_\pi(\phi)$.

6.4 Implementation

6.4.1 Experimental Setup

All experiments in this chapter are conducted on open source CARLA simulator version 0.9.12 [127]. CARLA provides a high fidelity urban driving environment with many traffic cases. It supports complete control of static and dynamic actors, sensor suits, various weather conditions, and maps covering a wide range of realistic scenarios (i.e., intersections, roundabouts, highways, and bridges). Training is performed in the most complex map provided by CARLA (Town03) and un-

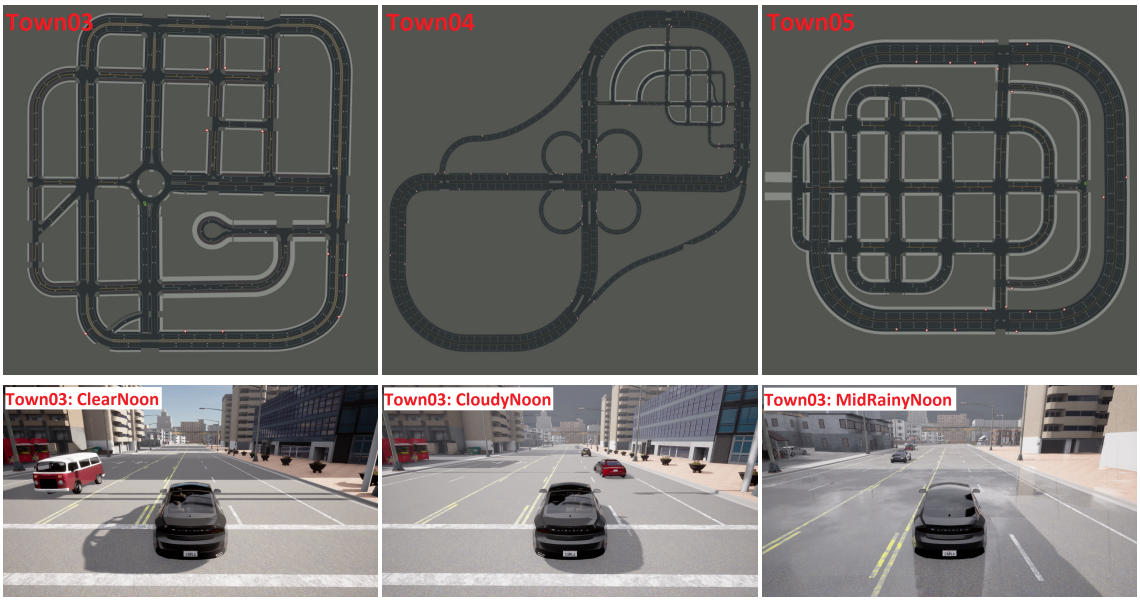


Figure 6.4 Top row: Map layout of the different environments used in our experiments. Bottom row: Examples of different weather conditions in Town03 environment.

der a clear sunny environment (ClearNoon). Further analysis is performed on our proposed learned policy where additional metrics are reported on three maps (Town03, Town04, and Town05) and under varying weather conditions (ClearNoon, CloudyNoon, and MidRainyNoon). A sample view of the different maps' layout and the weather conditions are illustrated in Figure 6.4.

In the environment, 100 surrounding vehicles and up to 100 pedestrians are deployed. Bikes are also spawned in the environment and are considered within the 100 sampled vehicles, as CARLA classifies bikes under the vehicles category. All spawned surrounding vehicles and pedestrians are controlled by CARLA and thus abide by the traffic rules. This means that the spawned vehicles and pedestrians do not collide with other agents, decelerate and eventually come to a complete stop when their path is blocked. Additionally, they continue following their dedicated route after their path is cleared. All spawned agents are assigned routes by the CARLA planner.

6.4.2 Reward Function

The objective of our trained agent (ego-vehicle) is to achieve safety, efficiency, and comfort in an urban environment. The scalar reward signal is the key factor in optimizing the agent's performance to realize the defined objectives. Usually, a learning agent is motivated to avoid undesirable

actions by assigning negative reward values acting as a penalty. Conversely, positive reward values are assigned to incentivize actions that get the agent closer to the objectives. Our reward function is designed to incorporate: 1) Safety by evaluating collisions, distance to lane center, and not exceeding the maximum defined speed; 2) Efficiency by evaluating speed, steering, lateral acceleration, and travel time; 3) Comfort by evaluating angular velocity. Our reward function r with all these components taken into consideration is defined in (6.16).

$$r = r^s_{col} + r^s_{lane} + r^s_{fast} + r^e_{vel} + r^e_{steer} + r^e_{lat} + r^e_{step} + r^c_{\omega} \tag{6.16}$$

r^s_{col} is used to penalize the agent for collisions. Due to the severity of a collision, the largest negative reward value is awarded to r^s_{col} relative to the other reward components. This stimulates the agent to avoid a collision at all costs. Another factor ensuring passenger safety is keeping the agent on the lane center. r^s_{lane} in (6.17) encourages the agent to stay as close as possible to the lane center by punishing the agent with $dist$, which is how far it is from the lane center. In case the agent runs outside the lane defined by the threshold $laneCenter_{max}$, the punishment jumps by -1. $\mathbb{1}$ denotes an indicator function.

$$r^s_{lane} = -|dist| - \mathbb{1}\left(|dist| > laneCenter_{max}\right) \tag{6.17}$$

The last component in promoting safety is r^s_{fast}, which restricts agent's speed vel to a maximum speed limit vel_{max}. r^s_{fast} is defined as follows:

$$r^s_{fast} = 5\left(vel_{max} - vel\right)\mathbb{1}\left(vel > vel_{max}\right) \tag{6.18}$$

In terms of efficiency, the agent is encouraged to drive as fast as possible by awarding it with its current speed weighted by the heading similarity $head_{sim}$. $head_{sim}$ is the cosine similarity between the agent's heading direction and the followed route. This reward is scaled by 0.5 and only rewarded if $head_{sim} \geq head_{max}$, else double that amount is assigned as a penalty to promote speeding

Table 6.1 Parameters of the reward function r.

Parameter	Value
r^s_{col}	-200
$laneCenter_{max}$	2m
vel_{max}	14m/s
$head_{max}$	0.75
r^e_{step}	-0.1
ω_{max}	20deg/s

up in the direction of the route. The speed reward component r^e_{vel} is given by:

$$r^e_{vel} = |head_{sim}|\, vel \left[0.5 - 1.5\, \mathbb{1}\left(head_{sim} < head_{max}\right)\right] \tag{6.19}$$

Furthermore, $r^e_{steer} = -5\, steer^2_{\alpha}$ is used to inspire the agent to steer, where $steer_{\alpha}$ represents the steering angle of the agent. Moreover, lateral acceleration is included in the reward $r^e_{lat} = -0.2\, |steer_{\alpha}|\, vel^2_{sim}$. The variable vel_{sim} denotes the cosine similarity between vel and the route direction. The last term that stimulates the efficiency of the driving agent is r_{step}. This adds a small negative value to r for incentivizing the agent to accomplish the task as quickly as possible.

