

Multimodal Spatio-Temporal Deep Learning Framework for 3D Object Detection in Instrumented Vehicles

Venkatesh Gurram Munirathnam , B.E, M.Tech

A Dissertation submitted in fulfilment of the
requirements for the award of
Doctor of Philosophy (PhD)
to the



DUBLIN CITY UNIVERSITY
School of Computing

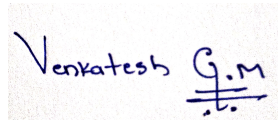
Supervised by
Dr. Suzanne Little and Prof. Noel E. O'Connor

January 2023

Declaration

I, hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy is entirely my own work, and that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed:

A handwritten signature in blue ink that reads "Venkatesh G.M." with a stylized flourish underneath.

Student No: 18213945

Date: 06/01/2023

(Venkatesh Gurram Munirathnam)

Acknowledgements

The satisfaction and euphoria accompanying the successful completion of this research work presented in this thesis would be incomplete without mentioning the people who made it possible, whose constant guidance and encouragement crowned my efforts with success. I sincerely consider privileged to express gratitude and respect towards all those who have guided me through the completion of this research.

The endeavour of my PhD would not have been possible without my supervisor, Dr Suzanne Little, for her constant support and encouragement from day one of the projects by giving her valuable time and insight into the field of research work for the execution of the project. Her advice at many stages in this research benefited me, particularly when exploring new ideas. Her positive outlook and confidence in my research inspired me and gave me the confidence to tackle the tough phases during my PhD journey. Her careful editing contributed enormously to the production of this thesis. I express my heartfelt gratitude to my supervisor, Prof. Noel E O'Connor for giving me an excellent opportunity to work under his supervision, supporting my research directions and encouraging me throughout my PhD, and being my true inspiration. I would like to thank my supervisor Dr Kevin McGuinness for his insightful feedback and sound advice during the initial phase of my PhD.

I would like to take this opportunity to express my pleasure to be a part of Insight SFI Research Centre for Data Analytics, DCU, for funding my research and providing complete freedom and the required facilities. I gained considerable experience and knowledge while executing the projects offering a healthy environment to conduct the research work.

I would like to thank Dr Naresh Y G and Dr Jaime Fernandez for the collaborative research on the Vi-DAS project. All these years, all the on-topic and off-topic conversations were always knowledgeable and fun. I would like to express my deepest gratitude to the people involved in the Huawei-Insight Autonomous Vehicle project. Especially Prof. Alan F. Smeaton and Dr Feiyan Hu for sharing their valuable experience in the field of computer vision and feedback during the project, which is one of the driving forces in the proposal of the research presented in this thesis. I had the pleasure of collaborating with Bianca Pereira, NUI Galway, on the Citizen Science project: Crowd4Access. The experience gained in this project was priceless and an eye-opener for many of the topics where technology can be applied to make cities more innovative, inclusive and accessible to everyone.

Dedication

This thesis is dedicated to the loving memory of my parents, Parvathi and Munirathnam, whose love for me had no boundaries. They taught me to respect others in everyday life and the value of hard work towards achieving your goals. Although they were my source of inspiration and driving force to pursue my doctoral degree, they could not see my graduation. This is for them.

I dedicate this work to my wife, Keerthi, who has been a constant support and encouragement during the challenges of my post-graduate journey and life. I am truly thankful for having you in my life. This work is also dedicated to my son, Rudhvi. His strong aspirations towards science and technology in building intelligent machines for better tomorrow have been monumental for my motivation to work hard for the things I aspire to achieve.

Finally, I dedicate this thesis to my family and friends on both continents who supported me throughout the process, and I will always appreciate all they have done.

List of Publications

- **Venkatesh, G. M.**, O'Connor, N. E., & Little, S. (2022, September). Incorporating Spatio-Temporal Information in Frustum-ConvNet for Improved 3D Object Detection in Instrumented Vehicles. In 2022 10th European Workshop on Visual Information Processing (EUVIP) (pp. 1-6). IEEE.
- Hu, F., **Venkatesh, G. M.**, O'Connor, N. E., Smeaton, A. F., & Little, S. (2021, January). Utilising visual attention cues for vehicle detection and tracking. In 2020 25th International Conference on Pattern Recognition (ICPR) (pp. 5535-5542). IEEE.
- **Venkatesh, G. M.**, Hu, F., O'Connor, N. E., Smeaton, A. F., Yang, Z., & Little, S. (2019, September). Saliency guided 2d-object annotation for instrumented vehicles. In 2019 International Conference on Content-Based Multimedia Indexing (CBMI) (pp. 1-7). IEEE.
- Fernandez, J. B., **Venkatesh, G. M.**, Zhang, D., Little, S., & O'Connor, N. E. (2019, September). Semi-automatic multi-object video annotation based on tracking, prediction and semantic segmentation. In 2019 International Conference on Content-Based Multimedia Indexing (CBMI) (pp. 1-4). IEEE.

Other Publications:

- **Venkatesh, G. M.**, Pereira, B., & Little, S. (2021, October). Urban footpath image dataset to assess pedestrian mobility. In Proceedings of the 1st International Workshop on Multimedia Computing for Urban Data (pp. 23-30).
- Salwala, D., Yadav, P., **Venkatesh, G. M.**, Little, S., O'Connor, N. E., & Curry, E. (2021, October). UrbanAccess: Query Driven Urban Analytics Platform for Detecting Complex Accessibility Event Patterns using Tactile Surfaces. In Proceedings of the 1st International Workshop on Multimedia Computing for Urban Data (pp. 19-21).
- YG, N., **Venkatesh, G. M.**, O'Connor, N. E., & Little, S. (2020). Object polygonization in traffic scenes using small Eigenvalue analysis. Irish Pattern Recognition & Classification Society.
- **Venkatesh, G. M.**, Naresh, Y. G., Little, S., & O'Connor, N. E. (2018). A deep residual architecture for skin lesion segmentation. In OR 2.0 Context-Aware Operating Theaters, Computer Assisted Robotic Endoscopy, Clinical Image-Based Procedures, and Skin Image Analysis (pp. 277-284). Springer, Cham.

Contents

Abstract	1
1 Introduction	3
1.1 Motivation	7
1.2 Hypothesis and Research Question	11
1.3 Research Contribution	13
1.4 Thesis Outline	14
2 Related Work and Background	17
2.1 2D Object Detection	18
2.2 3D Object Detection	19
2.2.1 Single Frame Reference Methods	20
Lidar point cloud data	20
Lidar point cloud and Camera Image	30
2.2.2 Multi Frame Reference Methods	37
2.3 Dataset	39
2.4 Evaluation Metric	44
2.4.1 KITTI evaluation metric	45
2.4.2 nuScenes evaluation metric	46
2.5 Summary	47
3 Preliminary Work	49
3.1 2D Object Detection	50
3.1.1 Single and Two Stage Object Detector	50
3.1.2 Multiple Object Tracking-by-Detection	55
3.2 3D Object Detection	59
3.2.1 Complex-YOLOv3	60
3.2.2 Complex-YOLOv3 with convolutional-LSTM	61
3.2.3 Baseline Architecture: Frustum-ConvNet	64
3.3 Summary	68
4 Data Representation and Performance Baselines	71
4.1 Dataset	73
4.1.1 KITTI	73
4.1.2 nuScenes	74
4.2 Understanding Temporal information	80
4.3 Exploratory Trials	82
4.3.1 Influence of 2D proposal on 3D estimations	83
4.3.2 Minimum Sequence for Temporal information	88
4.3.3 Varying the lidar feature vector size	91
4.3.4 Lidar Single-Sweep Versus Multi-Sweep	93

4.3.5	Training: Individual Class vs All-together	95
4.4	Summary	97
5	Temporal Encoding	99
5.1	Incorporation of Spatio-temporal Information	100
5.1.1	Frame Stacking	102
5.1.2	Convolution-LSTM	102
5.1.3	Kernel Attention Mechanism Module	104
5.2	Experiments	108
5.2.1	Dataset	108
5.2.2	Setup	108
5.2.3	Metrics	109
5.2.4	Results and Discussion	110
5.3	Summary	113
6	Multimodal Data Fusion and Temporal Encoding	117
6.1	Fusion Strategies	119
6.1.1	Global Multimodal Fusion (GMF)	119
6.1.2	Intermediate Multimodal Fusion (IMF)	120
6.1.3	Joint Multimodal Fusion (JMF)	121
6.2	Pipeline	122
6.2.1	Multimodal 3D Object detection	122
6.2.2	Multimodal Spatio-Temporal 3D object detection	123
6.3	Experimental Setup	124
6.4	Results and Discussion	125
6.5	Summary	131
7	Summary and Conclusions	135
7.1	Research Questions and Proposed Solutions	137
7.2	Research Contributions	144
7.3	Future Research	145
	Bibliography	147

List of Figures

1.1	Illustration of Autonomous Vehicle with sensor on board adapted from [32]	3
1.2	Some of the ego Vehicles used by various companies and universities working on autonomous driving	4
1.3	An example of a Google car’s internal map at an intersection, tweeted by Idealab founder Bill Gross. Gross claims that Google’s Self-Driving Car gathers almost 1 GB of data per second.	7
1.4	Generic perception system for autonomous vehicles	8
1.5	Representation of lidar and image data for the scene-0061 in the nuScenes dataset	10
2.1	structure of literature presented in this chapter	17
2.2	Representation captured from Complex-YOLO [100], the method operates on Lidar only based birds-eye-view (BEV) RGB-maps to estimate and localize accurate 3D multiclass bounding boxes	21
2.3	(a): Schematic diagram of the voxel procedure of LiDAR data from [146], (b): Voxel Representation using height directly as input feature as in [95] .	23
2.4	An illustration of early, late, and deep fusion [61, 7]	31
2.5	Sequence of frustums generated for a region proposal in F-ConvNet architecture [85]	34
3.1	YOLO-V5 network architecture [21, 25, 12]	50
3.2	Object detection using YOLOv5, left column represents input and right column represents corresponding detection results on KITTI Dataset	51
3.3	The SSD network architecture [123]	52
3.4	Sample output on PASCAL VOC dataset	53
3.5	Detection results on KITTI dataset, 1st row:ground truth images, 2nd row: SSD detection results and 3rd row: SSD detection results with saliency guided [84]	54
3.6	The Faster-RCNN network architecture [132]	54
3.7	Examples of Multiple Object Tracking using objectiveness mask on KITTI Dataset	56
3.8	Normal Distribution (left), Multi-peak Gaussian particle distribution (right) based on object motion	58
3.9	Examples of Multiple Object Tracking using objectiveness mask on De-TRAC Dataset	59
3.10	3D Object Detection using Complex-YOLO Architecture	60
3.11	Top row: output 3d bounding box projections on RGB image and bottom row: detection results in LiDAR data with BEV representation in KITTI dataset Image 007271 (left) and 007294 (right)	62

3.12	Incorporation of temporal information into Complex-YOLO architecture for 3D object detection	62
3.13	Simplified version of Frustum ConvNet Architecture [85]	65
4.1	Illustration of lidar point cloud, Voxel [107] and PointPillar [70] data Representation for person [8]	73
4.2	Image and Lidar Data Representation of the Scene:000218 in KITTI dataset	75
4.3	Front-View and Bird-eye-View Representation of lidar point cloud data for Image 000218 of KITTI training set	75
4.4	Sample distribution in the custom subset of nuScenes dataset	77
4.5	Image and Lidar Data Representation of the Scene:0553 in nuScenes dataset	79
4.6	Cumulative representation of lidar point cloud data over time	80
4.7	Illustration of Lidar data enclosed within the ROI of the truck on Image and Lidar frame for temporal understanding. The bounding box of the object is obtained from the ground truth annotations in the nuScenes dataset.	81
4.8	Average computation time per frame taken by the detector for 3D estimations depending on the proposal confidence and IoU threshold	84
4.9	Visual Comparison of single-sweep vs multi-sweep lidar point cloud data .	94
5.1	Proposed architecture to incorporate spatio-temporal feature into F-ConvNet architecture	101
5.2	Sequence of frustums generated for a region proposal in F-ConvNet architecture [85]	101
5.3	Spatio-temporal encoding using convolutional-LSTMs	102
5.4	Kernel Attention Mechanism (KAM) module	104
5.5	Architecture to extract spatio-temporal feature using KAM module	105
5.6	Comparison of the spatio-temporal 3D object detection results with the baseline and ground truth. The top row presents the ground truth (GT) annotation, middle row presents the Baseline results and bottom row presents the results obtained using the proposed spatio-temporal architecture. The orientation of the object is presented with a side having a cross mark on the 3D bounding box and all the GT annotated are obtained from the nuScenes dataset.	114
6.1	Illustration of fusion schemas available in the literature [61, 7, 68]	118
6.2	Pyramid Scene Parsing Network (PSPNet) Architecture [118]	120
6.3	Global multimodal fusion schema of Lidar and Image data representation. c1, c2, and c3 represent the multi-layer perceptron layers (mlp) of the PointNet Module	120
6.4	Intermediate multimodal fusion of Lidar and Image data representation. c1, c2, and c3 represent the multi-layer perceptron layers (mlp) of the PointNet Module	121
6.5	Joint multimodal fusion of Lidar and Image data representation. c1, c2, c3 and c4 represent the multi-layer perceptron layers (mlp) of the PointNet Module	122
6.6	Multimodal data fusion architecture	122
6.7	Multi-modal data fusion architecture incorporating spatio-temporal using CLSTM.	123

6.8	Multi-modal data fusion architecture incorporating spatio-temporal using CLSTM and kernel attention module.	124
6.9	Computation time analysis of models employed on KITTI dataset	127
6.10	Computation time analysis of models employed on nuScenes dataset	132
7.1	Visualisation of Velodyne’s new VLS-128 LiDAR sensor data captured for an urban scene, left, has 10 times the resolution of an image from its previous captured sensor Velodyne’s HDL-64.	145

List of Tables

2.1	Comparison of 3D Object detection methods evaluated on KITTI 3D Object detection benchmark	38
2.2	Comparison of autonomous vehicle dataset	39
3.1	YOLOv2 and YOLOv5 detection accuracy on KITTI 2D object detection validation set	51
3.2	Anchors generated using k-mean clustering, RF represents receptive field .	52
3.3	Detection results of SSD and with saliency [84] on KITTI 2D object detection training set. In legend: px represents pixel	53
3.4	Faster-RCNN detection accuracy on KITTI 2D object detection validation set. Legend: 'Easy', 'Mod.' 'Hard' and 'mAP' represent easy, moderate and hard case and mean Average Precision respectively	55
3.5	Multiple Target tracking accuracy on KITTI dataset for Car and Pedestrian classes with number of particles=100, Best performance in Bold.	59
3.6	3D object detection results on KITTI validation dataset, following short notation are used to indicate the methodology: cYv3 = Complex-YOLO3, cYv3-LSTM = Complex-YOLO3 with CLSTM module, cYv3-tiny = Complex-YOLOv3 tiny version and cYv3-tiny-LSTM = Complex-YOLOv3 tiny version with CLSTM module. Legend: Rcll = Recall, Prcn = Precision, F1 = F1 Score, AP = Average Precision, mAP = mean Average Precision, Best performance in Bold.	63
3.7	Fully convolutional Network (FCN) configuration used in Frustum-ConvNet architecture [85]	67
3.8	Performance of BEV and 3D detection of Frustum-ConvNet on KITTI validation dataset. Legend: 'Easy', 'Mod.' and 'Hard' represent easy, moderate and hard cases respectively	68
4.1	Number of instances in the KITTI train and validation datasets	74
4.2	Number of instances in the subset of nuScenes dataset	76
4.3	nuScenes detection metric of F-ConvNet for all the object class appearing in all the six cameras in the custom nuScenes test dataset. Trained only for 30 epochs	77
4.4	Number of instances in each of the classes in the subset of nuScenes dataset considered for training and testing	78
4.5	nuScenes detection metric of F-ConvNet for all classes in the custom nuScenes test dataset	78
4.6	Number of Region proposals generated '2D(r)' from 2D detector by varying IoU threshold ('Th.') and Confidence Score ('Conf.') and corresponding number of estimation by 3D detector generated '3D(e)' on KITTI validation set.	84

4.7	Average computation time per frame in milliseconds (msec) on KITTI validation set by varying the 2D detector confidence value (Conf.) and IoU threshold value (Th.).	84
4.8	BEV Detection in AP (%) for ‘Car’ Class depending on the regional proposal generated from the YOLOv5 2D detector by varying the confidence (Conf.) and IoU threshold value (Th.). Best performance in bold.	85
4.9	BEV Detection in AP (%) for ‘Pedestrian’ Class depending on the regional proposal generated from the YOLOv5 2D detector by varying the confidence (Conf.) and IoU threshold value (Th.). Best performance in bold.	85
4.10	BEV Detection in AP (%) for ‘Cyclist’ depending on the regional proposal generated from the YOLOv5 2D detector by varying the confidence (Conf.) and IoU threshold value (Th.). Best performance in bold.	85
4.11	3D Detection in AP(%) for ‘Car’ depending on the regional proposal generated from the YOLOv5 2D detector by varying the confidence (Conf.) and IoU threshold value (Th.). Best performance in bold.	86
4.12	3D Detection in AP(%) for ‘Pedestrian’ depending on the regional proposal generated from the YOLOv5 2D detector by varying the confidence (Conf.) and IoU threshold value (Th.). Best performance in bold.	86
4.13	3D Detection in AP(%) for ‘Cyclist’ depending on the regional proposal generated from the YOLOv5 2D detector by varying the confidence (Conf.) and IoU threshold value (Th.). Best performance in bold.	86
4.14	BEV detection and 3D detection AP in (%) on KITTI validation dataset using the regional proposal generated from the YOLOv5 2D detector with confidence (Conf. = 0.5) and IoU threshold value (Th. = 0.95).	87
4.15	3D Detection AP (%) of F-ConvNet with convolutional-LSTM (CLSTM) and convolutional-GRU (CGRU) on KITTI validation set. Best performance in bold.	88
4.16	BEV Detection AP (%) of F-ConvNet with convolutional-LSTM (CLSTM) and convolutional-GRU (CGRU) on KITTI validation set. Best performance in bold.	89
4.17	Detail of the input configurations utilised to investigate the minimum number of sequenced frames required to learn temporal information of an object by F-ConvNet with convolution-LSTMs	90
4.18	nuScenes Detection Score and mean True Positive metrics computed for various input configurations to understand the minimum number of sequences required to positively impact the temporal information encoding. Best performance in bold.	90
4.19	True Positive metrics of various input configurations to understand the minimum number of sequences required to positively impact the temporal information encoding	91
4.20	3D Detection AP (%) on KITTI validation set. Best performance in bold.	92
4.21	BEV Detection AP (%) on KITTI validation set. Best performance in bold.	92
4.22	Performance of Frustum-ConvNet for Car and Pedestrian class on custom nuScenes test set by varying the feature vector size of lidar point cloud data. Best performance in bold.	93
4.23	Overall Performance of Frustum-ConvNet on custom nuScenes test set by varying the feature vector size of lidar point cloud data. Best performance in bold.	93