Finally, r^c_{ω} is used to promote comfort by controlling the agent speed and the heading direction according to angular velocity ω. This rewards the agent by the value $0.5\, head_{sim}\, vel$, if $\omega \leq \omega_{max}$. Otherwise, the reward represented by $head_{sim}\, vel$ is controlled by the magnitude of ω, encouraging the agent to lower its speed when ω is high, leading to smoother steering. Similarly, it encourages the agent to increase speed when ω is low, i.e., driving straight. This also improves smoothness by controlling the amount of oscillation (right and left motion) while driving straight. All constants used in r are defined in Table 6.1.

$$r^c_{\omega} = head_{sim}\, vel \left[\, |\omega|^{-1}\, \mathbb{1}\left(|\omega| > \omega_{max}\right) + 0.5\, \mathbb{1}\left(|\omega| \leq \omega_{max}\right)\right] \tag{6.20}$$

6.4.3 Implementation Details

In this section, implementation details of *LiCaNext* on TensorRT v8.2 with its procured outcomes are provided. Second, the architectures of the neural networks that exist in the sequential latent MaxEnt RL algorithm are demonstrated. Next, details on the sensor inputs to our proposed approach are presented. Last, this section illustrates how training and evaluation are performed with their defined hyperparameters.

a) *LiCaNext*: *LiCaNext* is trained on nuScenes dataset [66]. Scenes in the nuScenes dataset are divided as follows: 500 scenes for training, 100 for validation, and 200 for testing. The specifications of the LIDAR sensor used in nuScenes dataset has 32 laser beams, 20Hz capture frequency, 360° horizontal field-of-view (FOV), and a vertical FOV ranging from $-30.67°$ to $10.67°$. As mentioned in Section 5.3, *LiCaNext* is trained with BEV input of dimensions $256 \times 256 \times 13 \times 5$, RV images of dimensions $1024 \times 32 \times 4$, and range residual image of dimensions $1024 \times 32 \times 4$. The BEV voxel resolution is $(0.25, 0.25, 0.4)$m and the defined range for the LIDAR points is $[-32, 32] \times [-32, 32] \times [-3, 2]$m. Only BEV and range residual images consist of temporal information extracted from 4 past frames. All historic frames are sampled with a time span of 0.2s, while the current frames are sampled at 2Hz for training and 1Hz for testing.

In terms of *LiCaNext* output, the perception and motion prediction are in the form of BEV images with dimensions of $256 \times 256 \times 5$ for the cell classification output head, $256 \times 256 \times 2 \times 20$ for the motion prediction output head, and $256 \times 256 \times 2$ for the state estimation output head. The first two dimensions in all output heads represent the length and width dimensions of the BEV image. The third dimension in the cell classification head represents the number of class categories. The motion prediction head predicts the forecasted pixel position for the next 20 frames, and lastly, the state estimation predicts whether each pixel in the BEV is static or dynamic. The output BEV image (a sample is presented in Figure 6.1) is constructed by using the furthest forecasted frame to represent the motion predictions. Additionally, the pixels that the state estimation head predicts as static, their corresponding motion is set to 0. Furthermore, the maximum predicted classification is used to determine the category of pixels. Thus, the *LiCaNext* BEV output image is of dimensions

$256 \times 256 \times 3$. The color codes used for the *LiCaNext* output image in Figure 6.1 are pink for background, blue for vehicles, orange for pedestrians, and green for bikes.

LiCaNext is trained using the same parameters as in the previous chapter. Once the trained *LiCaNext* model is attained, further optimization is applied using TensorRT v8.2 [45] to gain additional speed efficiency. TensorRT is a platform by NVIDIA for high-performance deep learning inference. TensorRT internally applies some optimization techniques such as layer fusion to enhance latency, power efficiency, memory consumption, and the throughput of a trained network. The output of the TensorRT optimizer is an optimized inference engine that accelerates inferencing. Table 6.2 displays results for *LiCaNext* and the TensorRT-based *LiCaNext*. Results for MotionNet are also included to show that the accuracies procured in the TensorRT-based network are still significantly better.

The metrics used to measure perception are mean classification error (MCA) - representing the average classification accuracy of the five category classes, and the overall accuracy (OA) - denoting the average classification accuracy over all pixels. The correctness of motion prediction is measured by L_2 displacement error for two-speed groups: static and dynamic. The static group is defined for pixels with no motion, while the dynamic group is for pixels with motion. The reason for distinguishing pixels into two groups based on their motion is to avoid bias due to the large percentage of staticity in a scene. Mean and median errors are computed for each group. According to Table 6.2, TensorRT-based *LiCaNext* achieved $1.82\times$ speedup compared to *LiCaNext* with only 0.6% and 0.2% trade-off in MCA and OA, respectively. Compared with MotionNet, TensorRT-based *LiCaNext* results are still higher by 3.1% and 0.4% in MCA and OA, respectively. Furthermore, the overall displacement errors have slightly increased for TensorRT-based *LiCaNext* compared to *LiCaNext*, but are still significantly lower than MotionNet. A minor loss in accuracy is compromised for a $1.82\times$ speedup as this multi-modal fusion network is used as part of a learning process for autonomous driving where speed is of paramount importance. It is worth noting that only in this section (Section 6.4.3), we differentiate between TensorRT-based *LiCaNext* and *LiCaNext*. In the rest of this chapter, TensorRT-based *LiCaNext* is refered to as *LiCaNext* to represent the multi-modal fusion network used in our proposed approach.

Table 6.2 Perception and motion prediction comparison between MotionNet, *LiCaNext*, and TensorRT-based *LiCaNext*.

Method	Static		Dynamic		Class. Acc. (%)		Time (ms)
	Mean	Median	Mean	Median	MCA	OA	
MotionNet [34]	0.0236	0	0.4104	0.1369	70.3	96.3	20.5
LiCaNext [167]	0.0222	0	**0.3875**	**0.1367**	**74.0**	**96.9**	29.7
TensorRT-based *LiCaNext*	**0.0218**	0	0.3912	0.1369	73.4	96.7	**16.3**

b) Sensor Inputs: As mentioned earlier, our proposed approach is dependent on one LIDAR sensor. Similar to the LIDAR sensor specifications used in the nuScenes dataset, the CARLA LIDAR sensor planted on the ego-vehicle is designed to have 32 laser beams, capture frequency of 20Hz, 360° horizontal FOV, and vertical FOV ranging from $-30.67°$ to $10.67°$. LIDAR sweeps are used to extract the BEV, RV, and residual range images to represent *LiCaNext* input. These perception and motion prediction are then fed with another simple BEV image into the sequential latent MaxEnt RL algorithm to learn urban autonomous driving. *LiCaNext* BEV input dimensions are set to $128 \times 128 \times 13 \times 5$. The voxel resolution is set to $(0.25, 0.25, 0.4)$m in the xyz-axis and the defined range to $[-16, 16] \times [-7, 25] \times [-3, 2]$m in the xyz-direction. We design the RV image to have dimensions of $1024 \times 32 \times 4$, where 4 channels represent the range, height, intensity, and a binary flag. Similarly, the range residual images are set to have dimensions of $1024 \times 32 \times 4$. The length of both RV and residual images is set to 32 to reflect the number of laser beams in the LIDAR sensor. Furthermore, all sweeps used in both BEV and range residuals are 0.1s apart.

The simple BEV is the only input image fed directly into the sequential latent MaxEnt RL without being passed into *LiCaNext*. This image is simple as it groups LIDAR points into just three height channels: short, medium, and tall. The short height group holds pixels at the ground or near-ground level; these pixels are mostly static. The medium height group encompasses most dynamic pixels representing vehicles, pedestrians, and bikes. The tall group mostly embraces pixels representing tall objects, which are mainly static (i.e., buildings, trees, and light poles). The defined range of the LIDAR points in the simple BEV image is $[-16, 16] \times [-7, 25] \times [-3, 2]$m in the xyz-direction. The z-direction is with respect to the LIDAR height position. The LIDAR is spawned on the ego-vehicle and is 2.1m above ground level. The length and width of the voxel

resolution are $(0.25, 0.25)$ in the xy-axis. However, the voxel resolution in the z-axis is different in the short height group compared to the other two groups. The short group has a height of 1.3m, representing LIDAR points defined from $[-3, -1.7]$m, and 1.85m for the medium and tall groups defined from $[-1.7, 0.15]$m and $[0.15, 2]$m in the z-direction, respectively. Thus, the dimensions of the simple BEV input image are $128 \times 128 \times 3$. Furthermore, the simple BEV image includes the route (denoted by blue) to be followed by the learning agent. The CARLA planner plans the route. As illustrated in Figure 6.1, pixels falling in the short, medium, and tall groups are denoted by red, light green, and dark blue colors in the simple BEV image.

c) Network Architecture of Sequential Latent MaxEnt RL: Now, the architectures of the neural networks present in the environmental loss function $J_M(\psi)$ and MaxEnt RL loss functions $J_Q(\theta)$ and $J_\pi(\phi)$; defined in equations (6.12), (6.13), and (6.15), respectively. The sequential latent environmental model includes five neural networks (parameterized with ψ): initial state distribution $p_\psi(z_1)$, state transition probability $p_\psi(z_{t+1}|z_t, a_t)$, decoder $p_\psi(x_t|z_t)$, and filtering model $q_\psi(z_1|x_1)$ and $q_\psi(z_{t+1}|z_t, x_{t+1}, a_t)$. The MaxEnt RL model consists of two networks: the Q network (parameterized with θ) $Q_\theta(z_t, a_t)$ and the policy network (parameterized with ϕ) $\pi_\phi(a_t|z_t)$. In our work, a two-layer hierarchy latent space structure is adopted as in [32], where $z_t^1 \in \mathbb{R}^{32}$ and $z_t^2 \in \mathbb{R}^{256}$. The network architectures are as follows:

- $\boldsymbol{p_\psi(z_1)}$ and $\boldsymbol{p_\psi(z_{t+1}|z_t, a_t)}$ constitutes two fully connected layers with 256 hidden units, followed by a Gaussian output layer.
- $\boldsymbol{p_\psi(x_t|z_t)}$ constitutes six deconvolutional layers in the following order $[(256, 4, 1)$, $(256, 3, 2)$, $(128, 3, 2)$, $(64, 3, 2)$, $(32, 3, 2)$, and $(3, 5, 2)]$. Each convolution layer is described by (number of filters, kernel size, and moving stride).
- $\boldsymbol{q_\psi(z_1|x_1)}$ and $\boldsymbol{q_\psi(z_{t+1}|z_t, x_{t+1}, a_t)}$ constitutes six convolutional layers in the following order $[(32, 5, 2)$, $(64, 3, 2)$, $(128, 3, 2)$, $(256, 3, 2)$, $(256, 3, 2)$, and $(256, 4, 1)]$, two fully connected layers with 256 hidden units, and a Gaussian output layer. The convolutional layers encodes the observations x_t into features, which are then applied into the fully connected layers and the Gaussian output layer to attain the latent state z_t.
- $\boldsymbol{Q_\theta(z_t, a_t)}$ constitutes two fully connected layers with 256 hidden units, and a linear output

layer.

- $\pi_\phi(a_t|z_t)$ constitutes two fully connected layers with 256 hidden units, a Gaussian layer, and a tanh bijector.

6.4.4 DRL Comparison Baselines

Evaluation of our proposed approach against state-of-the-art actor-critic baselines is performed to benchmark the performance and efficiency of our proposed approach. We compare our proposed approach to:

- **DDPG** [29] - A deterministic policy gradient algorithm that handles continuous action spaces. The Bellman operator is used to approximate the Q network, which is used to optimize the policy with policy gradient.
- **TD3** [30] - An improved algorithm over DDPG where function approximation errors are enhanced by taking the lowest value from two Q networks (critics) to set the target Q value, and by delaying policy updates.
- **SAC** [31] - An algorithm based on MaxEnt RL as it maximizes both the expected reward and entropy.

In addition, our proposed approach is compared to [24] and integration of MotionNet predictions. For all baseline algorithms, a simple BEV image (illustrated in Section 6.4.3) is used as input to the models. Except for the experiment that incorporates perception and motion prediction extracted from MotionNet, those MotionNet predictions are needed as input along with the simple BEV image. The baseline algorithms embrace the same encoding networks used in our proposed approach. Furthermore, to mimic time sequences used in the sequential latent MaxEntRL algorithm, a long short-term memory (LSTM) with a size of 40 and an output size of 100 are implemented in DDPG, TD3, and SAC.

6.4.5 Training and Evaluation Details

An episode is designed to terminate if the ego-vehicle collides, breaks out of the route by a threshold $laneCenter_{max}$, or successfully reaches the maximum number of timesteps (500). When an episode terminates, a new episode begins, and actors with the ego-vehicle are respawned in a

random valid position. The time step between consecutive frames is set to 0.1s. A speed efficient *LiCaNext* is used for all experiments. The sequential latent environmental model of our proposed approach is trained with a batch size of 32 and a learning rate of 1×10^{-4}, whereas the MaxEnt RL model is trained with a batch size of 64 and a learning rate of 3×10^{-4}. The length of the sequential model is set to 4, the discount factor to 0.99, and the buffer size to 10^5. Furthermore, a frame skip of 4 is used to keep actions unchanged for 4 environmental steps. The agent is trained for 200k environmental steps, and its performance is evaluated for 10 episodes every 5k environmental training steps. The same stochastic policy used in training is used for evaluation.

6.5 Experimental Results and Discussions

Results in Figure 6.5 show that our proposed approach outperforms the baseline algorithms. For each algorithm, 5 trials are executed. The solid curves and shaded regions correspond to the mean and standard deviation of the average returns relative to the environmental steps. It is evident from Figure 6.5 that DDPG and TD3 algorithms hardly perform any learning in such a complex environmental setup. The SAC algorithm learns a better policy compared to DDPG and TD3, mainly because of its exploration capability. Moreover, the sequential latent MaxEnt RL algorithm experiment with only a simple BEV image representing the input (LSAC (BEV)) outperforms the SAC algorithm. This confirms the effectiveness of improving sampling efficiency by separating environmental learning from reinforcement learning. Even though fusing MotionNet predictions (LSAC (BEV + MotionNet)) results in better performance than LSAC (BEV); however, fusing *LiCaNext* predictions pushed performance even further, resulting in the best driving policy compared to the other algorithms. This proves that exploiting multi-modal fusion to extract accurate perception and motion prediction and then inserting them as part of the input for training sequential latent MaxEnt RL algorithm further enhances performance. Our proposed approach (LSAC (BEV + *LiCaNext*)) learns to drive as fast as possible, with minimal oscillation and violation of maximum speed. The trained agent avoids collision by slowing down when a vehicle is ahead and eventually comes to a complete stop. In addition, the agent moves again when its path is cleared. Lastly, the learned driving policy successfully makes smooth turns. Thus, our proposed approach learns a driving policy that best ensures safety, efficiency, and comfort.

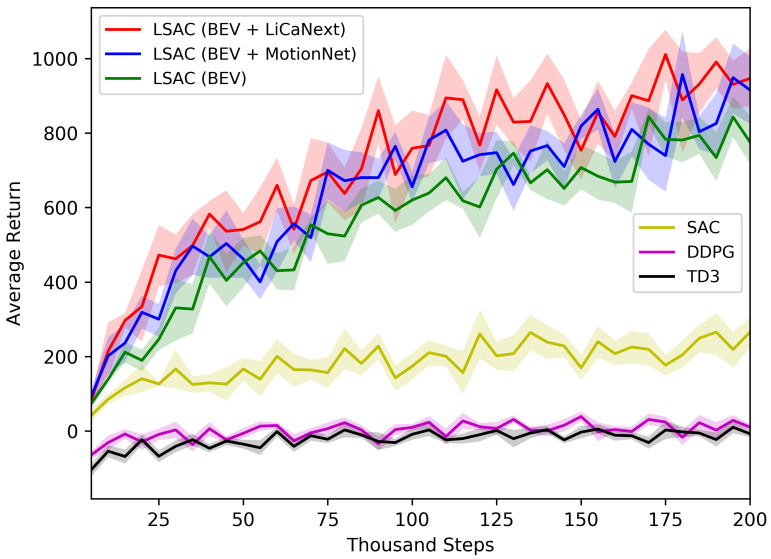


Figure 6.5 Average return evaluation of our proposed approach with baseline algorithms.

After verifying that our proposed approach secures the best learning policy in a complex environment (Town03) and under clear sunny weather (ClearNoon), further analysis is conducted on our proposed learned policy under different environments and varying weather conditions. These additional experiments examine the effect of changing the town environment and varying weather conditions on the effectiveness of our proposed learned policy. Each analysis experiment is performed for 100 evaluation episodes. The metrics measured are the success rate of an episode based on the number of completed timesteps, collision rate, rate of breaking out of lane, rate of exceeding the maximum allowed speed, the average speed, average angular velocity, average distance from lane center, average traveled distance, and average return. These metrics provide an infraction analysis indicating our agent's failure cases.

Table 6.3 reveals the outcomes of these experiments. Compared to the first experiment that is conducted under the same conditions as training (i.e., Town03 and clear weather), the other evaluation experiments procured challenging outcomes. This indicates that our proposed learned policy can generalize well to unseen scenarios even with varying weather conditions. Due to the high dynamicity of our environment setup, it is hard to identify consistent patterns between different towns and weather conditions; however, the most apparent ones are discussed. According to Figure 6.4, Town03 and Town05 have more similar layouts compared to Town04. The latter is a simple town that mainly consists of highways. Results in Table 6.3 demonstrate similar behavior

for both Town03 and Town05 with varying weather conditions; however, the outcomes in Town04 are superior.