4.24	Performance of F-ConvNet for Car and Pedestrian class on custom nuScenes test set by varying the feature vector size of multi-sweep lidar point cloud data. Best performance in bold.	95
4.25	Overall Performance of Frustum-ConvNet on custom nuScenes test set by varying the feature vector size of multi-sweep lidar point cloud data. Best performance in bold.	95
4.26	3D Object Detection and BEV Detection in AP (%) when trained classes separately vs together on KITTI validation set. Best performance in bold.	96
4.27	Overall Performance of Frustum-ConvNet when trained classes separately vs together on custom nuScenes test set. Best performance in bold.	96
5.1	Fully convolutional Network (FCN) configuration used in Frustum-ConvNet-CLSTM configuration 2 architecture	103
5.2	True positive metric scores for Cars and Pedestrians with multi-frames reference using CLSTM on the subset of nuScenes data. Best performance in bold.	104
5.3	nuScenes metric computed for CLSTM configurations with multi-frames reference on subset nuScenes data. Best performance in bold.	104
5.4	Performance of the Car and Pedestrians class using multi-frames with Kernel Attention Mechanism (KAM) on subset of nuScenes data. Best performance in bold.	107
5.5	Performance of the KAM configurations using multi-frames reference on subset nuScenes data. Best performance in bold.	107
5.6	True positive metric scores for cars and pedestrians on single frame ('s') and multi-frames ('m') reference using frame stacking (FS), convolutional-LSTMs (CLSTM) and Kernel Attention Mechanism (KAM) with the Baseline Frustum-ConvNet method on subset of nuScenes data.	111
5.7	nuScenes metric computed on single frame ('s') and multi-frames ('m') references using frame stacking (FS), convolutional-LSTMs (CLSTM) and Kernel Attention Mechanism (KAM) with the Baseline Frustum-ConvNet method on subset nuScenes data. Best performance in bold.	111
5.8	True positive metric scores for cars and pedestrians on multi-frames ('m') reference incorporating CLSTM-config-2 and Kernel Attention Mechanism (KAM) on subset of nuScenes data. Best performance in bold	112
5.9	nuScenes metric computed with multi-frames ('m') reference incorporating CLSTM-config-2 and Kernel Attention Mechanism (KAM) on subset nuScenes data. Best performance in bold.	112
5.10	True positive metric scores for cars and pedestrians with feature vector size 1,024 and 512 respectively on multi-frames ('m') reference incorporating CLSTM-config-2 and Kernel Attention Mechanism (KAM) on subset of nuScenes data. Best performance in bold	113
5.11	nuScenes metric computed with multi-frames ('m') reference for cars and pedestrians with feature vector size 1,024 and 512 respectively incorporating CLSTM-config-2 and Kernel Attention Mechanism (KAM) on subset nuScenes data. Best performance in bold.	113

6.1	Comparison of 3D Detection AP (%) results of the proposed architecture with the state-of-the-art methods, where global (gmf), intermediate (imf) and joint multimodal fusion (jmf) schema employed in the F-ConvNet (Arch) to fuse the Lidar (L) and Image (I) data. Best performance in bold.	126
6.2	Comparison of BEV Detection AP (%) results of the proposed architecture with the state-of-the-art methods on KITTI validation set, where global (gmf), intermediate (imf) and joint multimodal fusion (jmf) schema employed in the F-ConvNet (Arch) to fuse the Lidar (L) and Image (I) data. Best performance in bold.	126
6.3	True positive metric scores for cars and pedestrians on nuScenes custom test set, with global (gmf), intermediate (imf) and joint multimodal fusion (jmf) schema employed in the F-ConvNet (Arch) to fuse the Lidar (L) and Image (I) data. Best performance in bold.	128
6.4	nuScenes metric computed for cars and pedestrians on nuScenes custom test set, with global (gmf), intermediate (imf) and joint multimodal fusion (jmf) schema employed in the F-ConvNet (Arch) to fuse the Lidar (L) and Image (I) data. Best performance in bold.	129
6.5	True positive metric scores for cars and pedestrians on nuScenes custom test set, with global (gmf), intermediate (imf) and joint multimodal fusion (jmf) schema employed in the modified F-ConvNet (Arch) incorporating spatio-temporal encoding through CLSTM and KAM to to fuse the Lidar (L) and Image (I) data. Best performance in bold.	129
6.6	nuScenes metric computed for cars and pedestrians on nuScenes custom test set, with global (gmf), intermediate (imf) and joint multimodal fusion (jmf) schema employed in the modified F-ConvNet (Arch) incorporating spatio-temporal encoding through CLSTM and KAM to to fuse the Lidar (L) and Image (I) data. Best performance in bold.	130
6.7	Overview of Number of parameters and Average per frame computation time of the various models employed in this thesis work, with global (gmf), intermediate (imf) and joint multimodal fusion (jmf) schema employed in the modified F-ConvNet (Arch) incorporating spatio-temporal encoding through CLSTM and KAM to fuse the Lidar (L) and Image (I) data.	130

List of Acronyms

2D	Two Dimensional.
3D	Three Dimensional.
AAE	Average Attribute Error.
ACC	Adaptive Cruise Control.
AD	Autonomous Driving.
ADAS	Advance Driver Assistance System.
ADS	Automated Driving System.
AM	Attention Mechanism.
AOE	Average Orientation Error.
AOS	Average orientation similarity.
AP	Average Precision.
ASE	Average Scale Error.
ATE	Average Translation Error.
AV	Autonomous Vehicle.
AVE	Average Velocity Error.
BEV	Bird's-eye-View.
CLSTM	Convolutional Long Short Term Memory.
CMOS	Complementary Metal Oxide Semiconductor.
CNN	Convolutional Neural Network.
Conf.	Confidence Score.
DCNN	Deep Convolutional Neural Network.
FAR	Fragmentation.
FC	Fully Connected.
FCN	Fully Convolutional Network.
FMCW	Frequency Modulated Continuous Waves.
FOV	Field-Of-View.
FV	Feature Vector.

GB	Giga Byte.
GM-PHD	Gaussian Mixture PHD.
GMF	Global Multimodal Fusion.
GNSS	Global Navigation Satellite Systems.
GPS	Global Positioning System.
GPU	Graphic Processing Unit.
GRU	Gated Recurrent Units.
HSV	Hue Saturation and Value.
IMF	Intermediate Multimodal Fusion.
IMU	Inertial Measurement Units.
IoU	Intersection Over Union.
JMF	Joint Multimodal Fusion.
KAM	Kernel Attention Mechanism.
KG	Kalman Gain.
lidar	Light Detection and Ranging.
LSTM	Long Short Term Memory.
m/s	meters per second.
mAAE	mean Average Attribute Error.
mAOE	mean Average Orientation Error.
mAP	mean Average Precision.
mASE	mean Average Scale Error.
mATE	mean Average Translation Error.
mAVE	mean Average Velocity Error.
MEM	Microelectromechanical.
ML	Mostly Lost.
MOT	Multiple Object Tracking.
MOTA	MOT Accuracy.
MOTP	MOT Precision.
MT	Mostly Tracked.
mTP	mean True Positive.
NDS	nuScenes Detection Score.
NMS	Non-Maximal Suppression.

OPA	Optical Phase Array.
PHD	Probability Hypothesis Density.
Prcn	Precision.
PT	Partially Tracked.
radar	Radio Detection and Ranging.
RAM	Random Access Memory.
Rcll	Recall.
RCNN	Recurrent Neural Network.
RGB	Red Green Blue.
RNN	Recurrent Convolutional Neural Network.
RPN	Region Proposal Network.
SNR	Signal to Noise Ratio.
SOTA	State-Of-The-Art.
TCN	Temporal Convolution Network.
Th.	IoU Threshold.
TP	True Positive.

Abstract

Multimodal Spatio-Temporal Deep Learning Framework for 3D Object Detection in Instrumented Vehicles

Venkatesh Gurram Munirathnam

This thesis presents the utilization of multiple modalities, such as image and lidar, to incorporate spatio-temporal information from sequence data into deep learning architectures for 3D object detection in instrumented vehicles. The race to autonomy in instrumented vehicles or self-driving cars has stimulated significant research in developing autonomous driver assistance systems (ADAS) technologies related explicitly to perception systems. Object detection plays a crucial role in perception systems by providing spatial information to its subsequent modules; hence, accurate detection is a significant task supporting autonomous driving. The advent of deep learning in computer vision applications and the availability of multiple sensing modalities such as 360° imaging, lidar, and radar have led to state-of-the-art 2D and 3D object detection architectures.

Most current state-of-the-art 3D object detection frameworks consider single-frame reference. However, these methods do not utilize temporal information associated with the objects or scenes from the sequence data. Thus, the present research hypothesizes that multimodal temporal information can contribute to bridging the gap between 2D and 3D metric space by improving the accuracy of deep learning frameworks for 3D object estimations. The thesis presents understanding multimodal data representations and selecting hyper-parameters using public datasets such as KITTI and nuScenes with Frustum-ConvNet as a baseline architecture. Secondly, an attention mechanism was employed along with convolutional-LSTM to extract spatial-temporal information from sequence data to improve 3D estimations and to aid the architecture in focusing on salient lidar point cloud features. Finally, various fusion strategies are applied to fuse the modalities and temporal information into the architecture to assess its efficacy on performance and computational complexity.

Overall, this thesis has established the importance and utility of multimodal systems for refined 3D object detection and proposed a complex pipeline incorporating spatial, temporal and attention mechanisms to improve specific, and general class accuracy demonstrated on key autonomous driving data sets.

Keywords: *Autonomy, Object Detection, Autonomous Vehicles, Automated Driving, Perception System, Multiple sensing modalities, Temporal Information, Data Fusion.*

Chapter 1

Introduction

The race to autonomy among most global automotive manufacturers, including General Motors, Ford, Volkswagen, Toyota, Honda, Tesla, Volvo, and BMW, has led to the accumulation of deep knowledge about vehicle dynamics and the variety of data required to achieve full autonomy. Concurrently, breakthroughs in computer vision, the availability of new sensing modalities, such as lidar and radar, and the benefits of having improved computational hardware technology like GPUs have catalysed the research and development of autonomous vehicle (AV) technologies [126]. Fig. 1.1 presents a high-level representation of an ego vehicle with sensors on board, and Fig. 1.2 shows some of the ego vehicles employed by the various companies and universities working on autonomous driving. Typically these vehicles are equipped with multiple cameras, lidars, radars, and GPS/IMUs to collect images, 3D point clouds, and vehicle information. A brief overview of these sensors:

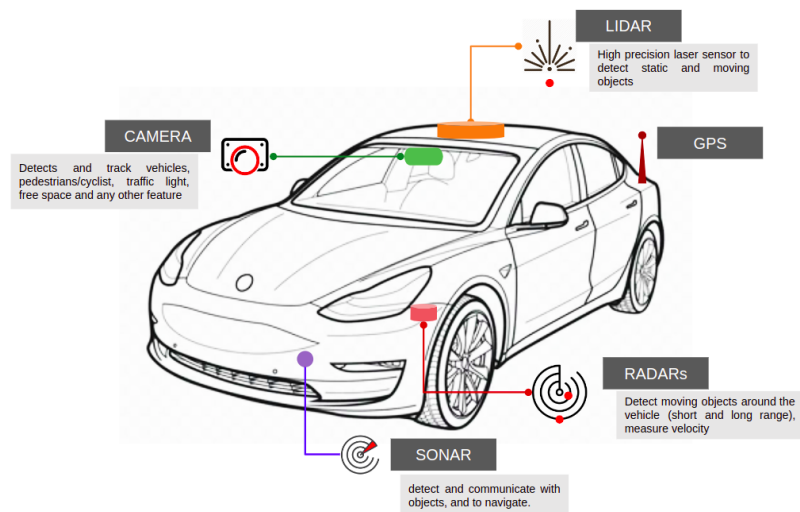


Figure 1.1: Illustration of Autonomous Vehicle with sensor on board adapted from [32]



Figure 1.2: Some of the ego Vehicles used by various companies and universities working on autonomous driving

- **Visual Cameras:** Visual cameras are the most common and widely employed sensor for data acquisition in a perception-related task in computer vision applications. The cameras are usually Complementary Metal Oxide Semiconductor (CMOS) sensors that capture the colour and texture information of a vehicle's surrounding or driving environment. However, these sensors are susceptible to lighting and weather conditions and cannot directly provide depth information. Cameras are a fundamental technology for fully autonomous navigation but can only be used to their maximum potential by fusing the data they provide with the likes of Lidar or Radar systems [89].
- **Lidar:** Light Detection And Ranging use active sensors that supply their illumination source. In automotive applications of lidar technology, the lidar sensors are installed on the top of the vehicle. Lidar sensors continuously rotate and generate thousands of laser pulses per second. These high-speed laser beams from lidar are continuously emitted in the 360-degree surroundings of the vehicle and are reflected by the objects in the way they are used to calculate accurate depth information of the surrounding objects in the form of 3D points [33]. As a result, lidar is robust to different lighting conditions and less affected by various weather conditions, such as fog and rain, than visual cameras. However, lidar sensors typically cannot capture objects' delicate

textures, and captured point clouds become sparse with distance.

The lidar sensors used in autonomous vehicles can be classified into two categories on the basis of the technology in use:

- **Electro-Mechanical Lidar:** These types of lidars are fabricated from multiple moving parts consisting of high-grade optics and a rotating assembly to create a 360-degree Field-Of-View (FOV). The mechanical aspect provides a high signal-to-noise ratio (SNR) over a wide FOV and is quite bulky, very expensive and prone to wear and tear in tough terrain. They are installed on the top of the vehicle and continuously rotate to scan the surroundings of the vehicle and typically cover a long range.
 - **Solid State Lidar:** These type of lidars are built entirely on a single chip and has no spinning mechanical components; thus, are compact in size, light in weight and cost-efficient. Since they have reduced FOV using multiple channels at the front, rear and sides of a vehicle and fusing their data creates a 360-degree FOV [41]. Solid-state lidars have multiple implementation methods, including Microelectromechanical systems (MEMs) lidar, Flash lidar, Optical phase array (OPA) and Frequency-modulated continuous wave (FMCW) lidar.
-
- **Radar:** Radar (Radio Detection And Ranging) technology in autonomous vehicles operates with millimeter waves and offers millimeter precision. The utilization of millimeter waves in autonomous vehicular radar ensures high resolution in obstacle detection and centimeter accuracy in position and movement determination and is often applied in adaptive cruise control (ACC) for road vehicles that automatically adjust the vehicle speed to maintain a safe distance from vehicles ahead. Automotive mmWave radar can be classified into three types based on the frequencies utilized: 24GHz (short-range radars), 77GHz (medium-range radars), and 79GHz (long-range radars), respectively. Impulse and Frequency modulated continuous wave radar is commonly used in AV. Radar sensors are robust against various lighting conditions and use Frequency Modulated Continuous Waves (FMCW) to detect moving or stationary targets, including cars, trains, trucks, and cargo, in extreme weather conditions. The

object's radial velocity is estimated using the Doppler effect by emitting radio waves to be reflected by an obstacle and measuring the signal run-time. Compared to lidar, radar has low accuracy and resolution making it difficult to distinguish all the objects on the road. Radar is also vulnerable to radio waves of the same wavelength the radar system uses, causing it to miss objects or consider moving objects stationary. For this reason, radar system technology is often paired with cameras and other sensor systems. For example, mmWave radar is employed along with two cameras in the automatic braking systems of automobiles

- **Other sensors:** Along with cameras, lidar and radar sensors, an AV is usually equipped with ultrasonic sensors that send out high-frequency sound waves to measure the distance to objects. They are typically applied for near-range object detection and low-speed scenarios, such as automated parking. Ultrasonic sensors are also considered to be the most accurate sensors in close-range applications. However, these sensors are seriously affected by acoustic interference as the sound can only propagate in the medium, the changes in environmental conditions such as temperature and humidity will greatly affect the performance of the sensor [89]. Global Navigation Satellite Systems (GNSS) is used to provide accurate 3D object positions. GNSS is also used together with HD Maps for path planning and ego-vehicle localization for autonomous vehicles [33]. Inertial Measurement Units (IMU) and odometers capture the vehicle's acceleration, rotational rates, and distance travelled. GNSS and IMU sensors can complement autonomous vehicle perception while generating common global reference between vehicles, identifying featureless roads and changing surroundings (highways). However, these sensors are Less reliable in high urban canyons and are not usable in long tunnels (a few km) due to the high drift rate of IMU.

The goal is to allow the AV to acquire knowledge of its surroundings, simulate human-like perception capable of comprehending the environment, precise position and path planning for the objects appearing in the complex mixed-traffic scenario. Perception of the environment is one of the crucial aspects of designing technologies related to advanced driver

assistance systems (ADAS). The perception system needs to process, understand and correlate the spatial and contextual information of the scenes and their associated objects analogous to how humans perceive their surroundings. Furthermore, object detection is essential in perception systems to provide the subsequent modules with the location, direction, object size and orientation. Hence, making accurate detection a significant task in supporting autonomous driving. The proposed research will utilise multiple sensing modalities specifically image and lidar data from KITTI [139, 138] and nuScenes [32] to extract temporal features and fuse the data in deep network architectures for improved 3D object detection.

This chapter provides an overview and motivates the research conducted in this thesis. Section 1.1 presents the motivation for deriving the hypothesis, which led to formulating a series of research questions listed in Section 1.2. Next, Section 1.3 presents the key contributions of the current research work and the layout of the thesis to investigate the hypothesis and address the research questions are presented in Section 1.4.

1.1 Motivation

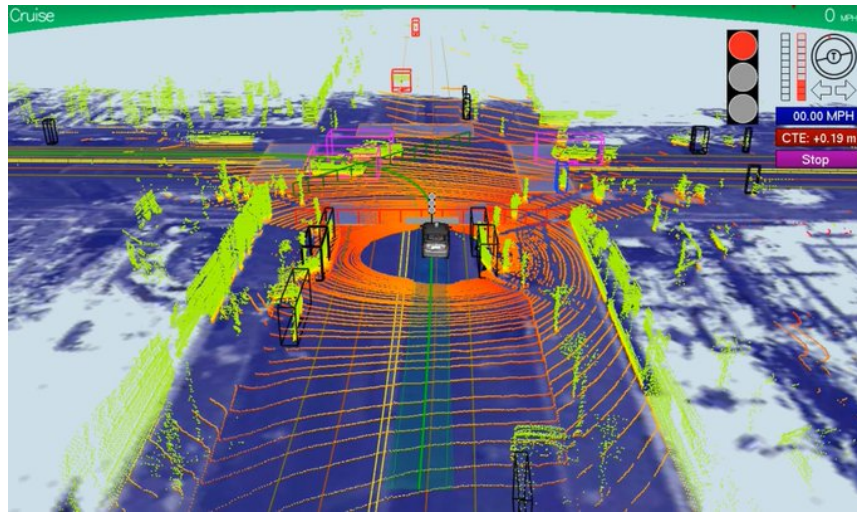


Figure 1.3: An example of a Google car's internal map at an intersection, tweeted by Idealab founder Bill Gross. Gross claims that Google's Self-Driving Car gathers almost 1 GB of data per second.

Complex driving environments, for instance, crowded city traffic scenarios presented in Fig. 1.3, often involve various types of road entities and users. Cars, cyclists, and pedestrians are highly relevant to autonomous driving. The software components of the perception

system for autonomous vehicles can be broadly categorized into three categories, namely perception, planning, and control, with the interactions between these subsystems and the vehicle's interactions with the environment depicted in Fig. 1.4. Perception refers to the ability of an autonomous system to collect information and extract relevant knowledge from the environment acquired from the various sensors mounted to the vehicle like cameras, lidar, radar, IMU and GNSS. Environmental perception refers to developing a contextual understanding of the environment, such as determining the location of obstacles, detecting road signs, and lane marking, and categorizing data by their semantic meaning. Localization refers to the ability of the system to determine its position with respect to the environment. Planning refers to the process of making purposeful decisions typically involving objectives like bringing the vehicle from a start location to a goal location while avoiding obstacles and optimizing over-designed heuristics. Finally, control refers to the ability to execute the planned actions that have been generated by the higher-level processes [114].

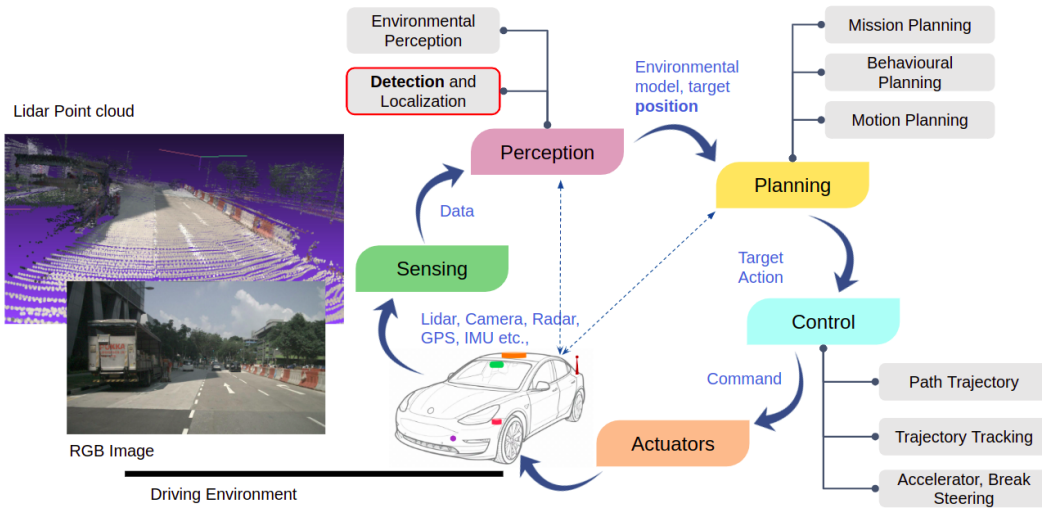


Figure 1.4: Generic perception system for autonomous vehicles

This thesis focuses on the perception of autonomous vehicles offer greater autonomy and complexity, emphasizing the detection and localization subsystem, particularly object detection. Object detection in Autonomous Driving (AD) applications is very complex as it requires the perception system [79] to detect the location and depth of these objects in a highly dynamic scene containing multiple object classes. Visible spectrum camera sensors are the primary perception modality in most 2D object detection research. Most image-based

state-of-the-art 2D detection methods have achieved the current best accuracy in localisation and classification of the objects appearing in a scene. However, image-based object detection methods have limitations in extracting depth information, specifically in the case of single-frame (monocular) or stereo-based depth estimations methodologies that cannot achieve the accuracy expected by the perception system. Furthermore, image-based depth estimation can be erroneous due to changes in lighting, weather, and other environmental conditions. It requires a complex and computationally expensive process to determine the object's depth with only image modality. Although, recent monocular image-based methods have been proposed that use an RGB image to predict objects on the 2D plane and then perform 2D to 3D lifting to create 3D object detection results with lesser computational cost. However, the performance is still far from satisfactory due to the lack of reliable depth prior and the variance of the object scale caused by perspective projection [28].

In contrast, 3D object detection using lidar produces more accurate depth estimation than image-based sensors. However, the accuracy of 3D detection achieved with lidar sensors or the combination of image and lidar sensors is lower compared to the accuracy of 2D object detection. Therefore, it is necessary to bridge the gap from 2D image space to the 3D metric space [61] to achieve a reliable object detection and depth estimation module for AV systems.

Spatial image features along with temporal information (i.e, timestamp, frame number, spatial feature variation over time) are extensively used for improving the accuracy of moving 2D object detection and tracking in video streams, object path prediction [34], human pose estimation, activity recognition [46] and flow predictions [119]. Unfortunately, the utilisation of temporal aspects in data for 3D object detection has had limited focus due to the non-availability of large-scale, sequentially annotated data sequences for extracting temporal information from multiple frame references and various modalities in the dataset. However, the release of large-scale perception datasets like nuScenes [32] by 'NuTonomy' and waymo [53] by 'Waymo Driver' consisting of extensive annotated sequence data has drawn considerable interest among researchers. The availability of sequenced data has enabled the extraction of spatial and temporal features of the object by employing multi-frame references. These sequenced references help the deep learning models to establish coherence for understanding the structural information of an object in the scene to estimate the

accurate depth and other attributes.

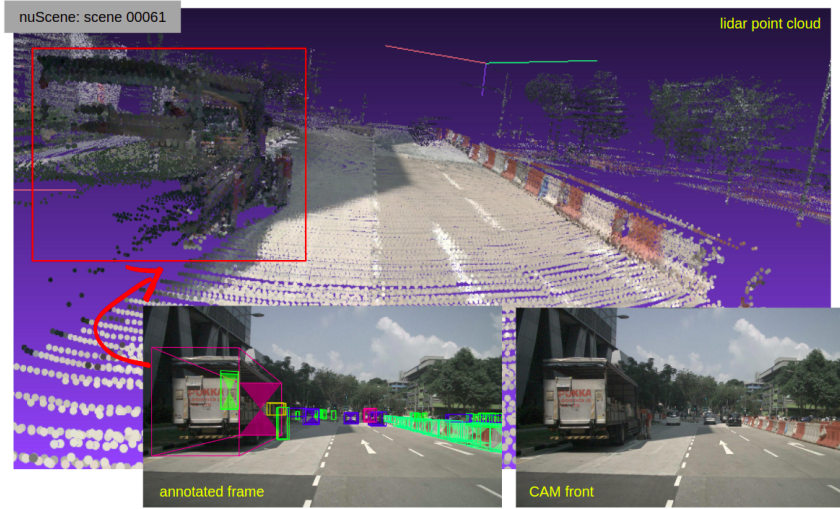


Figure 1.5: Representation of lidar and image data for the scene-0061 in the nuScenes dataset

Fig. 1.5 presents the visual representation of lidar and image data of a scenes-0061 from nuScenes [32] dataset containing various object annotations on the image plane. Autonomous vehicles rely on the perception of their surroundings to ensure safe driving. The perception system in AV uses object detection algorithms to accurately determine objects such as pedestrians, vehicles, traffic signs, and barriers in the vehicle's vicinity. Fig. 1.3 presents a visual depiction of an AV acquiring the data of its driving environment during its navigation. The data acquired from the image and lidar sensor can be used to extract the various temporal references of the objects over time that can be eventually utilized to develop a robust, stable and accurate detection model for the perception system.

Why this research? The automotive industry is transiting from ADAS to AD. The vehicle is equipped with ADAS technology really on the data of the sensor to provide assistance to the driver, leaving the driver's judgment to take responsibility for the situation. However, automated driving and the advancement in the sensor and hardware technologies used in the development of autonomous vehicles have increased the competition in the automotive industry. To enable fully autonomous driving and use by humans these vehicles should be highly reliable and trusted for having a safe driving system on the road with 'zero' casualties. This requirement has raised the expectation of the algorithms designed for the perception system to have human-like perseverance in the driving environment in determining

the precise positions of the objects in the scene. The continuous data acquisition in the AV exhibits temporal continuity, and objects appearing in successive frames will have relative motion due to their own motion or camera motion or when both are in motion . This allows the utilization of detection results and temporal features of the objects from earlier frames to refine predictions and/or parameter estimations in the current frame. There has been limited research about extracting temporal feature using lidar and image data, and its potential usage during the estimation of object parameters for improved localisation and detection. Further, availability of large-scale sequenced annotated lidar and image data has paved the way to enhance the knowledge to understand the dynamic nature of objects in an autonomous vehicle driving environment has motivated to propose the pipeline incorporating spatial features from lidar and image data detailed in Chapter 4, temporal feature extraction along with attention mechanism detailed in Chapter 5 and fusion strategies detailed in Chapter 6 for improved 3D objection detection.

1.2 Hypothesis and Research Question

The hypothesis of this research is defined as follows: *“Multimodal spatio-temporal features can contribute to bridging the gap between 2D image space and 3D metric space by improving the accuracy and reducing the error score of deep learning frameworks for 3D object detection.”*.

To investigate the above-stated hypothesis following questions are formulated:

- **Question 1 (RQ1): How best to represent/prepare multimodal data for training the object detection architecture?**

In this research, datasets with synchronised lidar and image data frames are considered in designing the 3D object detection framework and data representation is crucial while training a deep learning network to extract the high-level features for better representation of raw data and passed to the prediction layer for estimations depending on the task. The preliminary findings during the pilot study conducted in Chapter 3 led to the selection of the baseline framework. Further, exploration of data representation is

performed in Chapter 4 to utilise most suitable representations for the current research work effectively.

- **Question 2 (RQ2): How can temporal features be extracted from multimodal data?**

Spatial features and temporal information can aid the deep learning network in understanding the variation in the data about the object/s or the scenes by learning the changes in the successive appearances during the autonomous vehicle navigation in a dynamic driving environment. The preliminary finding of temporal understanding presented in Chapter 3 assisted in addressing the RQ2 by dividing it further into two aspects:

- *Question 2.1 (RQ2.1): What is the minimum number of sequential data needed to extract temporal information to positively impact the training and final accuracy during inference?*

The purpose of RQ2.1 is to understand the minimum number of frames containing the object of interest, which the network sees to extract the temporal information and gain knowledge regarding the structural variations in the object and the data representing the object to enable better estimation of the object parameters. Chapter 4 presents the investigation and findings of RQ2.1.

- *Question 2.2 (RQ2.2): Which is the most suitable approach for including temporal information? Potential candidates include Recurrent Neural Network (RNN) based Long Short Term Memory (LSTM), Gated Recurrent Unit (GRU), Temporal convolution network (TCN) or frame stacking for extracting temporal information from the sequence data.*

The purpose of RQ2.2 is to explore the approaches for extracting temporal features from the input data sequence. The most straightforward approach is 'frame stacking', where consecutive frames are stacked and fed to the network without any shuffle. In this approach, the deep detection network is expected to learn the temporal features from the sequenced data during training. Furthermore, temporal information can be extracted using RNN-based models such as Long

Short Term Memory (LSTM) or Gated Recurrent Unit (GRU) module. Both these models are explored in Chapter 4 to verify the suitability to adopt into the proposed framework presented in Chapter 5 and Chapter 6.

- **Question 3 (RQ3): How best to fuse multimodal and temporal data into object detection networks, and what fusion schemes need to be employed? Where multiple layers are used to incorporate data, how does the inclusion of additional layers impact the network’s complexity (i.e., time and memory usage)?**

The research focuses on employing lidar and image data. Combining multiple modalities (lidar and image) exploits the individual characteristics of the data employed to train the detection network, which eventually assists the network in learning a better representation of the data, resulting in reduced errors and improved accuracy. Other information from IMU, and local vehicle maps are not considered while exploring the fusion models. The RQ3 aims to investigate the fusion strategies to fuse lidar and image data representation into the detection network to mitigate the shortcoming of using a single modality. Chapter 6 explore the fusion strategies for combining the lidar and image data for estimating the object parameters. Any additional functionality or layer added to the framework increases the network’s complexity in terms of the number of parameters to be trained, memory usage, per-frame computation time, and model size. Chapter 6 provides a summary of all the network configurations employed in this research to understand the complexity associated with the network depending on the modality employed.

1.3 Research Contribution

The contribution of this research work are listed as follows:

- Recommending a data representation to effectively aid the detection architecture to learn the spatial features of the object and the temporal feature from the sequence data.

- Analysing the influence of prior localisation of the objects using an off-the-shelf pre-trained 2D detector while extracting the lidar point cloud features and its influence on the model’s performance in terms of mean average precision (mAP) using the KITTI dataset.
 - Analysing the minimum number of sequenced frames required to effectively extract the temporal information of the object to propagate into the detector network for understanding the structural variation through lidar point cloud distribution to handle the misalignment problems during the final estimation.
 - Analysing the impact of using single sweep versus multiple-sweep lidar point cloud data for training and estimating the 3D bounding box parameters on the nuScenes dataset.
 - Analysing the effect of varying the feature vector length of lidar data representation on the performance of the detection model on the KITTI and nuScenes dataset.
- Proposal of optimum attention mechanism configuration to focus on the salient points of the lidar point cloud distribution within the region proposals representing the object.
 - Incorporating spatio-temporal information coupled with attention mechanism into the 3D detection architecture to improve 3D detection accuracy and reduce the true-positive error metrics.
 - Evaluation of fusion strategies to fuse image and lidar data to improve the 3D object estimations.
 - Analysis of the computational and resource complexity while handling multiple modalities to improve 3D detection accuracy.

1.4 Thesis Outline

The thesis is structured as follows:

Chapter 2 presents an overview of the existing works in the literature related to object detection based on the sensing modalities and data representation used for estimating the

3D bounding boxes. This chapter also provides an overview of the datasets used for the development of the perception system-related task and the evaluation metrics to study the performance of the network employed in this research.

Chapter 3 briefly explains the preliminary work on multiple 2D object detection and tracking. The outcome achieved on 2D detection and tracking motivated to extend the horizon of object detection in 3D space by using lidar point cloud data and exploring suitable 3D detection architectures to estimate the 3D box parameters for selecting the framework to address the research questions.

Chapter 4 provides an overview of the KITTI and nuScenes dataset used for the investigation and to generate a baseline performance of the detection architecture. This chapter also presents a series of analyses carried out to understand the data, temporal aspect and training configuration, which are employed to manifest the training and testing of the proposed framework.

Chapter 5 presents the proposed framework using lidar-only data to extract the spatio-temporal information incorporated by employing the convolutional-LSTMs (CLSTM) and kernel attention mechanism (KAM) module. The learning of Chapter 4 was incorporated during the investigation, which led to improved accuracy with reduced true positive metric error scores.

Chapter 6 presents the fusion strategies to fuse the image and lidar representation into the 3D detection framework proposed in Chapter 5. The chapter also provides a brief overview of various configurations employed in this research, highlighting the computational and resource complexity involved while handling multiple modalities for improved 3D detections.

Chapter 7 summarises the research methodologies employed to address the proposed research questions and highlights the contributions made in this research work. The chapter discusses the feasibility of applying the proposed framework in the context of real-world applications in autonomous vehicles equipped with improved sensor and hardware technologies. Finally, concludes by providing future directions to current research on 3D detection using multiple modalities and how to handle uncertainties that might exist or arise within the data sequence.