Experiments in Town04, under all weather conditions, procured the highest success rate. This is because of the simple environment structure, where the agent merely needs to drive forward at high speed with minimal steering and slow down when a vehicle appears ahead of it. Thus, the average speed in Town04 experiments is the highest. As a result of this high speed, the speed violation rate is also highest, along with the collision rate. Even though the average speed is highest in Town04 experiments; however, it is less than half of the maximum speed limit vel_{max}. This is because the agent frequently slows down and occasionally stops due to vehicles ahead. Additionally, the agent decelerates as it approaches a turn. All these factors affect the average speed of the agent. In Town04 experiments, the angular velocity is higher because its value is associated with the agent's speed. Furthermore, as Town04 mainly contains straight lanes with minimal sharp turns, its average distance to lane center and its lane breaking rate are low. Overall, the average traveled distance and the average reward in Town04 experiments, regardless of weather, are the highest compared to other experiments.

In all experiments, the hindrance from achieving a perfect success rate is attributed to the termination causes (collisions and breaking out of lane). These situations occur in infrequent scenarios, such as when the agent fails to stop when it encounters a vehicle directly after a turn or when the agent approaches a sharp turn at high speed. Although our proposed learned policy breaks out of the lane and violates the maximum allowed speed; nevertheless, the percentage of those occurring is under 1% and 2%, respectively. Lastly, the collision rate in Town03 and Town05 experiments is lower than 8% and is less than 20% in Town04. Increasing the batch size for MaxEnt RL training should significantly reduce collisions, especially after turns, and help further diminish the rate of breaking out of lanes along with the speed violation rates.

6.6 Summary

This chapter proposed a multi-modal fusion network named *LiCaNext* to extract accurate perception and motion prediction. These predictions are leveraged as input with another simple BEV

Table 6.3 Further analysis on our proposed learned policy under different environments and varying weather conditions.

Metric	Town03			Town04			Town05		
	Clear	**Cloudy**	**Rainy**	**Clear**	**Cloudy**	**Rainy**	**Clear**	**Cloudy**	**Rainy**
Success Rate (%)	79.1	74.3	77.6	85.6	**89.9**	85.1	76.7	69.4	69.9
Collision Rate (%)	5.35	**2.04**	3.40	19.12	12.29	16.09	7.92	5.85	6.12
Lane Breaking Rate (%)	0.11	0.13	0.09	0.09	**0.08**	0.12	0.10	0.13	0.10
Speed Violation Rate (%)	**0.47**	0.55	1.24	1.75	1.49	1.15	1.33	0.61	1.60
Avg. Speed (m/s)	4.67	4.41	4.66	**6.14**	5.69	5.67	4.97	4.61	5.07
Avg. Angular Velocity (deg/s)	9.67	9.36	**9.27**	11.36	10.67	10.60	10.06	9.43	10.24
Avg. Lane Center (m)	0.39	**0.34**	0.34	0.36	0.37	0.37	0.40	0.37	0.36
Avg. Distance Traveled (m)	178	148	162	**261**	258	250	181	145	164
Avg. Return	1155	967	1068	**1921**	1882	1847	1201	953	1060

image to train a sequential latent MaxEnt RL algorithm for urban autonomous driving. The sequential latent environmental model and the MaxEnt RL model are learned end-to-end. The environmental model learns the low-dimensional latent space directly from inputs, and then MaxEnt RL trains on this latent space to generate a driving policy. The learned driving policy depends only on one LIDAR sensor and achieves the objectives of safety, efficiency, and comfort in an urban environment. Experiments were simulated on CARLA and compared the performance of our proposed approach to state-of-the-art algorithms. Results showed that our proposed approach outperformed the baselines algorithms. Further analysis was conducted demonstrating the effectiveness of our proposed approach under different environments and varying weather conditions.

CHAPTER 7
CONCLUSION AND FUTURE WORK

7.1 Concluding Remarks

This Ph.D. thesis presented a novel algorithm for learning a driving policy that achieves the objectives of safety, efficiency, and comfort in an urban environment. This is accomplished by proposing the exploitation of multi-modal fusion for generating accurate pixel-wise perception and motion prediction in real-time. These predictions are later used to train a sequential latent maximum entropy reinforcement learning (MaxEnt RL). Each development stage of our proposed algorithm is explained in a chapter. Furthermore, each chapter included a brief introduction, a thorough review, an extensive explanation of the proposed solutions, and experimental results on an autonomous driving dataset were discussed, and the last chapter includes experiments that were simulated on an autonomous driving simulator.

Chapter 3 considered an end-to-end fusion of LIDAR-based bird's-eye view (BEV) and range view (RV) features to generate accurate pixel-wise perception and motion prediction in real-time. A LIDAR sensor was used to generate both input representations. Experiments were evaluated on nuScenes dataset to verify that our proposed approach, which used multi-view feature fusion, outperformed the baseline MotionNet and other state-of-the-art methods. Performance advancement was registered for both perception and motion prediction, especially for small and distant objects.

Chapter 4 proposed a novel LIDAR Camera Network (LiCaNet) that achieved accurate pixel-wise joint perception and motion prediction in real-time. LiCaNet expanded on our earlier fusion network by incorporating a camera image into the fusion of LIDAR sourced sequential BEV and RV images. A comprehensive evaluation using nuScenes dataset was performed to validate the outstanding performance of LiCaNet compared to its predecessor. Experiments revealed that utilizing a camera sensor resulted in a substantial gain in perception and motion prediction. Moreover, most of the improvements achieved fall within the camera range, with the highest registered for small and distant objects, confirming the significance of incorporating a camera sensor into a fusion

network.

Chapter 5 explained the effect of incorporating sequential range residual images into the fusion of LIDAR-based BEV and RV images. In addition, the use of a camera sensor in the fusion process was evaluated. This proposed network is the *next* version of LiCaNet and is named *LiCaNext*. The addition of range residual images into the fusion process leads to accuracy advancement in joint perception and motion prediction while maintaining real-time requirements. Experiments conducted on the public nuScenes dataset demonstrated that incorporating sequential range residuals secured significant performance gain, with monotonic progress for a larger number of exploited residuals.

Chapter 6 considered exploiting multi-modal feature fusion to generate accurate joint perception and motion prediction, which are then used to train a sequential latent MaxEnt RL algorithm to learn a driving policy capable of achieving the objectives of safety, efficiency, and comfort. The novel *LiCaNext* was used to generate accurate pixel-wise perception and motion prediction in real-time from the multi-modal feature fusion that encompassed BEV, RV, and residual images. CARLA simulator was used to conduct experiments. The performance of the learned policy was compared to state-of-the-art baselines. Results showed that our proposed approach outperformed the prevalent baselines and maintained challenging performance under different environments and weather conditions.

In conclusion, we recommend exploiting multi-modal fusion in autonomous driving to enhance the accuracy of perception and motion prediction. Specifically, the fusion of camera images with LIDAR-based BEV, RV, and range residual images. Furthermore, to leverage real-time pixel-wise predictions extracted from a multi-modal fusion network in developing better driving policies. These policies are to be learned from a DRL algorithm that separates representation and task learning and has excellent exploration capability.

7.2 Potential Directions for Future Work

The research works presented in this Ph.D. thesis have opened new opportunities for further investigation in the following potential directions.