Chapter 2

Related Work and Background

Object detection is the process of identifying and localizing multiple semantic objects of a particular class (such as humans, buildings, or cars) in a single image. Several datasets have been released for object detection challenges, and specific performance metrics have been developed to account for the spatial position of the detected object and the accuracy of the predicted categories. The literature review presented in this section is structured as shown in Fig. 2.1. An overview of the most relevant work to explore the deep learning frameworks for object detection are briefed in Section 2.1 and Section 2.2. Section 2.4 briefly describes the metrics employed to evaluate the performance of the object detection architecture, and Section 2.3 provides the available autonomous vehicle datasets used by the researchers to develop technologies related to perception systems. Finally, Section 2.5 summarises the related work and provides insight into the planning of preliminary work detailed in Chapter 3 and methodologies to address the research questions in Chapter 5 and Chapter 6.

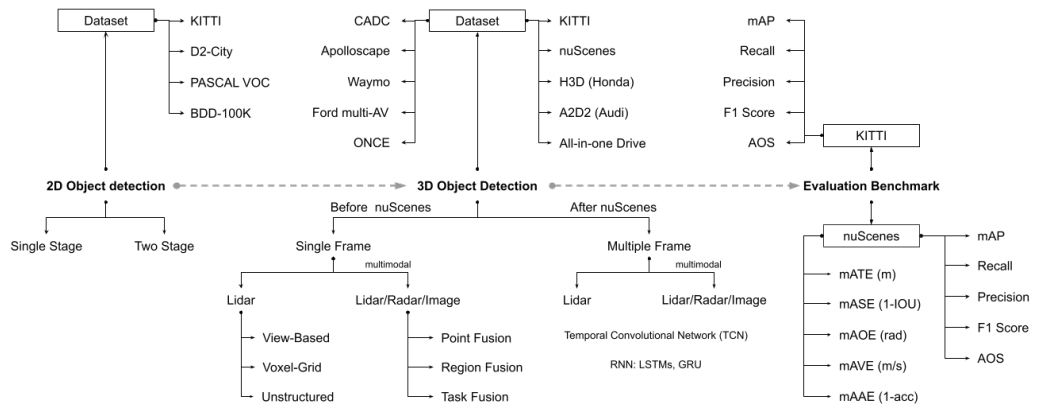


Figure 2.1: structure of literature presented in this chapter

2.1 2D Object Detection

The research work commenced with employing a 2D object detector to localise and track the objects for the Vision Guided Driver Assistance System (VIDAS)¹ project. The Deep convolutional neural network (DCNN) based detection architectures that are classified into single-stage and two-stage frameworks as mentioned in the literature [90, 45, 6, 5] were considered to design the pipeline for object detection and tracking. Single-stage detectors, such as You Only Look Once (YOLO) [117], and Single-shot multiBox Detection (SSD) [123] and their derivatives directly predict class probabilities and bounding box offsets simultaneously from feature maps extracted on an image scene. In contrast, two-stage detectors, including RCNN [134], R-FCN [121], SPPNet [128], Fast RCNN [127], Faster-RCNN [132], Mask RCNN [109] and Light head RCNN [111] consist of an intermediate step to generate region proposals where objects might be located, and then these proposals are further refined to predict the class and location of the object.

This section highlights some of the 2D object detection approaches which were considered while designing the pipeline for the VIDAS and Huawei-Insight projects which provided the initial understanding and motivation to explore further 3D object detection approaches presented in the next section. The exploration started with one of the most popular single-stage object detection networks from the YOLO family, initially proposed by Redmon et al. [125] in 2016. The YOLO is based on a unified detector that considers object detection as a regression problem from image pixels to spatially separated bounding boxes and associated class probabilities. The region proposal generation stage is dropped, and YOLO directly predicts detections using a small set of candidate regions. Unlike region-based approaches, e.g., the Faster-RCNN framework proposed by Ren et al. [132], which predicts detections based on features from local regions, YOLO uses the features from the entire image globally. Since YOLO sees the entire image when making predictions, it implicitly encodes contextual information about object classes and is less likely to predict false positives in the background. However, YOLO makes more localization errors due to the coarse grid division of the image into bounding boxes and therefore, YOLO may fail to localize

¹This work has been partly funded by the EU H2020 Projects VI-DAS (grant number 690772) link: <http://www.vi-das.eu/>

some objects, especially small ones, because, by design, each grid cell can only contain one object. Redmon and Farhadi [117] proposed YOLOv2, an improved version of YOLO in which the custom GoogLeNet network is replaced with a simpler DarkNet19. In addition, GoogLeNet utilizes strategies from existing work, such as batch normalization, removing the fully connected layers, and using good anchor boxes learned with k-means and multiscale training.

To achieve real-time processing without sacrificing detection accuracy, Liu et al. [123] proposed the Single Shot Multibox Detector (SSD), which is faster than YOLO [125] and has accuracy comparable with state-of-the-art region-based detectors, including Faster-RCNN [132]. SSD effectively combines ideas from Region Proposal Network (RPN) in Faster-RCNN, YOLO and multiscale convolutional features, to achieve fast detection speed while still retaining high detection quality. Like YOLO, SSD predicts a fixed number of bounding boxes and scores for the presence of object class instances in these boxes, followed by a non-maximal suppression (NMS) step to produce the final detection. However, SSD uses shallower layers with higher resolution for detecting small objects. In addition, SSD performs detection over multiple scales for objects of different sizes by operating on multiple convolution feature maps, each of which predicts category scores and box offsets for bounding boxes of appropriate sizes.

Recent approaches in 2D object detection covering many aspects of generic object detection, including detection frameworks, object feature representation, object proposal generation, context modelling, and training strategies, are outlined in [45, 6, 2].

2.2 3D Object Detection

Visual camera sensors are the primary perception modality in 2D object detection research. However, camera-based object detection performance can vary due to changes in lighting, weather, and other environmental conditions. Since the scale of the scene is unknown, it requires a computationally expensive process to depict the depth of the objects as the detection occurs in the projected image space. Depth estimation is a crucial and necessary functionality for Automated Driving Systems (ADS). Therefore it is necessary to bridge the gap from

2D image space to the 3D metric space [61] as depth estimation using image-based methods can be achieved using a single camera or by using stereo cameras are computationally expensive, requiring first matching and synchronizing the images. Although, recent monocular image-based methods like DFR-Net [28], MonoDIS [82], M3D-RPN [64] have been proposed that use an RGB image to predict objects on the 2D plane and then perform 2D to 3D lifting to create 3D object detection results with lesser computational cost. However, the performance is still far from satisfactory due to the lack of reliable depth prior and the variance of the object scale caused by perspective projection [28]. Lidar and Radar utilize other modalities and offer an alternative to address the challenge of depth perception.

In recent years, 3D object detection methods for autonomous vehicles have been categorized into single-frame reference and multiple-frame reference-based methods. Single-frame reference frameworks are designed by considering lidar data only or fusing information from multiple modalities. Further, based on the input data representation, these methods are divided into three sub-categories: image-based, point cloud-based and multimodal fusion-based methods. Image-based methods are not considered in this thesis due to the complex computational process required to compute the 3D estimation of the object. The point cloud-based and multimodal fusion-based methods are presented in Section 2.2.1. Multiple-frame reference framework employs more than one frame by considering the estimation made in the previous frame to learn the object feature and incorporate the variation during the computation of 3D estimation in the current frame. A brief overview of the methods in the literature is presented in Section 2.2.2.

2.2.1 Single Frame Reference Methods

Lidar point cloud data

3D object detector frameworks can be categorized into single-stage and two-stage detectors. The methods can be further divided into three sub-categories based on the representation of lidar point clouds: view-based, voxel-grid based and unstructured point cloud-based methods.

View-based methods: In view-based methods, a lidar 3D point cloud is projected onto a 2D plane, including image, spherical, cylindrical, or top-views (bird's-eye-views). Then these

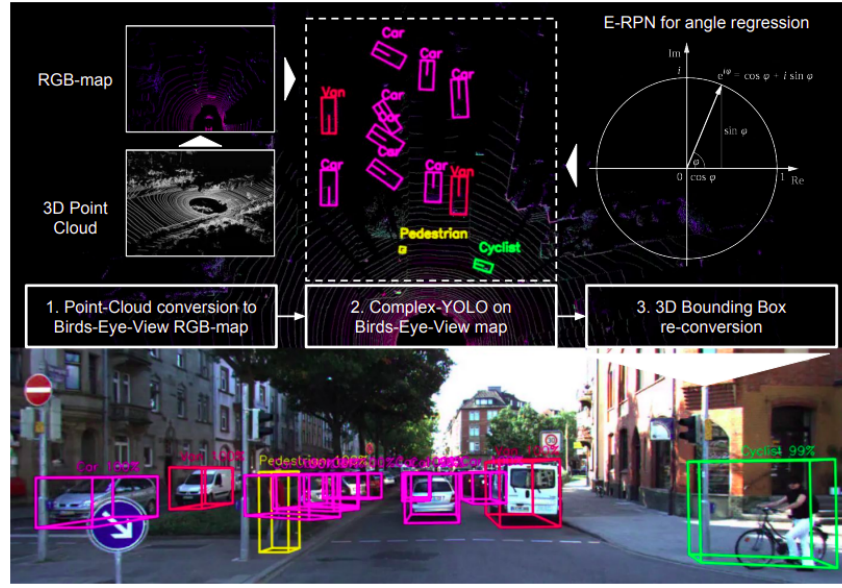


Figure 2.2: Representation captured from Complex-YOLO [100], the method operates on Lidar only based birds-eye-view (BEV) RGB-maps to estimate and localize accurate 3D multiclass bounding boxes

projected views are processed using a convolutional deep neural architecture to obtain 2D bounding boxes. These 2D bounding boxes are then regressed by position and dimension to compute 3D bounding boxes. Single-stage architectures, like Fast and furious (FaF) [95], Complex-YOLO [100] and YOLO3D [88], have demonstrated excellent performance in both speed and accuracy by exploiting a single-stage detector on bird's-eye-view (BEV) representations and mapping the features directly to classification scores and bounding boxes through a single-stage, unified CNN model.

Complex-YOLO [100] and YOLO3D [88] extended YOLOv2 [117] to perform 3D object detection and classification from the BEV representation projected from the 3D lidar point cloud with real-time performance. Complex-YOLO uses a simplified YOLOv2 architecture, expanding it to a specific complex regression strategy to increase speed and performance. To do this, Complex-YOLO, as shown in Fig. 2.2, utilises a specific Euler-Region-Proposal Network (E-RPN) for reliable angle regression to detect accurate multi-class 3D objects with orientation. However, it uses fixed height and z-centre locations in the predicted 3D bounding boxes. In addition, as Complex-YOLO translates the orientation vector to real and imaginary values, angle regression does not guarantee or preserve any correlation between the two components. Recognising this, YOLO3D extended the loss function

of YOLOv2 to include yaw angle, the 3D box centre in cartesian coordinates and the box height as a straightforward regression task.

FaF [95] conducts single-stage, proposal-free, real-time 3D object detection that efficiently uses height-encoded BEV input by assuming that the objects are on the ground. The input to the detector is a 4D tensor created from multiple consecutive temporal frames to perform 3D convolution. FaF pioneered jointly learning 3D detection, tracking and short-term motion forecasting from lidar point clouds in driving scenarios. Yang, Luo and Urtasun [103] proposed PIXOR, a single-stage proposal-free framework employing 2D convolution instead of 3D convolution to estimate the 3D object parameters using BEV representation comprising occupancy and intensity as the features maps of lidar data. The oriented 3D estimations are decoded from a single-branch multi-task header network based on pixel-wise neural network prediction. VeloFCN [122] proposed by Li, Zhang, and Xia generated lidar feature map using range and height feature which are fed to 2D fully convolutional networks (FCN) [131] to predict the object score and location. Bo Li [110] extended the work of VeloFCN to adapt FCN to perform 3D convolution on point cloud data to detect objects and estimate oriented object bounding boxes in an end-to-end fashion.

LMNet [96] used intensity, range, distance, side and height features to generate a five-channel lidar feature map as input to the network. Like VeloFCN, the authors used deconvolutional-based FCN to gather richer contextual information with lesser parameters to predict the 3D object parameters in the feature vector (FV) plane. Gregory P et.al [74] proposed using FCN to predict a multimodal distribution for each point, and then these distributions are fused to generate a prediction for each object. The input feature vector map consists of five-channel, which include features of the range, height, intensity, azimuth angle of the point and binary flag indicating whether the cell contains a point.

Summary: FV-based methods project the point cloud into a front view to form a dense 2D map easily to utilize the off-the-shelf 2D detector. The depth information encoded in the FV map allows capturing dependencies across different views, such as raw point cloud and camera image, which benefits sensor fusion. However, projection inevitably causes a loss of detail. Due to the perspective nature, the scale varies and occlusion problems encountered in 2D detectors are tricky in FV-based detection as well. In Projection-based

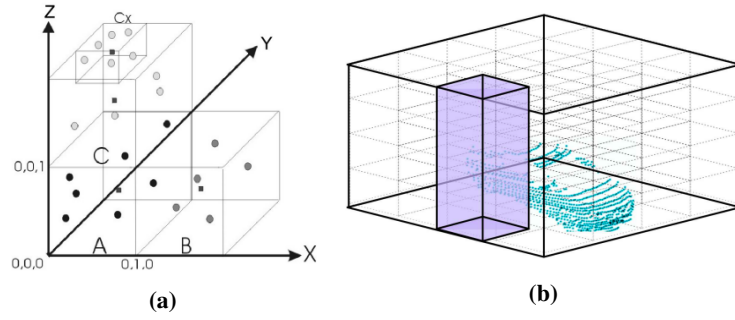


Figure 2.3: (a): Schematic diagram of the voxel procedure of LiDAR data from [146], (b): Voxel Representation using height directly as input feature as in [95]

methods, the point cloud data is projected in a bird's eye view to forming a 2D BEV map. This technique has been widely researched, but it heavily relies on the update of 2D detection algorithms. Calculation of object yaw angle is easier as the size of the object in the BEV map remains constant regardless of range, which provides prior information on object size and improves the network training process. However, due to the small effective area, the network cannot handle occlusion in vertical space or pole-like object which has few points after BEV projection. Also, ground plane fluctuations will accumulate height prediction errors. Various hand-engineered features are designed to keep the spatial feature of a 2D projection map. However, since the resolution of the projection map and the number of feature channels affect the calculation efficiency, the irreversible information loss introduced by projection still exists and limits the object detection accuracy.

Voxel-based methods: In voxel-grid-based methods, the lidar point cloud data is discretized into a volumetric 3D grid or voxel representation. The voxelization process is presented in Fig. 2.3 showing three voxels labelled A to C. Each object has either binary occupancy (present/absent in the voxel) or a continuous point density (number of points in the voxel). The shape information is preserved in the volumetric representation and can be applied directly to the 3D CNN. Zhou and Tuzel proposed a lidar-only architecture named VoxelNet [107], which takes raw point cloud and simultaneously learns a discriminative feature representation to predict accurate 3D bounding boxes using a single end-to-end trainable network. VoxelNet has three functional blocks: (1) Feature learning network, (2) Convolutional middle layers, and (3) Region proposal network. First, the feature learning network separates the point cloud into equally spaced 3D voxels, transforming points within each

voxel to a vector represented as a 4D tensor characterizing shape information through voxel feature encoding (VFE) layers. After that, the convolutional middle layers aggregate the neighbourhood voxel features, adding more context to the shape description and converting the point cloud into a high-dimensional volumetric portrayal. Finally, the region proposal network takes the volumetric representation and obtains the 3D detection results.

The major bottleneck with VoxelNet [107] methods was the high computational cost involved with a 3D convolutional network, which increased the computational complexity of 3D CNN cubically with the voxel resolution. To shorten the training and inference speed, Yan, Mao, and Li proposed SECOND [102] framework to improve VoxelNet. SECOND used sparse convolutional layers [92] after the VFE to convert the sparse voxel data into 2D images and applied an angle loss regression mechanism to improve the orientation estimation. This sparse convolution only operates on the locations associated with input points, resulting in much stronger 3D detection performance with improved speed. However, SECOND still includes the bottleneck of using expensive 3D convolutions.

Since lidar point cloud data is sparse when compared to dense image data, researchers transform the point cloud data into regular 3D voxels or project them into perspective images. There is always a loss of spatial information as a result. Some of the recently proposed architectures Pointnet [115] and Pointnet++ [115] use unordered raw point clouds without pre-processing. Focusing on the potential information in the verticals and tackle the computation complexity of the 3D convolutions, Lang, Alex H., et al. [70] proposed a 2D convolution-based end-to-end learning framework that learns a representation of point clouds organized in pillars, where a pillar is a column that can extend infinitely up and down. After voxelization, the authors constructed a sparse pseudo-BEV feature map by transforming the points into vertical columns (pillars). They employed a simplified version of Pointnet [115] to extract the point-wise features inside the pillars of the voxel space. Max-pooling is applied over channels aggregated with z-axis information encoded as pillar position is passed to a single-stage 2D SSD [123] to detect and relate the 3D oriented boxes instead of expensive 3d convolutions.

Kuang, Hongwu, et al. expanded the idea of VoxelNet [107] and proposed a single-stage Voxel Feature pyramid Network (Voxel-FPN) [42] that uses raw data from lidar sensors only

for 3D object detection. Unlike its former voxel-based approaches, Voxel-FPN encodes multi-scale voxel features, which are realised using two significant blocks: voxel feature encoding - FPN (VFE-FPN) and region proposal network - FPN (RPN-FPN). In the VFE-FPN block, the raw points clouds are divided into voxels of multiple scales, and a simple PointNet is used to extract the voxel features in a bottom-up manner, followed by an FPN network to fuse voxel features of different sizes. The RPN-FPN blocks perform the multi-scale feature aggregation, decode the feature maps in a top-down way from various scales, and associates them with the final detection output.

Yi, Hongwei, et al. [60] handled the voxelization processing by considering the free-of-charge semantic information extracted from the bird's eye view feature map. An FPN was employed to extract the contextual information through a segmentation mask to aid the network in learning the probability of a segment containing the object within the voxel cell. While SegVoxelNet [60] proposed the design depth-aware detection head with convolution layers of different kernel sizes and dilation rates to explicitly model the distribution differences with increasing depth and aid the model in automatically focusing on detecting the target detection range.

Liu, Zhe, et al. [47] emphasized the need to improve the detection accuracy for complex objects like pedestrians and the effect of noisy point cloud data on the performance of the 3D object detection architecture and proposed end-to-end voxel-based 3d object detection pipeline with triple attention (TA). The TA module makes the model selectively focus on informative points and suppress the unstable points by jointly learning the point-wise, channel-wise and voxel-wise attention followed by element-wise multiplication. Finally, the localization accuracy of the model is elevated using a coarse-to-fine regression model without adding any overhead to the computation cost.

Confident IoU-Aware single-Stage object Detector (CIA-SSD) [27] highlights the problems associated with the single-stage detector while aligning the localization accuracy and classification confidence as object localization, and classification tasks are addressed separately. CIA-SSD uses a lightweight spatial semantic feature aggregation module to adaptively fuse high-level abstract semantic features and low-level spatial features for accurate

bounding box prediction and improved classification confidence score. The prediction confidence is further rectified through the IoU-aware confidence rectification module to make the confidence more consistent with the localization accuracy. Non-maximal suppression (NMS) based on the distance-variant IoU-weighted is used to obtain smoother regressions and avoid redundant prediction.

In contrast, Yin, Zhou and Krahenbuhl proposed a two-stage framework CentrePoint [26] to address the axis alignment issues observed in anchor-based methods [107, 70, 36] when the objects are rotated in the 3D world. CentrePoint employs VoxelNet or PointPillar to represent the input point-cloud data which are then transformed into an overhead map view for applying a keypoint detector to find the object corners. All the objects are detected and tracked as points. In the first stage, point features are extracted using a keypoint detector to estimate the object centres and are regressed to determine the size, orientation and velocity in 3D space. These estimations are further refined to localize the object in the second stage. Similarly, Wang, Guojun, et al. also addressed the axis-alignment issues by proposing an anchor-free CentreNet3D [55] architecture, inspired by CentreNet [87]. CentreNet3D uses the keypoint estimation to determine the objects' center points and directly regress 3D bounding boxes. In addition, the corner attention module is used to focus the model on the points for accurate boundary estimation. Finally, the alignment of the predicted bounding boxes with the corresponding classification confidence is aligned using a keypoint-sensitive warping operation.

Summary: Voxel-based methods are well-researched in-point cloud-related tasks which make use of 3D CNN to learn the local dependencies for estimating the depth information in a 3D representation. The non-homogeneous points of the lidar data with various channels are naturally normalised during voxelization. However, it also suffers the loss of information due to the merging of multiple points falling in one voxel. Although previous works consider simple statistical features on the scatter of points, most methods have shifted towards a learned paradigm to capture the 3D geometric inside voxels. The explicit modelling of the entire 3D space results in a sparse representation and inevitable calculations, which increases the computation complexity of the model and this also makes the training of a voxel-based model not feasible.

point-based methods: Projection-based and voxel-based methods represent the raw point cloud into image grids or 3D voxels to extract feature maps and estimate the bounding box parameters. Although the representation in these methods is simple and easily adapted to utilise the 2D detection framework, they exhibit a loss of information and suffer geometric misalignment. The introduction of PointNet [115] architecture paved the way to directly process the unstructured lidar point cloud data to capture local and global point features and ease the information loss. The researchers captured the idea of PointNet to adapt the segmentation task into various 3D-related tasks leading to the proposal of point-based 3D object detection methods. Shi, Wang and Li proposed PointRCNN [80], a two-stage bottom-up 3D object detection framework. In the first stage, initial box proposals are generated by learning the point-wise features extracted using pointnet++ [116] to segment the raw point cloud and generate 3D proposals from the segmented foreground points simultaneously. The framework employs a bin-based bounding box generation strategy to split each foreground segmented point into a series of bins. The one with the highest object confidence score is selected as the object centre for the current foreground points. In the second stage of PointRCNN, the box location and orientation are further refined by the semantic point-wise features, foreground mask and canonical coordinates of its inside points. Although these methods yielded good performance due to their longer inference time, they cannot be applied to real-time autonomous driving systems.

Qi, Charles R., et al. [78] observed the unstructured and sparse nature of lidar data; the object centres are likely to be in space, even far away from any scanned points from the sensor. Therefore, directly proposing 3D boxes from a visible point makes it hard to aggregate the context around the object centre resulting in incorrect detection results. The object's centre is first generated through a voting mechanism similar to the classical Hough voting to solve this problem. Centroid points are then clustered and aggregated to generate box proposals.

Li, Jiale, et al. proposed 3d IoU-Net [44], a two-stage framework which generates the proposals and point-wise features directly from the point cloud data. The method uses the attention corner mechanism (ACA) to perform feature pooling to obtain perception invariant prediction head by aggregating the local point cloud feature from each perspective of eight

corners in the bounding box. Weights to these predictions are assigned using perspective-wise and channel-wise attention mechanisms. Corner-geometry encoding (CGE) encodes the geometric information of the bounding box to enhance IoU sensing feature learning. Finally, the IoU alignment is used to resolve the mismatching at the NMS stage. The complex upsampling layer and refinement stage are crucial and commonly employed in most point-based method [107, 80, 104, 86]. Yang, Zetong, et al. proposed a single-stage anchor-free point-based detector 3D-SSD [59] incorporating fusion sampling (FS) strategy in a set abstraction (SA) layer to achieve the detection on less representative points. The framework used a 3D centre-ness assignment based on VoteNet [78] to generate candidate points later passed to an anchor-free regression head to estimate the size centre coordinates and orientation of the 3D box for each candidate point.

point-voxel based: Chen, Yilun, et al. proposed Fast Point R-CNN [67], a unified two-stage framework to use both voxel representation and raw point cloud data. Initial box proposals are generated by employing a voxel-based region proposal network that directly exploits point clouds' voxel representation, followed by a lightweight PointNet to refine the predictions further. This assists in inspecting coordinate points to capture more localization information with enlarged receptive fields. The attention mechanism is used to effectively fuse each interior point's coordinates with each point's convolution feature in the initial prediction. 2D and 3D convolutions are employed to preserve information, and the attention mechanism makes each point aware of its context information for final box refinement. Fast-PointRCNN consumes more time for generating point-wise representation, similar to methods proposed in [80, 104]. Yang, Zetong, et al. proposed a two-stage sparse-to-dense 3D Object Detector (STD) [86] where regional proposals are generated by seeding each of the points in the point cloud data element with appropriate spherical anchors to a proposal generation network. The semantic context feature for each point and generating object-ness score to filter anchors are extracted using PointNet++, which are then passed to the PointsPool layer. By following this strategy, the detector takes advantage of both point- and voxel-based methods for proposal feature generation by transforming interior point features from sparse expression to dense representation, enabling efficient CNNs and end-to-end training. The final refined prediction is achieved in the second stage, where the 3D IoU

prediction branch helps alignment between classification score and localization. Although the speed of STD is improved from its former point-based methods, it is still slower than the voxel-based methods.

Point-Voxel CNN [72] took the leverage of the point-based to reduce the memory footprint and voxel-based methods to obtain the contiguous memory access pattern and improve the localization. The framework segregated fine-grained feature transformation and neighbour aggregation processes to take advantage of each branch effectively. The upper voxel-based branch initially transforms the points into low-resolution voxel grids, and then it aggregates the neighbour points by voxel-based convolution followed by de-voxelization to convert them back to points. The lower point-based branch extracts the features for each point using MLP and can afford high resolution without aggregating the neighbour's information. Information from the upper and lower branches is fused to get complementary cues for the prediction of objects. Similarly, PointVoxel-RCNN [52] extracts the point-wise features from the PointNet-based network to refine the proposals generated from 3D voxel CNN. The point-wise feature is encoded from a small set of keypoints sampled by the farthest point sampling (FPS) algorithm using the whole point cloud data. The keypoint feature is generated by concatenating features extracted from the raw point cloud, BEV features extracted by voxelization and z-axis features obtained by aggregating the neighbouring voxel-wise features via PointNet-based set abstraction for summarizing multi-scale point cloud information. All the keypoints are re-weighted so that the foreground points contribute more to predicting the accurate box parameters.

Downsampling of convolutional features is a commonly employed strategy in most detection architectures. This results in a loss of spatial information, eventually affecting the bounding box's localization in the 3d space. He, Chenhang, et al. [36] proposed Structure aware single-stage 3D detector (SA-SSD) network using lidar point cloud data to improve the localization precision of single-stage detectors by explicitly leveraging the structure information of 3D point cloud. A detachable auxiliary network was used to learn this structural information and convert the extracted convolutional features in the backbone network back to point-level representations. To rectify the discordance between the predicted bounding

boxes and corresponding classification confidences, an efficient part-sensitive warping operation was adapted to improve the detection accuracy by reducing the misalignment between the predicted bounding boxes and the classification confidence.

Summary: Point-based methods directly process the irregular point cloud to extract point-wise features by aggregating the neighbouring local feature and then generating predictions based on each point. Most of the point-based methods adopt either Pointnet [115] or Pointnet++ [115] to infer 3D geometry and shape features from irregular points without information loss. Although some of the methods rely on camera image for proposal locating or expensive per-point foreground segmentation networks. Methods like [36, 44, 86] directly locate objects leveraging the sparse characteristics of the point cloud. Point-based methods show strong capabilities and great potential in the current research for designing the 3D detection architecture.

Lidar point cloud and Camera Image

Each of the various sensors usually available in advanced driver-assisted systems (ADAS) has certain limitations. For example, camera-based processing methods provide rich colour, texture information, no depth information. On the other hand, lidar-based processing methods provide depth information but lack texture information. However, the availability of new lidar technologies can provide texture information. The dataset considered in most of the SOTA methods has not used employed an open dataset with such sophisticated sensors and where the dataset contains such sensor data, object annotations are not available which can be used for training 3D detection models. Hence, high-level features extracted from the image data a vital role in object detection and classification, while depth information is critical for accurately estimating 3D location and object size. Furthermore, as the distance from the sensor increases, the point cloud density decreases proportionally. Since texture and depth modalities are essential in 3D object detection, some methods use images and point clouds to improve the overall performance by adopting fusion schemes, as illustrated in Fig 2.4. The deep learning-based approaches available in the literature can be categorised into region-based, point-based, and task-based fusion methods are briefly described in this section. Architectures like MVX-Net [83], ImVoteNet [51] fall into more than one category.

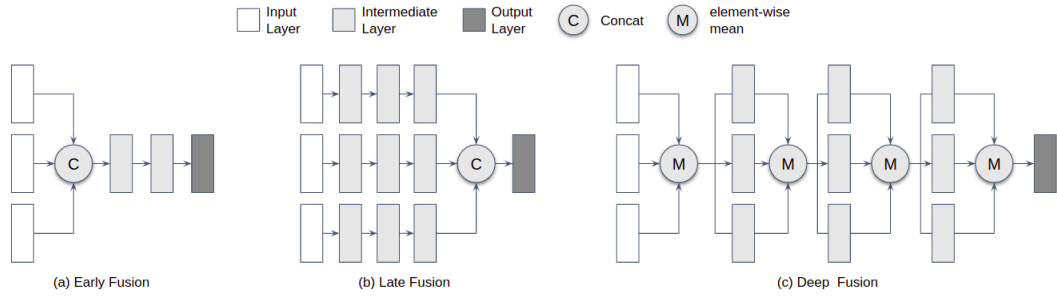


Figure 2.4: An illustration of early, late, and deep fusion [61, 7]

Region-based Fusion Methods: MV3D [65] takes the bird’s eye view and front-view projection of a 3D point cloud together with the RGB image as input to extract feature maps using multi-stream CNNs. The lidar bird’s-eye-view feature map is passed to an RPN, extended from Faster-R-CNN [132], to generate highly accurate 3D region proposals. Each of these 3D region proposals is projected onto all three views of the feature maps. Finally, three types of fusion networks are used to combine region-wise features from multiple views for object classification and oriented 3D box regression. Similar to MV3D, AVOD [93] introduced an early fusion architecture that takes a bird’s-eye-view projected from a lidar point cloud and RGB images to generate high-resolution feature maps to perform multi-modal feature fusion to generate reliable 3D object proposals. These 3D proposals are then transferred to the second stage detector network for oriented 3D bounding box regression and classification. AVOD [93] converts the orientation vector to sine and cosine with four corner representations to preserve orientation information at the corner order while performing corner-to-corner matching.

Liang, Ming, et al. [71] propose to exploit information extracted while performing multiple perception-related tasks to improve 3D detection using multiple sensors. The information from depth maps, lidar bird-eye-view feature maps, images and sparse depth images generated by projecting the LiDAR points to the image are fused to accomplish multiple tasks, including depth estimation, 2D object detection and 3D object detection. All these tasks take advantage of the representation learnt from cross-modalities and enhance the performance of 3D object detection.

Sindagi, Vishwanath et al. proposed a Multimodal voxelnet for 3d object detection (MVX-Net) [83] to combine the early fusion of point features and late fusion of voxel

feature to integrate the lidar and image information fully. Early fusion is achieved by the PointFusion approach, where the point features from the lidar data are projected onto the image plane, followed by image feature extraction from a pre-trained 2D detector. The point feature and image feature are jointly processed by VoxelNet architecture. The late fusion is accomplished through the VoxelFusion approach, where the 3D voxel features generated from the VoxelNet are projected to the image, followed by extracting image features for every projected voxel using a pre-trained CNN. These features are then pooled and appended to the Voxel Feature Encoding (VFE) layer for every voxel and further used by the 3D region proposal network (RPN) to produce 3D bounding boxes.

Tian, Yonglin, et al. [54] proposed to adopt various modalities of lidar data to generate rich features and employ an adaptive and azimuth-aware network to aggregate the local features from the image, bird's eye view maps, and point cloud. The 3D anchors are generated using a ground plane network that uses features extracted from the point cloud to predict the parameters of a plane. The region proposal is generated using image features and bird's eye view maps. An adaptive azimuth-aware fusion network is used to integrate heterogeneous image and point cloud features by explicitly adjusting the intensity of multiple local features and achieving orientation consistency between image and lidar data.

Point-based Fusion: Liang, Ming, et al. [94] proposed an end-to-end learnable architecture that exploits continuous convolutions to fuse image and lidar feature maps at different levels of resolution. The network is trained to fuse the image features by transforming to BEV space where 3D estimations are made. The continuous fusion layer encoded the dense accurate geometric relationship between the positions under two modalities. Meyer, Gregory P., et al. extended the work of LaserNet [74] and proposed LaserNet++ [75] to fuse 2D image data with the 3D lidar data. The image features extracted using the CNN model are mapped to the lidar points in the native range view and passed to a deep layer aggregation network for feature fusion. This fusion technique significantly boosted the detector's performance for long-range object detection and improved the semantic segmentation of the 3D points. Huang, Tengting, et al. addressed the issues related to the fusion of multiple sensors and inconsistency associated between the localization and classification confidence by enhancing the point features with image semantics for 3D object detection and proposed an end-to-end

learnable framework named EPNet [39]. The framework fuses the point features with corresponding semantic image features in a point-wise manner without image annotations. The detection accuracy is improved by stimulating the consistency between the localization and classification confidence through the consistency enforcing loss (CE loss) function. This aided the network in allowing the boxes with high confidence scores to have a large overlap with the ground truth and vice versa.

Qi, Charles R., et al. designed a joint 2D-3D voting scheme for 3D object detection named ImVoteNet [51]. The architecture was built on top of VoteNet [78] framework to estimate the 3D bounding boxes using RGB-D data by fusing 2D votes in images and 3D votes in point clouds. Instead of using multiple modalities, the authors explicitly employ 2D images to extract geometric and semantic features, which are transformed into 3D using camera parameters and depth information to generate pseudo-3D votes. The pseudo-3D votes aided the network in recovering heavily truncated objects or objects with few points improving the network confidence to distinguish geometrically similar objects. A multi-tower training scheme with gradient blending was proposed to ensure the network balances the contributions of 2D and 3D features. However, the fusion methods that employed BEV and Voxel feature representations usually involve the loss of information leading to incorrect estimation. Xie, Liang, et al. [58] tackled this issue by employing a point-based attentive cont-conv fusion (PACF) module to effectively fusion the multi-sensor data information directly on 3D points for 3D object detection. Based on the PACF module, the authors propose a 3D multi-sensor multi-task network called PointCloud-Image RCNN (PI-RCNN), which handles the image segmentation and 3D object detection tasks. The performance of the PACF module is further improved by applying point-pooling and attentive aggregation. These descriptive semantic features, along with the fused features from the PACF module, are utilised to improve the performance of 3D detection.

Task-based Fusion: The computational complexity of processing sparse lidar point cloud data for localizing the objects in 3D space typically grows cubically with respect to resolution and becomes too expensive for large scenes. Qi, Charles R., et al. [97] leveraged the availability of matured 2D detector to localize the objects in image space and later extract the 3D bounding frustum of an object by extruding these 2D bounding boxes. The framework

performed both the instance segmentation and amodal 3D bounding box regression using two variants of PointNet within the trimmed 3D frustums. All the regressions are carried out in a local coordinate system, allowing more accurate prediction than in an absolute coordinate system. Although the performance of the F-PointNet [97] outperformed MV3D [65] and voxel-based method [107] the framework was not an end-to-end learning process. Cao, Pei, et al. extended the framework of F-PointNet [97] by including a multi-view aspect. In this multi-view version of F-PointNet (MVFP) [66], the initial detection results are obtained by combining the RGB image and raw lidar point cloud. An auxiliary BEV feature map encoded with the raw lidar point cloud is used to predict the 2D bounding boxes, which are later used to handle the missed detection of F-PointNet. The authors used the intersection over union (IoU) measure to match all the initial detection results of F-PointNet with BEV maps prediction results. All the missed-detected objects from BEV maps are projected back to the pipeline of F-PointNet until all the objects in BEV maps find a matching detection result in the set of F-PointNet detection results. In 2019, Shin et al. proposed RoarNet [81], where initial estimations of 3D pose are made on monocular images and then derives multiple geometrically feasible candidate locations. These locations are later passed to a two-stage 3D object detection framework similar to Fast-RCNN [127] with PointNet [115] as a backbone network. RoarNet [81] addressed the sensor synchronization issue between the 2D image and 3D lidar point cloud data.