- The spherical projection used to engender RV images from LIDAR sweeps results in two noise problems: empty pixels and covered points. When spherical projection is used to project 3D LIDAR points into 2D RV image, the issue of empty pixels occurs when no point gets projected into a pixel. On the other hand, covered points occur when multiple points get projected into one pixel. In that case, only one point is selected to fill that pixel. Typically, the nearest point to the LIDAR sensor is selected. Such noise problems can limit the performance of the perception and motion prediction network. An obvious solution would be to increase the RV dimensions. This would reduce the effect of covered points but can also increase the number of empty pixels. Additionally, the length dimension of the RV image is not flexible and is limited to the number of laser beams of the LIDAR. Another critical problem with generating finer images is the increase in processing complexity. A potential direction towards addressing the limitation of empty pixels would be applying filters, while k-nearest neighbor (kNN) can resolve the covered points issue [145, 175]. These issues can appear in any approach that projects 3D points into 2D image representations.
- This Ph.D. thesis and other recent works confirmed the great potential of deep reinforcement learning (DRL) in complex decision-learning problems such as urban autonomous driving. On the other hand, supervised learning has also shown excellent capability in learning from data. Imitation learning (IL) is one supervised learning approach that learns to map observations to control commands from expert driving demonstrations. Although IL has drawbacks that include 1) the need for a large dataset composed of expert driving demonstrations to train on; 2) the achieved performance is limited to the expert's level; 3) the dataset does not cover extreme scenarios. A hybrid framework integrating IL with DRL can leverage both human experience and machine intelligence [176]. Human expert demonstrations may have significant advantages in learning efficiency, decision-making performance, and enhancing the adaptability of DRL models in complex urban environments. Thus, a potential direction is to embed IL with DRL to advance the performance of the learned driving policy. The resulting agent will perform better than agents trained on IL only and exhibits more human-like behaviors than agents trained on DRL and IL.
- This Ph.D. thesis confirmed the power of exploiting multi-modal fusion to generate perception and motion prediction, then using these predictions to train sequential latent MaxEnt

RL algorithm to attain the objectives of safety, efficiency, and comfort; however, interactions between vehicles were not considered. Connected autonomous vehicles (CAVs) communicate to share essential information enabling them to coordinate their behavior. Recently, researchers investigated CAVs in solving several autonomous driving applications including car-following [177], intersection navigation [178], and roundabout navigation [179]. Indeed, the existence of vehicular communication plays an important role in enhancing road safety, traffic efficiency, and reducing fuel consumption [180]. For example, CAVs can share their velocities and intentions; and the other agents can adjust their velocities accordingly to avoid congestion or even adapt their path-planning. Therefore, a potential direction to further boost safety, traffic efficiency, and comfort, in addition to reducing energy consumption, is to involve vehicular communication between DRL-controlled agents.

REFERENCES

[1] "Waypoint - the official waymo blog: Introducing the 5th-generation waymo driver: Informed by experience, designed for scale, engineered to tackle more environments," Mar 2020. [Online]. Available: https://blog.waymo.com/2020/03/introducing-5th-generation-waymo-driver.html

[2] "How autonomous vehicles sensors fusion helps avoid deaths: Intellias blog," Nov 2020. [Online]. Available: https://www.intellias.com/sensor-fusion-autonomous-cars-helps-avoid-deaths-road/

[3] M. Buehler, K. Iagnemma, and S. Singh, *The DARPA urban challenge: autonomous vehicles in city traffic*. Springer, 2009, vol. 56.

[4] M. Buehler, K. Iagnemma, and S. Singh, *The 2005 DARPA grand challenge: the great robot race*. Springer, 2007, vol. 36.

[5] T. Litman, "Autonomous vehicle implementation predictions: Implications for transport planning," 2020.

[6] M. Maurer, J. C. Gerdes, B. Lenz, and H. Winner, *Autonomous driving: technical, legal and social aspects*. Springer Nature, 2016.

[7] "Sae international releases updated visual chart for its "levels of driving automation" standard for self-driving vehicles," Dec 2018. [Online]. Available: https://www.sae.org/news/press-room/2018/12/sae-international-releases-updated-visual-chart-for-its-%E2%80%9Clevels-of-driving-automation%E2%80%9D-standard-for-self-driving-vehicles

[8] M. Channon, L. McCormick, and K. Noussia, *The law and autonomous vehicles*. Taylor & Francis, 2019.

[9] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58 443–58 469, 2020.

[10] H. Zhu, K.-V. Yuen, L. Mihaylova, and H. Leung, "Overview of environment perception for intelligent vehicles," vol. 18, no. 10, pp. 2584–2601, 2017.

[11] S. D. Pendleton, H. Andersen, X. Du, X. Shen, M. Meghjani, Y. H. Eng, D. Rus, and M. H. Ang, "Perception, planning, control, and coordination for autonomous vehicles," *Machines*, vol. 5, no. 1, p. 6, 2017.

[12] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annual Review of Control, Robotics, and Autonomous Systems*, 2018.

[13] L. Chen, Y. He, Q. Wang, W. Pan, and Z. Ming, "Joint optimization of sensing, decision-making and motion-controlling for autonomous vehicles: A deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, 2022, (Early Access - 14 February 2022).

[14] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, vol. 37, no. 3, pp. 362–386, 2020.

[15] A. Tampuu, T. Matiisen, M. Semikin, D. Fishman, and N. Muhammad, "A survey of end-to-end driving: Architectures and training methods," *IEEE Trans. Neural Netw., Learning Syst.*, 2020.

[16] M. Morales, *Grokking deep reinforcement learning*. Manning Publications, 2020.

[17] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274*, 2017.

[18] "Road traffic injuries," 2021. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries

[19] M. A. Geistfeld, "A roadmap for autonomous vehicles: State tort liability, automobile insurance, and federal safety regulation," *Calif. L. Rev.*, vol. 105, p. 1611, 2017.

[20] Ronan, "Road safety annual report 2021: The impact of covid-19," 2021. [Online]. Available: https://www.itf-oecd.org/road-safety-annual-report-2021

[21] M. M. Morando, Q. Tian, L. T. Truong, and H. L. Vu, "Studying the safety impact of autonomous vehicles using simulation-based surrogate safety measures," *Journal of advanced transportation*, vol. 2018, 2018.

[22] T. Agarwal, H. Arora, and J. Schneider, "Learning urban driving policies using deep reinforcement learning," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2021, pp. 607–614.

[23] M. Ahmed, A. Abobakr, C. P. Lim, and S. Nahavandi, "Policy-based reinforcement learning for training autonomous driving agents in urban areas with affordance learning," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[24] J. Chen, S. E. Li, and M. Tomizuka, "Interpretable end-to-end urban autonomous driving with latent deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[25] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[26] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[27] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proc. of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[28] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[29] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[30] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1587–1596.

[31] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *arXiv preprint arXiv:1801.01290*, 2018.

[32] A. X. Lee, A. Nagabandi, P. Abbeel, and S. Levine, "Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model," *arXiv preprint arXiv:1907.00953*, 2019.

[33] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[34] P. Wu, S. Chen, and D. N. Metaxas, "MotionNet: Joint perception and motion prediction for autonomous driving based on bird's eye view maps," in *Proc. IEEE CVPR*, 2020, pp. 11 385–11 395.

[35] M. Liang, B. Yang, W. Zeng, Y. Chen, R. Hu, S. Casas, and R. Urtasun, "PnPNet: End-to-end perception and prediction with tracking in the loop," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 553–11 562.

[36] W. Zeng, W. Luo, S. Suo, A. Sadat, B. Yang, S. Casas, and R. Urtasun, "End-to-end interpretable neural motion planner," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8660–8669.

[37] W. Luo, B. Yang, and R. Urtasun, "Fast and Furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 3569–3577.

[38] S. Fadadu, S. Pandey, D. Hegde, Y. Shi, F.-C. Chou, N. Djuric, and C. Vallespi-Gonzalez, "Multi-view fusion of sensor data for improved perception and prediction in autonomous driving," *arXiv preprint arXiv:2008.11901*, 2020.

[39] D. Feng, C. Haase-Schütz, L. Rosenbaum, H. Hertlein, C. Glaeser, F. Timm, W. Wiesbeck, and K. Dietmayer, "Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges," 2020.

[40] G. P. Meyer, J. Charland, D. Hegde, A. Laddha, and C. Vallespi-Gonzalez, "Sensor fusion for joint 3d object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR) Workshops*, 2019, pp. 0–0.

[41] M. Liang, B. Yang, Y. Chen, R. Hu, and R. Urtasun, "Multi-task multi-sensor fusion for 3d object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 7345–7353.

[42] H. Rashed, M. Ramzy, V. Vaquero, A. El Sallab, G. Sistu, and S. Yogamani, "FuseMODNet: Real-time camera and lidar based moving object detection for robust low-light autonomous driving," in *Proc. of the IEEE/CVF Int. Conf. on Computer Vision Workshops*, 2019, pp. 0–0.

[43] S. Kong, *Pixel-Level Prediction: Models and Applications*. University of California, Irvine, 2019.

[44] S. Zhang, Z. Wang, Q. Wang, J. Zhang, G. Wei, and X. Chu, "EDNet: Efficient disparity estimation with cost volume combination and attention-based spatial residual," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 5433–5442.

[45] "Nvidia TensorRT," 2021. [Online]. Available: https://developer.nvidia.com/tensorrt/

[46] S. Shi, X. Wang, and H. Li, "PointRCNN: 3d object proposal generation and detection from point cloud," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 770–779.

[47] S. Shi, Z. Wang, J. Shi, X. Wang, and H. Li, "From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2020.

[48] Y. Zhou and O. Tuzel, "VoxelNet: End-to-end learning for point cloud based 3d object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 4490–4499.

[49] Y. Chen, S. Liu, X. Shen, and J. Jia, "Fast point r-cnn," in *Proc. IEEE Int. Conf. on Comput. Vision (ICCV)*, 2019, pp. 9775–9784.

[50] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li, "PV-RCNN: Point-voxel feature set abstraction for 3d object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020, pp. 10 529–10 538.

[51] X. Liu, C. R. Qi, and L. J. Guibas, "FlowNet3D: Learning scene flow in 3d point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 529–537.

[52] X. Gu, Y. Wang, C. Wu, Y. J. Lee, and P. Wang, "HPLFlowNet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3254–3263.

[53] M. Schreiber, S. Hoermann, and K. Dietmayer, "Long-term occupancy grid prediction using recurrent neural networks," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 9299–9305.

[54] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 3354–3361.

[55] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2446–2454.

[56] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estima-

tion," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4040–4048.

[57] L. Fan, X. Xiong, F. Wang, N. Wang, and Z. Zhang, "RangeDet: In defense of range view for lidar-based 3d object detection," *arXiv preprint arXiv:2103.10039*, 2021.

[58] A. Bewley, P. Sun, T. Mensink, D. Anguelov, and C. Sminchisescu, "Range conditioned dilated convolutions for scale invariant 3d object detection," *arXiv preprint arXiv:2005.09927*, 2020.

[59] S. Casas, W. Luo, and R. Urtasun, "IntentNet: Learning to predict intention from raw sensor data," in *Conference on Robot Learning*. PMLR, 2018, pp. 947–956.

[60] S. Casas, C. Gulino, R. Liao, and R. Urtasun, "Spatially-aware graph neural networks for relational behavior forecasting from sensor data," *arXiv preprint arXiv:1910.08233*, 2019.

[61] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.

[62] G. P. Meyer, J. Charland, S. Pandey, A. Laddha, S. Gautam, C. Vallespi-Gonzalez, and C. Wellington, "Laserflow: Efficient and probabilistic object detection and motion forecasting," *IEEE Robotics and Automation Letters*, 2020.

[63] A. Laddha, S. Gautam, G. P. Meyer, C. Vallespi-Gonzalez, and C. K. Wellington, "RV-FuseNet: Range view based fusion of time-series lidar data for joint 3d object detection and motion forecasting," *arXiv preprint arXiv:2005.10863*, 2020.

[64] N. Djuric, H. Cui, Z. Su, S. Wu, H. Wang, F.-C. Chou, L. S. Martin, S. Feng, R. Hu, Y. Xu *et al.*, "MultiNet: Multiclass multistage multimodal motion prediction," *arXiv preprint arXiv:2006.02000*, 2020.

[65] B. Yang, W. Luo, and R. Urtasun, "PIXOR: Real-time 3d object detection from point clouds," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 7652–7660.

[66] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuScenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 621–11 631.

[67] C. Xu, B. Wu, Z. Wang, W. Zhan, P. Vajda, K. Keutzer, and M. Tomizuka, "SqueezeSegV3: Spatially-adaptive convolution for efficient point-cloud segmentation," in *European Conf. on Comput. Vision*. Springer, 2020, pp. 1–19.

[68] T. Cortinhal, G. Tzelepis, and E. E. Aksoy, "SalsaNext: Fast, uncertainty-aware semantic segmentation of lidar point clouds," in *Int. Symposium on Visual Computing*. Springer, 2020, pp. 207–222.

[69] V. E. Liong, T. N. T. Nguyen, S. Widjaja, D. Sharma, and Z. J. Chong, "AMVNet: Assertion-based multi-view fusion network for lidar semantic segmentation," *arXiv preprint arXiv:2012.04934*, 2020.

[70] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[71] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *European Conf. on Computer Vision*. Springer, 2016, pp. 21–37.

[72] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2016.

[73] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," *arXiv preprint arXiv:1605.06409*, 2016.

[74] M. Carranza-García, J. Torres-Mateo, P. Lara-Benítez, and J. García-Gutiérrez, "On the performance of one-stage and two-stage object detectors in autonomous vehicles using camera data," *Remote Sensing*, vol. 13, no. 1, p. 89, 2021.

[75] Y. Cai, T. Luan, H. Gao, H. Wang, L. Chen, Y. Li, M. A. Sotelo, and Z. Li, "YOLOv4-5D: An effective and efficient object detector for autonomous driving," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–13, 2021.

[76] E. Hassan, Y. Khalil, and I. Ahmad, "Learning feature fusion in deep learning-based object detector," *Journal of Engineering*, vol. 2020, 2020.

[77] X. Hu, X. Xu, Y. Xiao, H. Chen, S. He, J. Qin, and P.-A. Heng, "SINet: A scale-insensitive convolutional neural network for fast vehicle detection," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 3, pp. 1010–1019, 2018.

[78] D. Guo, L. Zhu, Y. Lu, H. Yu, and S. Wang, "Small object sensitive segmentation of urban street scene with spatial adjacency between object classes," *IEEE Trans. Image Process.*, vol. 28, no. 6, pp. 2643–2653, 2018.

[79] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 8759–8768.

[80] C. Ma, Y. Guo, Y. Lei, and W. An, "Binary volumetric convolutional neural networks for 3-d object recognition," *IEEE Trans. Instrum. Meas.*, vol. 68, no. 1, pp. 38–48, 2018.

[81] L. Porzi, S. R. Bulo, A. Colovic, and P. Kontschieder, "Seamless scene segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 8277–8286.

[82] Y. Xiong, R. Liao, H. Zhao, R. Hu, M. Bai, E. Yumer, and R. Urtasun, "UPSNet: A unified panoptic segmentation network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8818–8826.

[83] D. Yang, X. Zhong, D. Gu, X. Peng, and H. Hu, "Unsupervised framework for depth estimation and camera motion prediction from video," *Neurocomputing*, vol. 385, pp. 169–185, 2020.

[84] Y. Cui, R. Chen, W. Chu, L. Chen, D. Tian, Y. Li, and D. Cao, "Deep learning for image and point cloud fusion in autonomous driving: A review," *IEEE Trans. Intell. Transp. Syst.*, 2021.

[85] M. Liang, B. Yang, S. Wang, and R. Urtasun, "Deep continuous fusion for multi-sensor 3d object detection," in *Comput. Vis. – ECCV 2018*. Cham: Springer International Publishing, 2018, pp. 663–678.

[86] S. Vora, A. H. Lang, B. Helou, and O. Beijbom, "PointPainting: Sequential fusion for 3d object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020, pp. 4604–4612.

[87] V. A. Sindagi, Y. Zhou, and O. Tuzel, "MVX-Net: Multimodal voxelnet for 3d object detection," in *2019 Int. Conf. on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 7276–7282.

[88] A. Laddha, S. Gautam, S. Palombo, S. Pandey, and C. Vallespi-Gonzalez, "MVFuseNet: Improving end-to-end object detection and motion forecasting through multi-view fusion of lidar data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2865–2874.

[89] R. S. Sutton, A. G. Barto *et al.*, *Introduction to reinforcement learning*. MIT press Cambridge, 1998, vol. 135.

[90] G. Bonaccorso, *Machine learning algorithms*. Packt Publishing Ltd, 2017.

[91] C. J. C. H. Watkins, "Learning from delayed rewards," 1989.