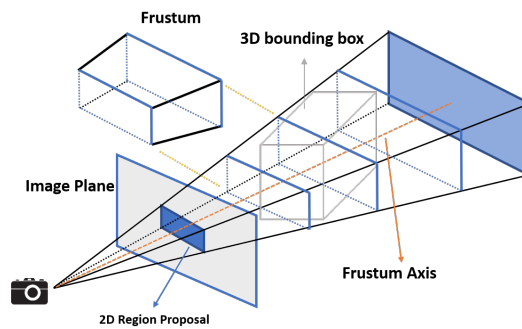


Figure 2.5: Sequence of frustums generated for a region proposal in F-ConvNet architecture [85]

Based on the frustum generation concept in the F-PointNet framework, wang and Jia. adapted fully-connected network and proposed an end-to-end network named Frustum-ConvNet (F-ConvNet) [85] for estimating 3D oriented bounding box from amodal lidar

point clouds. The frustum is generated from the region proposal obtained from a 2D detector as shown in Fig. 2.5, and each of these frustums is segmented into fixed parts for extracting partial features using parallel PointNet. All these features are aggregated to form a frustum-level feature, later passed to the fully-connected network module to extract the multi-resolution frustum features. The object classification and 3D oriented boxes are predicted using two parallel Convolutional layers, which are added on top of fully-connected network. Wu, Yi, et al. [24] also employed a matured 2D detector to get 2D bounding boxes and category labels of objects as prior regional information. These priors are then fed to a precision anchor generation network (PAG-Net) for projecting the boxes into 3D frustum space for more precise and category-adaptive 3D anchors. A multi-layer fusion model is employed that effectively conducts nonlinear and iterative combinations of features from multiple convolutional layers and merges the global and local features of RGB image and point cloud. The lidar point cloud data's irregularities are handled by encoding the point cloud with the bird's eye view (BEV) representation. Wang, Chunwei, et al. proposed PointAugmenting [23], a cross-modal augmentation algorithm for fusing image features and lidar point cloud features for 3D object detection. The method first enhances the point cloud data points with the corresponding point-wise CNN features extracted from a pre-trained 2D detector for adapting to object appearance variations and then performs 3D detection over these enhanced point clouds. The cross-modal data augmentation is designed based on CentrePoint [26] where the virtual objects are constantly pasted into images and point clouds during network training. Bijelic, Mario, et al. [31] proposed a multimodal dataset acquired in over 10,000 km of driving in northern Europe to address the failure of perception systems in adverse weather where the sensory streams can be asymmetrically distorted. The dataset contains rare scenarios, such as heavy fog, heavy snow, and severe rain, with 100k labels for lidar, camera, radar, and gated NIR sensors. The authors propose a single-shot deep multi-modal sensor fusion network that adaptively fuses features driven by measurement entropy. The model achieved good performance mAP in hard scenarios independent of weather, including light fog (84.90 %), dense fog (76.79 %), snow/rain (77.85 %), and clear (79.46 %) conditions.

Another interesting approach proposed by Jianhao Jiao et. al. [40] is to improve the accuracy of 3D object detection by collectively processing the information from multiple lidar sensors. The framework is developed on top of SECOND [102] addressing the extrinsic perturbation in the multi-lidar system for 3D object detection and its influence on geometric tasks. Information captured from the multiple lidar sensors is fused in the first stage, and extrinsic perturbation is handled in the second stage. However, the aggregation of the detection results from different sensors and regularizing onto one common reference frame may lead to errors in a few of the estimations, like counting the number of objects, misplaced overlapping bounding boxes, or false detections. Muresan, Mircea Paul, Ion Giosan, and Sergiu Nedevschi proposed stabilizing and validating 3D object position using multimodal sensor fusion and semantic segmentation [49] to address these issues and proposed two data association methods. The first data association method focused on tracking 3D bounding box objects in a lidar frame and combined multiple appearance and motion features to exploit the available information. The second data association method focused on finding the correspondences between sensor fused objects and the objects detected in RGB images such that the final fused data are enriched by adding the semantic class information. A fusion approach based on the Unscented Kalman Filter [148] and a single-layer perceptron was used for precise object positioning in the common frame and handling the unpredictable behaviour of objects which are provided by multiple types of complementary sensors. Validation of the estimated precise 3D position of the object is done using a fuzzy logic technique combined with a semantic segmentation image.

Similarly, Nobis, Felix, et al. proposed CameraRadarFusionNet (CRF-Net) [76] to address the camera's sensor quality in adverse weather conditions and increased sensor noise in low lighting and at night conditions. The framework enhances current 2D object detection networks by fusing camera data and projected sparse radar data in the network layers and automatically learns at which level the fusion of the sensor data is most beneficial for the detection result. A BlackIn [129] training strategy was introduced to enable the network to focus the learning on a specific sensor type. Nabati, Ramin, and Hairong Qi. proposed CenterFusion [18], a middle-fusion approach to exploit radar and camera data for 3D object detection. The framework first uses a CentreNet [87] to detect objects by identifying their

center points on the image. It then, a frustum-based data association method is used to associate the radar detections to their corresponding object's center point in the image. Finally, the associated radar detections are used to generate radar-based feature maps to complement the image features and to estimate 3D bounding boxes accurately.

Bai, Xuyang, et al. [1] proposed to use a transformer network with a soft-association mechanism to handle the fusion of lidar and image information under inferior image conditions. Spatial and contextual information are fused using a transformer decoder consisting of two layers. The first layer predicts the initial bounding boxes using the lidar point cloud data obtained by a sparse set of object queries, and the second layer fuses the queries from the first layer with the image feature. The framework makes use of an attention mechanism module analogous to methods [47, 55, 44] to determine salient features for effectively fusing image and lidar information.

2.2.2 Multi Frame Reference Methods

The single-frame reference-based 3D object detection methods outlined in Section 2.2.1 were restrained in utilising the temporal aspects aspect in the data due to the nonexistent sequenced annotations. However, open datasets like nuScenes [32] and waymo [53], provide annotated sequence data which has led to the proposal of multi-frame based 3D object detection [26, 9] and the incorporation of temporal information into the deep learning models proposed by El Sallab et al. [91] and McCrae and Zakhori [48] who used convolution-LSTMs [133]. Similar works [16, 38] emphasise utilising temporal information to address the effect of occlusion during estimation of box parameters and show the improvement in object detection and localisation with temporal inputs.

Koh, Junho, et al. [15] used detector and tracker modules to work in tandem to generate spatio-temporal representations of camera and lidar data to perform detection and tracking. Previously 3D CNNs for multi-frame BEV features were used by Luo, Yang and Urtasun [95] while a temporal fusion model (TFM) was used in [10] to fuse the features extracted from raw image and lidar point cloud data through a Frustum PointNet [97]. To aggregate the scene-level temporal features from a multiple frame reference, Yin, Zhou, and Krahenbuhl [26] employed motion cues between the successive frames, while Huang et al. [38]

propose an RNN-based method. Since this method utilises the whole point cloud data, motion compensation was required to align features in time, and Erçelik, Yurtsever & Knoll [9] focus on the need for spatial alignment of features between successive frames while aggregating the features. In contrast, Kumar and Al-Stouhi [43] used a non-linear attention mechanism to build the spatio-temporal relation between multiple frames. Finally, object detection and motion forecasting are jointly performed in Laddha et al. [16] by fusing range view and BEV representation for spatial and temporal feature learning. A comprehensive review and comparison of recent state-of-the-art 3D object detection technology approaches are presented in [3, 57, 63]. Challenges involved while handling multimodal information and methodologies to fuse multiple modality information to complement individual sensor capabilities effectively are discussed by Di Feng et. al. in [61].

Table 2.1: Comparison of 3D Object detection methods evaluated on KITTI 3D Object detection benchmark

Method	Type	Car			Pedestrian			Cyclist		
		Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
BirdNet+ [29]	L	70.14	51.85	50.03	37.99	31.46	29.46	67.38	47.72	42.89
VoxelNet [107]	L	89.60	84.81	78.57	-	-	-	-	-	-
SECOND [102]	L	89.96	87.07	79.66	-	-	-	-	-	-
PointPillars [70]	L	82.58	74.31	68.99	51.45	41.92	38.89	77.10	58.65	51.92
PointRCNN [80]	L	86.96	75.64	70.70	47.98	39.37	36.01	74.96	58.82	52.53
TANet [47]	L	84.39	75.94	68.82	53.72	44.34	40.49	75.70	59.44	52.53
SegVoxelNet [60]	L	86.04	76.13	70.76	-	-	-	-	-	-
Voxel-FPN [42]	L	85.64	76.70	69.44	-	-	-	-	-	-
Fast Point R-CNN [67]	L	85.29	77.40	70.24	-	-	-	-	-	-
3DSSD [59]	L	88.36	79.57	74.55	54.64	44.27	40.23	82.48	61.10	56.93
PV-RCNN [52]	L	90.25	81.43	76.82	52.17	43.29	40.29	78.60	63.71	57.65
3D IoU-Net [44]	L	87.96	79.03	72.78	-	-	-	-	-	-
STD [86]	L	87.95	79.91	75.09	53.29	42.27	38.35	78.69	61.59	55.30
F-ConvNet [85]	LI	87.36	76.39	66.96	52.16	43.38	38.80	81.98	65.07	56.54
MV3D [65]	LI	74.97	63.63	54.00	-	-	-	-	-	-
MLOD [40]	LI	77.24	67.76	62.05	47.58	37.47	35.07	68.81	49.43	42.84
F-PointNet [97]	LI	82.19	69.79	60.59	50.53	42.15	38.08	72.27	56.12	49.01
AVOD-FPN [65]	LI	77.63	63.78	55.89	50.46	42.27	39.04	63.76	50.55	44.93
MMF [71]	LI	88.46	77.43	70.22	-	-	-	-	-	-
MVX-Net [83]	LI	83.20	72.70	65.20	-	-	-	-	-	-
PointPainting	LI	82.11	71.70	67.08	50.32	40.97	37.84	77.63	63.78	55.89
Point Fusion [101]	LI	77.92	63.00	53.27	33.36	28.04	23.38	49.34	29.42	26.98
EPNet [39]	LI	88.76	78.65	78.32	66.74	59.29	54.82	83.88	65.50	62.70

Table. 2.1, presents the comparison results of the state-of-the-art methods on the KITTI object detection test set. These results depict the intent of the researcher working on 3D object detection to make it more reliable and improve the accuracy of the 3D detection. Although the newer methods achieve a marginal improvement in the accuracy from prior approaches,

the results are still valuable for understanding the scope for further improvements and the need to reduce the gap between the SOTA results and the acceptable detection accuracy (maybe $\geq 99.9\%$) for the autonomous vehicles. Further, the methods proposed in [15, 10, 43] are the most recent and arise from the availability of large sequenced annotated datasets and demonstrate the advantage of considering multi-frame reference and temporal information to improve the performance of the 3D object detector. However, there still exists scope to explore the strategies further to make use of the attention mechanism while learning temporal features from the sequences data, which could further improve the detection accuracy during occlusion, change in perception or appearance of the object.

2.3 Dataset

In this section, a brief review of popular perception datasets employed for the task related to autonomous driving is presented. A comparison of AV datasets used for perception-related tasks is provided in Table.2.2. These datasets are selected based on the availability of annotations for both lidar and image data for 3D object detection.

Table 2.2: Comparison of autonomous vehicle dataset

Dataset	Year	Annotated frames	Classes	RGB Imgs	Lidar	Radar	3D boxes	Sensors (lidar & camera)
KITTI [138]	2012	15k	8	15k	15k	0	200k	1 x Velodyne HDL-64, 3 x stereo cameras
ApolloScape [69]	2018	144k	8-32	144k	0	0	70k	1 x VMX-1HA 2 x Riegl
NuScenes [32]	2019	40k	23	1.4M	400k	1.3M	1.4M	1 x Velodyne HDL-32, 6 x cameras
H3D [77]	2019	27k	8	8.3k	27k	0	1.1M	1 x Velodyne HDL-64S2, 3 x cameras
Lyft [37]	2019	323k	9	46k	46k	0	1.3M	1 x 64-beam roof, 2 x 40 beam bumper, 7 x cameras
A2D2 [35]	2019	12k	14	12k	12k		-	5 x Velodyne VLP-16, 6 x cameras
Waymo Open [53]	2019	200k	4	1M	200k	0	12M	1 x 360 75m range, 4 x HoneyComb 20m range, 5 x cameras
A*3D [50]	2019	39k	7	39k	39k	0	230k	1 x Velodyne HDL-64ES3, 2 x cameras
CADC [20]	2020	7k	6	263k	32k		-	1 x Velodyne VLP-32C, 8 x cameras
All in One Drive [19]	2020	250k	-	250k	-		26 M	3 x lidar, 5 x cameras (synthetic data)
Once [17]	2021	16k	5	7M	1M		417k	1 x 40 beam roof lidar, 7x cameras

2.3.1 KITTI Benchmark Suite

The KITTI 3D object detection benchmark [139, 138] consists of 7,481 training images and 7,518 test images as well as the corresponding point clouds, comprising a total of 80,256 labelled objects comprising eight classes: 'Car', 'Van', 'Truck', 'Pedestrian', 'Person (sitting)', 'Cyclist', 'Tram' and 'Misc'. The platform is equipped with four high-resolution video cameras and a Velodyne laser scanner. It is comprised of 389 stereo and optical flow image pairs, stereo visual odometry sequences of 39.2 km length, and more than 200K 3D object annotations captured in cluttered scenarios (up to 15 cars and 30 pedestrians are visible per image) by driving around the mid-size city of Karlsruhe, Germany, in rural areas, and on highways. The dataset was focused on developing novel, challenging benchmarks for the tasks of stereo, optical flow, visual odometry / SLAM and 3D object detection.

2.3.2 The apolloscape

ApolloScape [69, 73] is public dataset from real scenes captured under various weather conditions, day time across four regions in China. The data acquisition platform is equipped with two VUX-1HA laser scanners, six cameras which include two high-resolution frontal cameras, and a measuring head with IMU/GNSS. The whole system is internally calibrated and synchronized and is mounted on the top of a mid-size SUV. The dataset contains annotation including holistic semantic dense point cloud for each site, stereo, per-pixel semantic labelling, lanemark labelling, instance segmentation, 3D car instance, high accurate location for every frame in various driving videos from multiple sites, cities and day times (morning, noon, night). The dataset is larger and richer compared to KITTI [138] with 100+ hours of stereo driving, 20+ driving sites containing dense semantic 3D point cloud, 160K+ and 144K+ annotated images with 35 classes for lanemark and per-pixel per-frame semantic labels, 90K annoated images with 8 classes for semantic 2D instances segmentation, and 2D car keypoints and 3D car instance labelling for 70K cars.

2.3.3 H3D: Honda Research Institute 3D Dataset

Honda Research Institute 3D Dataset (H3D) [77], a large-scale full-surround 3D multi-object detection and tracking dataset collected in 4 urban areas in the San Francisco Bay Area.

H3D comprises 160 crowded and highly interactive traffic scenes with one million labelled instances in 27,721 frames. H3D was the first dataset to provide full-surround 3D multi-object detection and tracking in crowded urban scenes comprising 1,071,302 3D bounding box labels of 8 common traffic participants, including Cars, Pedestrian, Cyclist, Truck, Misc, Animals, Motorcyclist, and Bus. The data acquisition platform is equipped with three high-resolution Grasshopper3 video cameras, one Velodyne HDL-64E S2 3D LiDAR, and Automotive Dynamic Motion Analyzer (ADMA) with DGPS output gyros, accelerometers and GPS sensors.

2.3.4 nuScenes: A multimodal dataset for autonomous driving

nuTonomy scenes (nuScenes) [32], was the first dataset to carry the fully autonomous vehicle sensor suite consisting of six cameras, five radar sensors, a lidar scanner, and GPS-IMU sensor, all with a full 360-degree field of view. The nuScenes comprises 1000 scenes, each 20s long and fully annotated with 3D bounding boxes for 23 classes and eight attributes. It has 7x as many annotations and 100x as many images as the pioneering KITTI dataset. The data was collected by driving vehicles in Boston (Seaport and South Boston) and Singapore (One North, Holland Village and Queenstown). These two cities are known for their dense traffic and highly challenging driving situations. The data also covered the diversity across locations regarding vegetation, buildings, vehicles, road markings and right versus left-hand traffic.

2.3.5 Lyft Prediction Dataset:

Lyft prediction dataset [37, 14] is a large self-driving dataset for motion prediction to date, with over 1,000 hours of data collected by a fleet of 20 autonomous vehicles equipped with seven cameras, 3 LiDARs, and five radars driven along a fixed route in Palo Alto, California. It consists of 170k scenes, each 25 seconds long, capturing the movement of the self-driving vehicle and nearby vehicles, cyclists, and pedestrians over time. In addition, the dataset also provided a high-definition (HD) semantic map of the area, capturing the road rules and lane geometry and other traffic elements counting over 15,000 human annotations, including

8,500 lane segments. Further, a high-resolution aerial picture of the area was also provided that can be used to develop semantic-map-free solutions for the prediction task.

2.3.6 Audi Autonomous Driving Dataset (A2D2)

The Autonomous Driving Dataset (A2D2) [35] dataset consists of simultaneously recorded images and 3D point clouds, together with 3D bounding boxes, semantic segmentation, instance segmentation, and other data extracted from the test vehicle. The sensor suite comprises six cameras and five lidar units, providing full 360-degree coverage. The recorded data is time synchronized and mutually registered. Annotations are for non-sequential frames: 41,277 frames with semantic segmentation image and point cloud labels, of which 12,497 frames also have 3D bounding box annotations for objects within the field of view of the front camera. In addition, 392,556 sequential frames of unannotated sensor data are provided for recordings in three cities in the south of Germany.