[92] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[93] X. Chen, S. Li, B. Mersch, L. Wiesmann, J. Gall, J. Behley, and C. Stachniss, "Moving object segmentation in 3d lidar data: A learning-based approach exploiting sequential data," *arXiv preprint arXiv:2105.08971*, 2021.

[94] H. Robbins and S. Monro, "A stochastic approximation method," *The annals of mathematical statistics*, pp. 400–407, 1951.

[95] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2016, pp. 1995–2003.

[96] M. Fortunato, M. G. Azar, B. Piot, J. Menick, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin *et al.*, "Noisy networks for exploration," *arXiv preprint arXiv:1706.10295*, 2017.

[97] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[98] M. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable mdps," *arXiv preprint arXiv:1507.06527*, 2015.

[99] S. Latif, H. Cuayáhuitl, F. Pervez, F. Shamshad, H. S. Ali, and E. Cambria, "A survey on deep reinforcement learning for audio-based applications," *arXiv preprint arXiv:2101.00240*, 2021.

[100] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications," *IEEE Trans. on Cybern.*, 2020.

[101] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Computing Surveys (CSUR)*, vol. 52, no. 1, pp. 1–38, 2019.

[102] H. T. Mouftah, M. Erol-Kantarci, and S. Sorour, *Connected and Autonomous Vehicles in Smart Cities*. CRC Press, 2020.

[103] B. Luo, D. Liu, and H.-N. Wu, "Adaptive constrained optimal control design for data-based nonlinear discrete-time systems with critic-only structure," *IEEE Trans. Neural Netw., Learning Syst.*, vol. 29, no. 6, pp. 2099–2111, 2017.

[104] L. Le Mero, D. Yi, M. Dianati, and A. Mouzakitis, "A survey on imitation learning techniques for end-to-end autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, 2022, (Early Access - 01 February 2022).

[105] A. O. Ly and M. Akhloufi, "Learning to drive by imitation: An overview of deep behavior cloning methods," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 2, pp. 195–209, 2020.

[106] F. Codevilla, E. Santana, A. M. López, and A. Gaidon, "Exploring the limitations of behavior cloning for autonomous driving," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9329–9338.

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

[108] A. Prakash, A. Behl, E. Ohn-Bar, K. Chitta, and A. Geiger, "Exploring data aggregation in policy learning for vision-based urban autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 763–11 773.

[109] S. Hecker, D. Dai, and L. Van Gool, "End-to-end learning of driving models with surround-view cameras and route planners," in *Proceedings of the european conference on computer vision (eccv)*, 2018, pp. 435–453.

[110] S. Hecker, D. Dai, A. Liniger, M. Hahner, and L. Van Gool, "Learning accurate and human-like driving using semantic maps and attention," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 2346–2353.

[111] J. Dong, S. Chen, P. Y. J. Ha, Y. Li, and S. Labi, "A DRL-based multiagent cooperative control framework for CAV networks: a graphic convolution q network," *arXiv preprint arXiv:2010.05437*, 2020.

[112] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[113] P. Y. J. Ha, S. Chen, J. Dong, R. Du, Y. Li, and S. Labi, "Leveraging the capabilities of connected and autonomous vehicles and multi-agent reinforcement learning to mitigate highway bottleneck congestion," *arXiv preprint arXiv:2010.05436*, 2020.

[114] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of sumo-simulation of urban mobility," *International journal on advances in systems and measurements*, vol. 5, no. 3&4, 2012.

[115] M. Huegle, G. Kalweit, B. Mirchevska, M. Werling, and J. Boedecker, "Dynamic input for deep reinforcement learning in autonomous driving," *arXiv preprint arXiv:1907.10994*, 2019.

[116] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "End-to-end deep reinforcement learning for lane keeping assist," *arXiv preprint arXiv:1612.04340*, 2016.

[117] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah, "Learning to drive in a day," in *IEEE International Conference on Robotics and Automation*. IEEE, 2019, pp. 8248–8254.

[118] A. Alizadeh, M. Moghadam, Y. Bicer, N. K. Ure, U. Yavas, and C. Kurtulus, "Automated lane change decision making using deep reinforcement learning in dynamic and uncertain highway environment," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 1399–1404.

[119] A. Kesting, M. Treiber, and D. Helbing, "General lane-changing model mobil for car-following models," *Transportation Research Record*, vol. 1999, no. 1, pp. 86–94, 2007.

[120] F. Ye, X. Cheng, P. Wang, C.-Y. Chan, and J. Zhang, "Automated lane change strategy using proximal policy optimization-based deep reinforcement learning," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 1746–1752.

[121] Y. Chen, C. Dong, P. Palanisamy, P. Mudalige, K. Muelling, and J. M. Dolan, "Attention-based hierarchical deep reinforcement learning for lane change behaviors in autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[122] A. Folkers, M. Rick, and C. Büskens, "Controlling an autonomous vehicle with deep reinforcement learning," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 2025–2031.

[123] P. Zhang, L. Xiong, Z. Yu, P. Fang, S. Yan, J. Yao, and Y. Zhou, "Reinforcement learning-based end-to-end parking for automatic parking system," *Sensors*, vol. 19, no. 18, p. 3996, 2019.

[124] D. Quang Tran and S.-H. Bae, "Proximal policy optimization through a deep reinforcement learning framework for multiple autonomous vehicles at a non-signalized intersection," *Applied Sciences*, vol. 10, no. 16, p. 5722, 2020.

[125] L. Chen, X. Hu, B. Tang, and Y. Cheng, "Conditional dqn-based motion planning with fuzzy logic for autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[126] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

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

[128] G. Li, S. Li, S. Li, Y. Qin, D. Cao, X. Qu, and B. Cheng, "Deep reinforcement learning enabled decision-making for autonomous driving at intersections," *Automotive Innovation*, vol. 3, no. 4, pp. 374–385, 2020.

[129] D. Isele, R. Rahimi, A. Cosgun, K. Subramanian, and K. Fujimura, "Navigating occluded intersections with autonomous vehicles using deep reinforcement learning," in *IEEE International Conference on Robotics and Automation*. IEEE, 2018, pp. 2034–2039.

[130] C. Li and K. Czarnecki, "Urban driving with multi-objective deep reinforcement learning," 2019.

[131] Z. Qiao, K. Muelling, J. M. Dolan, P. Palanisamy, and P. Mudalige, "Automatically generated curriculum based reinforcement learning for autonomous vehicles in urban environment," in *IEEE Intelligent Vehicles Symposium*. IEEE, 2018, pp. 1233–1238.

[132] M. Bouton, A. Nakhaei, K. Fujimura, and M. J. Kochenderfer, "Safe reinforcement learning with scene decomposition for navigating complex urban environments," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 1469–1476.

[133] K. Mokhtari and A. R. Wagner, "Safe deep q-network for autonomous vehicles at unsignalized intersection," *arXiv preprint arXiv:2106.04561*, 2021.

[134] D. M. Saxena, S. Bae, A. Nakhaei, K. Fujimura, and M. Likhachev, "Driving in dense traffic with model-free reinforcement learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 5385–5392.

[135] M. Bouton, A. Nakhaei, D. Isele, K. Fujimura, and M. J. Kochenderfer, "Reinforcement learning with iterative reasoning for merging in dense traffic," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2020, pp. 1–6.

[136] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM review*, vol. 59, no. 1, pp. 65–98, 2017.

[137] M. Bouton, A. Nakhaei, K. Fujimura, and M. J. Kochenderfer, "Cooperation-aware reinforcement learning for merging in dense traffic," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 3441–3447.

[138] L. Szoke, S. Aradi, T. Becsi, and P. Gaspar, "Vehicle control in highway traffic by using reinforcement learning and microscopic traffic simulation," in *2020 IEEE 18th International Symposium on Intelligent Systems and Informatics (SISY)*. IEEE, 2020, pp. 21–26.

[139] B. Balaji, S. Mallya, S. Genc, S. Gupta, L. Dirac, V. Khare, G. Roy, T. Sun, Y. Tao, B. Townsend, E. Calleja, S. Muralidhara, and D. Karuppasamy, "DeepRacer: Autonomous racing platform for experimentation with sim2real reinforcement learning," in *IEEE International Conference on Robotics and Automation*. IEEE, 2020, pp. 2746–2754.