2.3.7 Waymo Dataset

The Waymo Open dataset [53] was initially released in 2019 and consisted of 1000 scenes for training and validation and 150 scenes for testing. Each of the scenes spans the 20s containing around 12 million lidar 3D box annotations and around 12 million camera 2D box annotations, giving rise to around 113k lidar object tracks and around 250k camera image tracks. The data acquisition platform consists of five LiDAR sensors and five high-resolution pinhole cameras captured across various conditions in multiple cities, namely San Francisco, Phoenix, and Mountain View, with large geographic coverage within each city. Later in 2021, waymo expanded the dataset [11] to facilitate motion forecasting research by providing a large-scale, diverse dataset with specific annotations for interacting objects to promote the development of models to predict interactive behaviours jointly. The dataset was collected by mining for interesting interactions between vehicles, pedestrians, and cyclists across six cities within the United States, containing 100,000 scenes, each 20 seconds long.

2.3.8 A*3D Dataset

A*3D Dataset [50] is high-density images, heavy occlusions, and a large number of night-time frames to address the gaps in the existing datasets to push the boundaries of tasks in autonomous driving research to more challenging, highly diverse environments. It consists of 230K 3D object annotations for seven classes, including vehicles (Car, Van, Bus, Truck), pedestrians, cyclists, and motorcyclists in 39,179 LiDAR point cloud frames and corresponding frontal-facing RGB images captured at different times (day, night) and weathers (sun, cloud, rain). The autonomous vehicle setup is equipped with a rotating Velodyne lidar and two colour PointGrey Chameleon3 cameras on either side of the lidar. The data collection covers entire Singapore, including highways, neighbourhood roads, tunnels, urban, suburban, industrial, HDB car parks, coastline, etc., with varying weather conditions to handle diverse environments for autonomous driving tasks.

2.3.9 Canadian adverse driving conditions (CADC) dataset

The Canadian Adverse Driving Conditions (CADC) dataset [20] was collected with an autonomous vehicle platform equipped with eight camera sensors, a lidar scanner and a GNSS+INS system in Waterloo, Canada, during winter. It consists of 3D annotation of 7,000 lidar and 56,000 image frames containing 300K objects that include classes: Car, Truck, Bus, Bicycle, Horse and Buggy, Pedestrian, Pedestrian with Object, Animal, Garbage Container on Wheels, and Traffic Guidance Object and has attributes [Parked, Stopped, Moving] in every frame.

2.3.10 All-In-One Drive

All-In-One Drive (AIODrive) dataset [19] is a large-scale synthetic dataset with high-density long-range point clouds. The dataset provides comprehensive sensors, annotations and environmental variations to target perception-related problems like 3D segmentation, radar sensing, and large-scale training under one umbrella. The dataset also facilitates annotation for perception-related tasks like detection, tracking, trajectory prediction, segmentation, and depth estimation. It consists of 250 sequences with 250K annotated images frames containing 26M 2D and 3D bounding boxes for the objects generated using a simulator by

employing eight sensor modalities: five high-resolution RGB cameras (one stereo pair); five depth cameras, 1,000 meter range lidar at multiple levels of density (up to 1M points), 1,000 meter range SPAD-lidar, Radar, IMU, and GPS. The data can be generated by varying the simulator's parameters during simulation, which will cover rare driving scenarios such as adverse weather and lighting, crowded scenes, high-speed driving, violation of traffic rules, and accidents.

2.3.11 Once dataset

The ONCE (One millionN sCenEs) dataset [17] for 3D object detection in the autonomous driving scenario consists of 1 million lidar scenes and 7 million corresponding camera images. The data is selected from 144 driving hours collected across multiple cities in China, traversing different areas, periods (morning, noon, afternoon, night) and weather conditions (sunny, cloudy, rainy). The data is collected using one lidar scanner and seven camera sensors for a 360° field of view. The lidar and camera sensors are synchronised and enabled with a fusion of multiple sensors and scenes. The dataset covers 16K annotated scenes with 3D ground truth boxes of 5 categories (car, bus, truck, pedestrian and cyclist), leading to 417K 3D bounding boxes. In addition, around 769K 2D annotations are generated for camera images by projecting 3D boxes into image planes.

2.4 Evaluation Metric

An evaluation benchmark aims to get meaningful measures of a system's ability to perform object detection tasks. Metrics include the number of correctly detected (True Positives) objects, falsely detected (False Positives) objects, or misdirected (False Negatives), Objects and Backgrounds (True Negatives). Generally, all metrics that contain true negatives are ignored in object detection because there would be too many true negatives, which makes the metric difficult to use. Further, the ground truth does not have boxes annotated for true negatives. Popular benchmarks for 3D object detection include KITTI [139] and nuScenes [32]. A description of different metrics used in this benchmark is presented here.

2.4.1 KITTI evaluation metric

Precision: Precision is defined as the fraction of detected objects that are correct.

$$\text{Precision} = \frac{TP}{TP + FP} \quad (2.1)$$

Recall: The Recall is defined as the fraction of objects that were correctly detected among all the objects that should have been detected.

$$\text{Recall} = \frac{TP}{TP + FN} \quad (2.2)$$

Intersection over Union (IoU): This measures how good our prediction is in the object detector with the ground truth. IoU is only used as an evaluation metric represented by the mean average precision (mAP). The IoU is calculated as the overlap of the predicted bounding box and the ground truth bounding box.

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} \quad (2.3)$$

Precision-Recall Curve: plot obtained by varying the object detector confidence value for each of the object class Mean Average Precision (mAP): The Average Precision (AP) numerically summarizes the shape of the precision-recall curve and is defined as the mean precision over N equally spaced discrete levels of recall $\{r_n\}_{n=1}^{n=N}$, which is given as

$$\text{AP} = \frac{1}{N} \sum_{n=1}^N \max P(r_i); n \leq i \leq N \quad (2.4)$$

where $p(r_i)$ is the calculated precision at recall r_i . The value of N is set to 11. After calculating the AP of each class, the mean Average Precision (mAP) is calculated for overall performance evaluation.

Average Orientation Similarity: Average orientation similarity (AOS) measures the performance of jointly 2D detection and 3D orientation by weighting the 2D average precision score on the image plane with cosine similarity between the predicted and ground-truth orientations. AOS is define as:

$$\text{AOS} = \frac{1}{11} \sum_{r \in 0, 0.1, \dots, 1} \max s(\tilde{r}); \tilde{r} : \tilde{r} \geq r \quad (2.5)$$

where $r = TP \setminus (TP + FN)$ is the recall and the orientation similarity $s \in [0, 1]$ is normalized by the cosine similarity defined as

$$s(r) = \frac{1}{|D(r)|} \sum_{i \in D(r)} \frac{1 + \cos \Delta_{\theta}^{(i)}}{2} \delta_i \quad (2.6)$$

where $D(r)$ is the set of all object detection at recall rate r , $\Delta_{\theta}^{(i)}$ is the angle difference between estimated and ground truth orientation of detection ‘i’ and δ_i defines if a detection ‘i’ has been assigned to a ground truth bounding box or not.

2.4.2 nuScenes evaluation metric

In the nuScenes benchmark, average precision is defined as a match by thresholding the 2D center distance ‘d’ on the ground plane instead of the intersection over union (IOU). This is done to handle objects with small footprints, like pedestrians and bikes, where a small translation error gives zero IOU value. The final mAP is computed as average over matching thresholds of $D = 0.5, 1, 2, 4$ meters and the set of classes C :

$$\text{mAP} = \frac{1}{|C||D|} \sum_{c \in C} \sum_{d \in D} \text{AP}_{c,d} \quad (2.7)$$

True Positive Metric: A set of five True positive metrics are proposed in nuScenes benchmark [32] to quantify the quality of the detections. All these metrics are positive scalar and are calculated using $d = 2\text{m}$ center distance during matching. The metrics are measured in their native units and are defined as follows:

- **Average Translation Error (ATE)** is the Euclidean center distance in 2D (units in meters).
- **Average Scale Error (ASE)** is the 3D intersection over union (IOU) after aligning orientation and translation ($1 - \text{IOU}$).

- **Average Orientation Error (AOE)** is the smallest yaw angle difference between prediction and ground truth (radians).
- **Average Velocity Error (AVE)** is the absolute velocity error as the L2 norm of the velocity differences in 2D (m/s).
- **Average Attribute Error (AAE)** is defined as 1 minus attribute classification accuracy ($1 - acc$). attributes: parked, stopped, moving, with rider, without rider, standing, sitting, walking

For each TP metric the final mean TP metric (mTP) over all classes is computed as follows:

$$mTP = \frac{1}{|C|} \sum_{c \in C} TP_c \quad (2.8)$$

nuScenes Detection Score (NDS): Only mAP metrics can not capture all aspects of the nuScenes detection tasks, like velocity and attribute estimation. Further, it couples location, size and orientation estimates. nuScenes detection score (NDS) [32] was employed to handle errors obtained while computing on different object classes and is defined as follows.

$$NDS = \frac{1}{10} [5mAP + \sum_{mTP \in TP} (1 - \min(1, mTP))] \quad (2.9)$$

Where mAP is mean Average Precision, mTP is set of five mean True Positive Metric, Half of NDS is thus based on the detection performance while the other half quantifies the quality of the detections in terms of box location, size, orientation, attributes, and velocity. Since mAVE, mAOE and mATE can be larger than '1', each metric is normalised between '0' and '1'.

2.5 Summary

An overview of the related work on object detection frameworks employing various sensing modalities and representations is presented in this chapter. The methods listed in this chapter are initially classified based on the number of frames employed by the detection

framework into single-frame and multi-frame references. Further, based on the data representation, these methods are categorised into image-based, point cloud-based, and multi-modal fusion-based methods. This chapter also highlights the available public datasets and benchmark evaluation metrics used by the researchers in this project to develop solutions for the perception-related task for autonomous vehicles. Recent works [83, 51, 54, 39, 23, 18] have demonstrated state-of-the-art results using fusion-based techniques where multiple modalities are combined to train the model to utilise individual modalities' characteristics for improved detection accuracy. However, with the availability of new large-scale datasets like nuScenes, waymo etc., further exploration of multimodal data and fusion strategies remains an exciting field of research. In addition, these datasets cover certain instances [61] where the state-of-the-art fails even after achieving good performance on the benchmark evaluation dataset. The limited research available in the literature that employs multiple-frame reference [26, 40, 16, 38] for extracting the temporal information for 3D object detection paved the way for designing the pilot study to understand the detection framework are, presented in Chapter 3 and further investigations are detailed in Chapter 4.

Chapter 3

Preliminary Work

This chapter presents the preliminary work conducted to implement and train a deep learning framework for 2D object detection and localization on vehicle datasets. The work was part of the deliverable for two projects, namely VI-DAS ¹ (Vision Inspired Driver Assistance System) and Huawei-Insight Autonomous Vehicle ². Section 3.1 provides a brief overview of the You Only Look Once (YOLO), single shot multi-box detection (SSD) and faster-RCNN detectors used for detection and localization of 'car', 'pedestrian' and 'cyclist' in the KITTI 2D object dataset. The estimated object locations were later fed to the multiple-object tracking module to keep track of the objects in the scene and to handle occlusion or detector failure. The outcomes and experience gained with the 2D detection and tracking task cultivated the idea of employing visual saliency in the framework presented in Section 3.1.2. The need for an object detector able to localize and estimate the depth information in the autonomous vehicle led to subsequent pilot studies for understanding 3D object detection, and methodologies to incorporate temporal cues are presented in Section 3.2.2. Section 3.2.3 briefly describes the baseline architecture employed for investigations and the selection of probable baseline architecture appropriate for exploring fusion techniques for fusing multiple modalities. The work presented in Section 3.1.1 and Section 3.1.2 are published in Venkatesh, Hu, et al. (2019) [84] and Hu, Venkatesh, et al. (2020) [13].

¹This work has been partly funded by the EU H2020 Projects VI-DAS (grant number 690772) link: <http://www.vi-das.eu/>

²This work has been sponsored by Huawei HIRP and Science Foundation Ireland under Grant Number SFI/12/RC/2289 and SFI/16/SP/3804.

3.1 2D Object Detection

3.1.1 Single and Two Stage Object Detector

Pilot Study-1:

The Pilot Study-1 presents the work carried out in the VI-DAS project. One of the project's key requirements was to develop a pipeline to detect and track the objects in the image sequence acquired from four wide-angle camera sensors mounted on a test vehicle to achieve 360°surround view sensing. The initial prototype was designed based on the tracking-by-detection framework for a single camera with a maximum computation time of not more than 30 milliseconds. The criterion for choosing a detector module was to yield the detection results with minimum computation time and reasonable detection accuracy. Single-stage detectors such as You Only Look Once (YOLO), and Single Shot Multi-Box Detection (SSD) architectures were the most favourable options available during the prototype development phase. The YOLO versions namely YOLOv2 [117], YOLOv3 [99] and YOLOv5 [21, 25, 22] architectures were explored as the detector module with Kalman-filter based SORT [120] algorithms for tracking the objects in the image sequence.

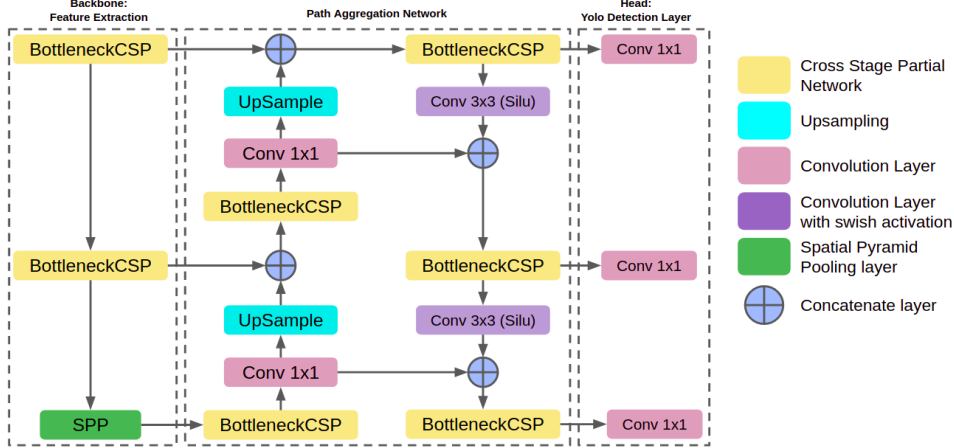


Figure 3.1: YOLO-V5 network architecture [21, 25, 12]

The most recent YOLOv5 architecture is shown in Fig 3.1 and other architecture considered in this work can be found in the articles [117, 99]. The YOLOv2 was implemented using Keras and YOLOv5 in pytorch with both having Tensorflow as backend on an Intel i7 processor with a single GPU (GeForce GTX TITAN X, 12GB RAM). The networks



Figure 3.2: Object detection using YOLOv5, left column represents input and right column represents corresponding detection results on KITTI Dataset

were trained on the KITTI dataset with pre-trained MS-COCO weights. During the prototype development, only 'car', 'pedestrian' and 'cyclist' classes were considered for performance evaluation. As in [117, 21], all input image sizes and annotations were scaled to 416x416. Both the model were trained for 200 epochs using adam optimizer [106] with learning rate=1e-3, weight decay=1e-4, batchsize=8 and the learning rate was reduced by 1/10 after every 50 epochs. Overall Mean Average Precision (mAP) of $\approx 58.44\%$ and $\approx 66.5\%$ are obtained for YOLOv2 and YOLOv5 respectively. The corresponding individual class mAPs obtained using YOLOv2 and YOLOv5 are depicted in Table 3.1. The performance of YOLOv5 architecture has a better mAP of $\approx 8\%$ in comparison to the YOLOv2 on validation dataset. Hence, YOLOv5 was used to generate the prior region proposals during the exploratory trial discussed in Section 4.3.1 of Chapter 4. The detection results obtained using YOLOv5 on the KITTI validation set are presented in Fig. 3.2.

Table 3.1: YOLOv2 and YOLOv5 detection accuracy on KITTI 2D object detection validation set

Model	Car	Pedestrian	Cyclist	Overall mAP
YOLOv2	81.25	42.41	51.63	58.44
YOLOv5	81.80	51.30	66.20	66.5

The code repository of YOLOv5 was obtained from the GitHub repository of Ultralytics [22] and trained on the KITTI dataset [139]. The network is initialised with the weighted trained on MS-COCO [136] dataset. A total of nine anchor sizes are estimated using the k-mean clustering method, and each receptive field is initialised with three anchor sizes as depicted in Table 3.2.

Table 3.2: Anchors generated using k-mean clustering, RF represents receptive field

small RF	medium RF	large RF
16x13	12x32	26x19
51x21	54x38	31x77
94x43	112x76	171x91

Pilot Study-2:

The Pilot Study-2 presents the adoption of SSD [123] and Faster-RCNN [132] detectors to explore the influence of visual saliency on detection and tracking tasks in the Huawei-Insight Autonomous Vehicle project. The tracking by detection framework was considered in designing the pipeline, and initial experiments involving the SSD and Faster-RCNN detectors were prepared to validate the idea of using saliency on detection. Hence, SSD architecture is used for object detection, and a pre-trained salGAN [113] model is used for extracting the salient regions in the images, which are then passed to the detector as input to prior locate the objects in the region. Later, the idea of using salient regions was adopted into the Faster-RCNN architecture.