[140] Z. Ruiming, L. Chengju, and C. Qijun, "End-to-end control of kart agent with deep reinforcement learning," in *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2018, pp. 1688–1693.

[141] H. Ho, V. Ramesh, and E. T. Montano, "NeuralKart: A real-time mario kart 64 ai," 2017.

[142] J. Chen, B. Yuan, and M. Tomizuka, "Model-free deep reinforcement learning for urban autonomous driving," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 2765–2771.

[143] X. Qi, Y. Luo, G. Wu, K. Boriboonsomsin, and M. Barth, "Deep reinforcement learning enabled self-learning control for energy efficient driving," *Transportation Research Part C: Emerging Technologies*, vol. 99, pp. 67–81, 2019.

[144] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 1907–1915.

[145] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "RangeNet++: Fast and accurate lidar semantic segmentation," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4213–4220.

[146] Y. H. Khalil and H. T. Mouftah, "Integration of motion prediction with end-to-end latent RL for Self-Driving vehicles," in *Proc. 17th Int. Wireless Comm. Mobile Comp. Conf. (IWCMC) Veh. Comm. Symposium*. IEEE, 2021, pp. 1111–1116.

[147] J. Phillips, J. Martinez, I. A. Bârsan, S. Casas, A. Sadat, and R. Urtasun, "Deep multi-task learning for joint localization, perception, and prediction," *arXiv preprint arXiv:2101.06720*, 2021.

[148] Y. H. Khalil and H. T. Mouftah, "Lidnet: Boosting perception and motion prediction from a sequence of lidar point clouds for autonomous driving," in *2022 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2022, under review.

[149] A. Mohta, F.-C. Chou, B. C. Becker, C. Vallespi-Gonzalez, and N. Djuric, "Investigating the effect of sensor modalities in multi-sensor detection-prediction models," *arXiv preprint arXiv:2101.03279*, 2021.

[150] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[151] J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman *et al.*, "A perception-driven autonomous urban vehicle," *Journal of Field Robotics*, vol. 25, no. 10, pp. 727–774, 2008.

[152] Y. H. Khalil and H. T. Mouftah, "End-to-end multi-view fusion for enhanced perception and motion prediction," in *2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall)*. IEEE, 2021, pp. 1–6.

[153] Y. H. Khalil and H. T. Mouftah, "Licanet: Further enhancement of joint perception and motion prediction based on multi-modal fusion," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 3, pp. 222–235, 2022.

[154] G. Saur, W. Krüger, and A. Schumann, "Extended image differencing for change detection in uav video mosaics," in *Video Surveillance and Transportation Imaging Applications 2014*, vol. 9026. International Society for Optics and Photonics, 2014, p. 90260L.

[155] K. Sehairi, F. Chouireb, and J. Meunier, "Comparative study of motion detection methods for video surveillance systems," *Journal of Electronic Imaging*, vol. 26, no. 2, p. 023025, 2017.

[156] L. S. P. Annabel and K. Sekaran, "Automatic signal clearance system using density based traffic control," in *2021 5th International Conference on Trends in Electronics and Informatics (ICOEI)*. IEEE, 2021, pp. 1630–1635.

[157] M. Ramzy, H. Rashed, A. E. Sallab, and S. Yogamani, "RST-MODNet: Real-time spatio-temporal moving object detection for autonomous driving," *arXiv preprint arXiv:1912.00438*, 2019.

[158] G. Rahmon, F. Bunyak, G. Seetharaman, and K. Palaniappan, "Motion U-Net: Multi-cue encoder-decoder network for motion segmentation," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 8125–8132.

[159] W. Zheng, K. Wang, and F.-Y. Wang, "A novel background subtraction algorithm based on parallel vision and bayesian gans," *Neurocomputing*, vol. 394, pp. 178–200, 2020.

[160] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3061–3070.

[161] J. Vertens, A. Valada, and W. Burgard, "SMSnet: Semantic motion segmentation using deep convolutional neural networks," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 582–589.

[162] M. Siam, H. Mahgoub, M. Zahran, S. Yogamani, M. Jagersand, and A. El-Sallab, "MODNet: Motion and appearance based moving object detection network for autonomous driving," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 2859–2864.

[163] A. Dewan, G. L. Oliveira, and W. Burgard, "Deep semantic classification for 3d lidar data," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 3544–3549.

[164] E. Mohamed, M. Ewaisha, M. Siam, H. Rashed, S. Yogamani, W. Hamdy, M. Helmi, and A. El-Sallab, "Monocular instance motion segmentation for autonomous driving: Kitti instancemotseg dataset and multi-task baseline," *arXiv preprint arXiv:2008.07008*, 2020.

[165] H. Rashed, M. Essam, M. Mohamed, A. E. Sallab, and S. Yogamani, "BEV-MODNet: Monocular camera based bird's eye view moving object detection for autonomous driving," *arXiv preprint arXiv:2107.04937*, 2021.

[166] Z. Zhang, J. Gao, J. Mao, Y. Liu, D. Anguelov, and C. Li, "STINet: Spatio-temporal-interactive network for pedestrian detection and trajectory prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 346–11 355.

[167] Y. H. Khalil and H. T. Mouftah, "LiCaNext: Incorporating sequential range residuals for additional advancement in joint perception and motion prediction," *IEEE Access*, vol. 9, pp. 146 244–146 255, 2021.

[168] K. Shao, Z. Tang, Y. Zhu, N. Li, and D. Zhao, "A survey of deep reinforcement learning in video games," *arXiv preprint arXiv:1912.10944*, 2019.

[169] C. Yu, J. Liu, S. Nemati, and G. Yin, "Reinforcement learning in healthcare: A survey," *ACM Computing Surveys (CSUR)*, vol. 55, no. 1, pp. 1–36, 2021.

[170] N. Le, V. S. Rathour, K. Yamazaki, K. Luu, and M. Savvides, "Deep reinforcement learning in computer vision: a comprehensive survey," *Artificial Intelligence Review*, pp. 1–87, 2021.

[171] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: a survey," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2020, pp. 737–744.

[172] Y. H. Khalil and H. T. Mouftah, "Exploiting multi-modal fusion for urban autonomous driving using latent deep reinforcement learning," *IEEE Trans. on Vehicular Technology*, 2022, under review.

[173] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, "Learning latent dynamics for planning from pixels," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2555–2565.

[174] H. Wang, H. Gao, S. Yuan, H. Zhao, K. Wang, X. Wang, K. Li, and D. Li, "Interpretable decision-making for autonomous vehicles at highway on-ramps with latent space reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 9, pp. 8707–8719, 2021.

[175] D. Cheng, D. Zhao, J. Zhang, C. Wei, and D. Tian, "Pca-based denoising algorithm for outdoor lidar point cloud data," *Sensors*, vol. 21, no. 11, p. 3703, 2021.

[176] Z. Huang, J. Wu, and C. Lv, "Efficient deep reinforcement learning with imitative expert priors for autonomous driving," *IEEE Transactions on Neural Networks and Learning Systems*, 2022, (Early Access - 26 January 2022).

[177] S. Wei, Y. Zou, X. Zhang, T. Zhang, and X. Li, "An integrated longitudinal and lateral vehicle following control system with radar and vehicle-to-vehicle communication," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1116–1127, 2019.

[178] B. Peng, M. F. Keskin, B. Kulcsár, and H. Wymeersch, "Connected autonomous vehicles for improving mixed traffic efficiency in unsignalized intersections with deep reinforcement learning," *Communications in Transportation Research*, vol. 1, p. 100017, 2021.

[179] L. Zhao, A. Malikopoulos, and J. Rios-Torres, "Optimal control of connected and automated vehicles at roundabouts: An investigation in a mixed-traffic environment," *IFAC-PapersOnLine*, vol. 51, no. 9, pp. 73–78, 2018.

[180] S. E. Li, Y. Zheng, K. Li, L.-Y. Wang, and H. Zhang, "Platoon control of connected vehicles from a networked control perspective: Literature review, component modeling, and controller synthesis," *IEEE Transactions on Vehicular Technology*, 2017.