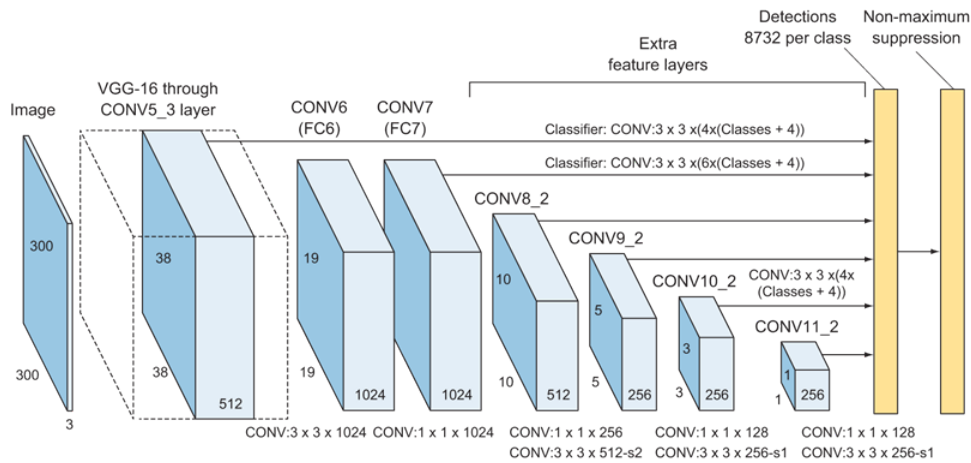


Figure 3.3: The SSD network architecture [123]

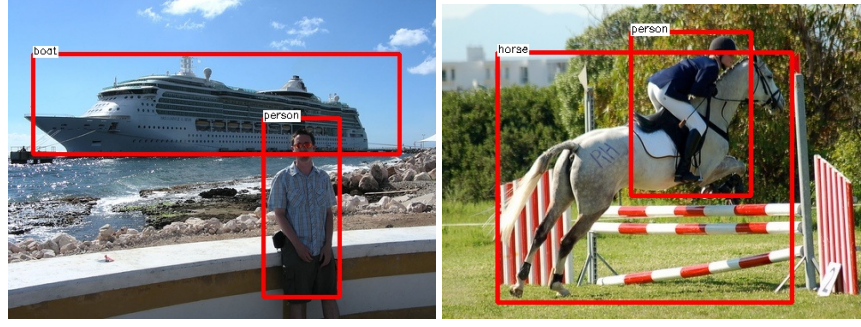


Figure 3.4: Sample output on PASCAL VOC dataset

The SSD architecture is presented in Fig. 3.3. The model was implemented using Keras with a Tensorflow backend on an Intel i7 processor with a single GPU (GeForce GTX TITAN X, 12GB RAM) and trained on the PASCAL VOC 2007 [142] dataset with 20 object classes. The network was initialised with pre-trained MS-COCO weights and all the input images and annotations were scaled to 300x300 during training and testing. The model was trained for 200 epochs using an SGD optimiser with learning rate=1e-2, momentum=0.9, batchsize=8, and a learning rate scheduler was employed to adjust the learning rate during training. An overall mAP of **76.19%** was obtained, and the sample results are shown in Fig. 3.4. Further, details on the individual class mAP can be found in the paper [84].

Table 3.3: Detection results of SSD and with saliency [84] on KITTI 2D object detection training set. In legend: px represents pixel

	All Sizes		> 15 ² px		< 75 ² px	
	SSD	sal-SSD	SSD	sal-SSD	SSD	sal-SSD
Person	8.00	15.39	8.15	15.68	0.13	7.54
Cars	37.37	52.40	36.53	52.62	2.56	40.50
mAP	22.19	33.90	22.34	34.15	1.34	24.02

The pipeline consists of SalGAN [113] implemented in Pytorch, which was pre-trained on the SALICON [130] dataset with no fine-tuning, along with the trained SSD model to investigate the influence of saliency on object detection. The results demonstrated the positive impact of using saliency as a guiding entity for object detection. The pipeline with saliency guidance significantly improved the SSD benchmark by almost **12%**, and further breakdown of individual object class performances are depicted in Table 3.3. The mAP of the 'person' was increased by **c.8%**, and for the 'car', it increased by **c.16%**. From the obtained results, the objects with an area of more than 15×15 pixels have slightly better mAP compared to

the objects of all sizes. However, for the objects with an area smaller than 75×75 pixels, the pipeline with saliency guidance significantly improves the SSD benchmark by almost **24%**. The experiments results on the KITTI dataset shown in Fig. 3.5 demonstrate that the performance of an off-the-shelf SSD detector was improved significantly when the detection is guided by saliency and particularly for smaller objects.



Figure 3.5: Detection results on KITTI dataset, 1st row:ground truth images, 2nd row: SSD detection results and 3rd row: SSD detection results with saliency guided [84]

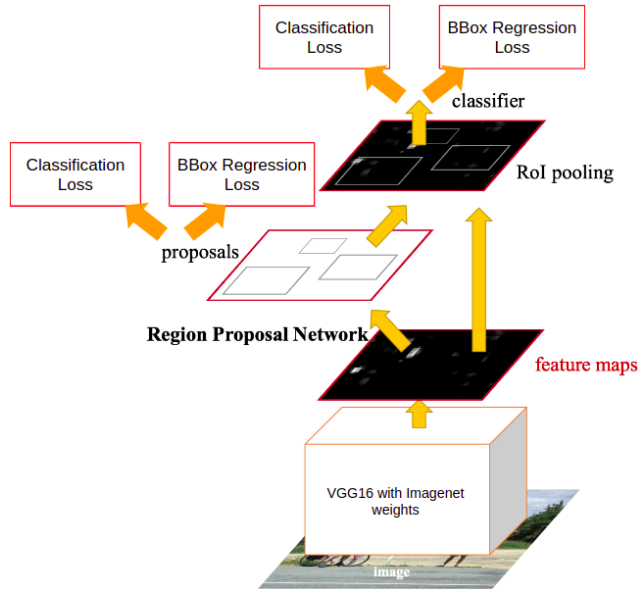


Figure 3.6: The Faster-RCNN network architecture [132]

The experiment's outcome motivated to adopt a two-stage detector: Faster-RCNN, to

explore the strategies to incorporate visual attention (saliency) for object detection and tracking. During this exploratory phase of a two-stage detector, replication of Faster-RCNN architecture, as shown in Fig. 3.6 was done using PyTorch with TensorFlow as backend on an Intel i7 processor with a single GPU (GeForce GTX TITAN X, 12GB RAM). The network was trained on the KITTI dataset with three object classes, namely 'car', 'pedestrian' and 'cyclist'. A VGG16 model pre-trained on the Imagenet dataset was used as the backbone network to extract image features with all the input image sizes and annotations scaled w.r.t shortest side of the image during training and testing. All models are trained for 30 epochs, and the gradients are updated using standard stochastic gradient descent (SGD) with momentum=0.9, the learning rate was set to 0.001, and the value is decreased by 10 times for every 10 epochs, and the batch size of 8 was used. All the results in detection are reported with average precision (AP) using an intersection over union (IoU) value of 0.7. Anchor boxes are generated using a size of 4, 8, 16 and 32 and ratios of [0.5, 1, 2]. Overall mAP of **69.41%** was obtained, and the individual class mAP is depicted in Table 3.4.

Table 3.4: Faster-RCNN detection accuracy on KITTI 2D object detection validation set.
Legend: 'Easy', 'Mod.' 'Hard' and 'mAP' represent easy, moderate and hard case and mean Average Precision respectively

Class	Easy	Mod.	Hard	mAP
Car	90.54	90.09	80.83	80.16
Cyclist	70.50	59.62	58.15	57.31
Pedestrian	67.45	58.15	50.28	49.68
Overall	80.32	76.66	71.59	69.41

3.1.2 Multiple Object Tracking-by-Detection

This section presents a brief overview of the contribution made by the author presented in [13] in developing an object tracking module guided by saliency to mitigate the loss of localization information due to occlusion and detector failure in the scene. A modified sequential Monte Carlo probability hypothesis density (PHD) filter utilising the attention maps (saliency) generated by the detection module was used. Fig. 3.7. and Fig. 3.9 presents sample examples of Multiple Object Tracking (MOT) using an objectiveness mask on KITTI [139] and DeTRAC [56] dataset respectively. The attention map assists the tracker

in correcting the predicted position of the targets during detection failure. The tracking-based metrics [143, 124] (also called object-based metrics) were considered for the evaluation of the object tracking module. The metrics take the identity and the complete trajectory of each object separately over the test sequence and compare the ground-truth tracks with the predicted tracks based on best correspondence. Then, based on these correspondences, various error rates and performance metrics are computed. The use of visual attention showed a considerable improvement of $\approx 8\%$ in object detection, which effectively increased tracking performance by $\approx 4\%$ on the KITTI dataset and the results are depicted in Table 3.5.

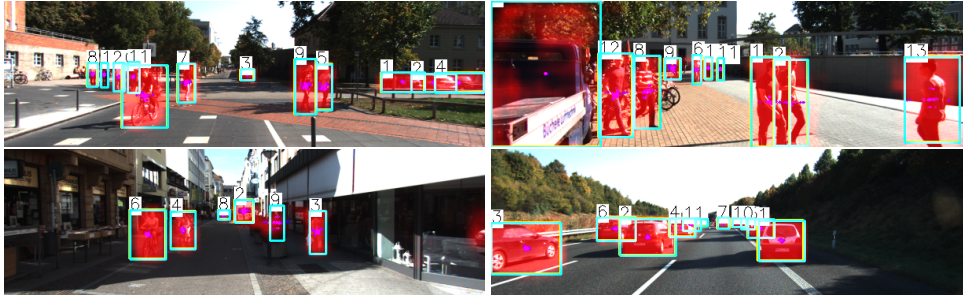


Figure 3.7: Examples of Multiple Object Tracking using objectiveness mask on KITTI Dataset

In the tracking-by-detection framework, the tracker receives the position and bounding box of the targets in the scene from the detection module. The main objective of the tracker module is to estimate the position of the object in the scene during occlusion and detection uncertainties. The probability hypothesis density filter (PHD) [145] is the adaptation of a random finite set for multi-target tracking [137] to handle this uncertainty. Depending on the kind of complexity (linear or non-linear) in the target, PHD filters are implemented in two popular schemes. Firstly, when the target dynamics are linear and can be assumed to be a Gaussian process, then the Gaussian mixture (GM-PHD) filter [144] is employed. Secondly, if the dynamics are highly non-linear and non-Gaussian, the Sequential Monte Carlo or particle-PHD filter [147] is used.

In the standard particle-PHD filter, it is difficult to guide the particles to the region of interest as they are scattered, and due to the absence of a state correction step, the error between the actual measurement and estimated measurements is not minimized and can lead to failure in posterior estimation. Similar to the approach mentioned in [108], kalman gain (KG) is used to minimize the error between the estimated and actual measurements. This

correction mechanism will guide validated particles in the particle-PHD filter to converge towards the region of a higher likelihood of the observed measurements. This mechanism helps the tracker in approximating the posterior estimations at each time step. In the proposed filter, Kalman gain, along with the visual cues, is used to compute the inter-frame displacement of the objects to facilitate the particle distribution and re-sampling process. The modified particle-PHD filter is summarised as follows:

- **Initialisation:**

- At time $k=0$, instead of using Bernoulli and Poisson processes of object birth process to initialise the PHD $D_{k|k}$, multi-peak Gaussian distribution by number of particles with associated randomised weights is adapted
- At the time $k \geq 1$, particle approximation of the density function and Kalman gain parameters are obtained using the previous prediction and updated results.

- **Particle State Prediction and weight computation:**

- State estimation is performed based on the weighted IoU and distance metric computed on the temporal histogram extracted by utilising the track history and visual attention cues.
- After computing the prior state of the objects, the particle with the maximum weight is taken as the final predicted position of the target

- **Particle State Update:**

- The same state update step incorporating IoU and histogram distance metric is followed.
- Kalman filter parameters are also updated.

- **Particle Resampling:** the motion cues during the sampling process are considered, which assists the PHD filter in localizing the density function along the target's motion. Residual resampling strategy is applied in the North, South, West and East directions of the particle position, with most of the particles distributed in the direction of the target's motion as shown in Fig. 3.8.

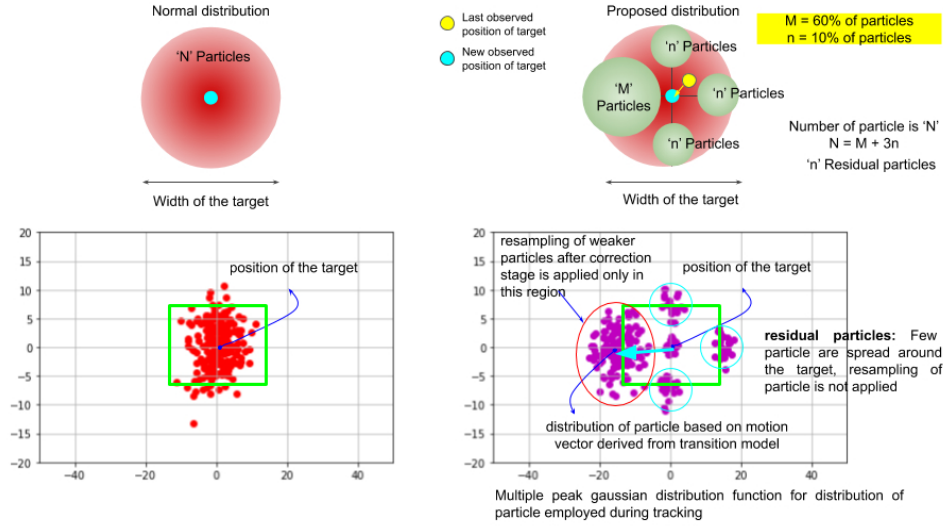


Figure 3.8: Normal Distribution (left), Multi-peak Gaussian particle distribution (right) based on object motion

- **Refine and update using visual attention cues:** In this stage, an additional correction mechanism is applied along with the Kalman correction. The predicted box position is scanned for the presence of any attention map for retaining prediction during the detection/prediction failure or when the object is leaving the field of view of the camera. The intersection area of the attention map region over the predicted box position gives the occupancy. If the computed occupancy measure is less than 30% of the area of the predicted box, then the prediction is ignored and a correction is made based on the occupancy of the attention cues. The density function and Kalman parameters are based on the corrected target position, and this function is used for re-distributing the particles in the next frame.

Data Association of the detected objects between the frames is obtained using the Hungarian assignment algorithm with two equally weighted measures: Intersection over Union (IoU) of the bounding box and an HSV (Hue Saturation and Value) colour histogram of the objects. The histogram is generated by concatenating the Hue channel with 50 bin normalisation and the saturation channel with 60 bin normalisation. For data association, the Bhattacharyya distance is computed between previous frame track results and the detected objects in the current frame. Then, the detection is assigned to the track with a minimum cost value. When the detection module fails to detect a previously detected object for two

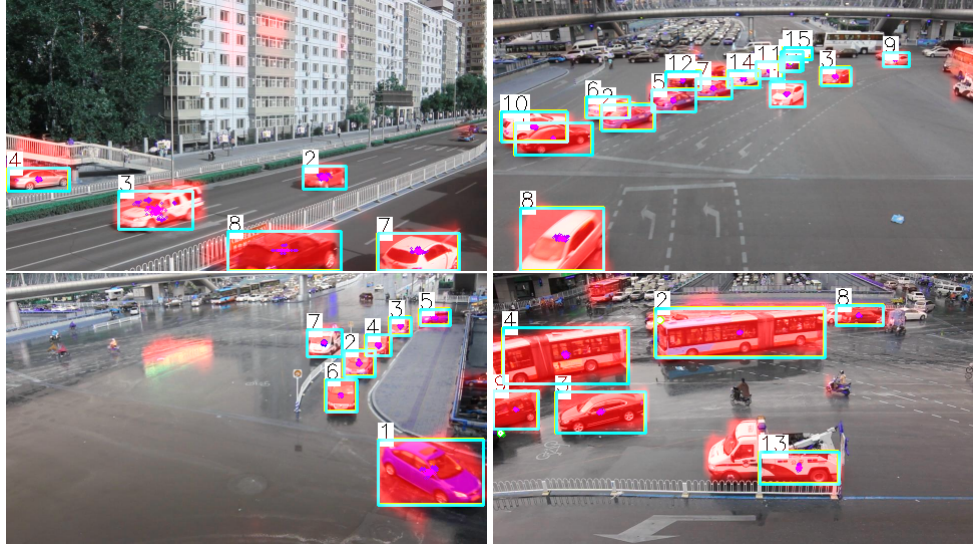


Figure 3.9: Examples of Multiple Object Tracking using objectiveness mask on DeTRAC Dataset

consecutive frames, the tracker will terminate the track.

Table 3.5: Multiple Target tracking accuracy on KITTI dataset for Car and Pedestrian classes with number of particles=100, Best performance in Bold.

	MOTA	MOTP	Rcll	Prcn	F1	FAR	MT	PT	ML	IDs	FM
Baseline: (Fast-RCNN-particlePHD) without saliency											
Car	79.13	80.69	85.31	85.31	90.91	8.25	65.43	29.79	4.79	244	655
Ped.	58.56	75.05	65.97	92.17	76.90	7.85	35.93	54.49	9.58	147	547
(Fast-RCNN-PartilcePHD) with Saliency											
Car	84.82	81.88	90.94	96.79	93.77	10.52	77.13	19.86	3.01	268	553
Ped.	64.49	76.93	71.81	92.88	80.99	7.70	48.50	44.91	6.59	161	516

3.2 3D Object Detection

The Huawei-Insight Autonomous Vehicle project shed light on many interesting topics for further research in object detection. During this project, various challenges were encountered, followed by technical brainstorming sessions, which cultivated the idea of employing spatial and temporal information for 3D object detection. Unfortunately, the temporal information for the 3D object detection task is hardly explored in the literature.

To get more insight and broaden the knowledge on 3D object detection approaches, the exploratory work started by reviewing the recently published survey articles on 3D object

detection using deep learning networks by Arnold, Eduardo et al. [63]. The article highlighted the pros and cons of using sensing modalities like lidar for 3D object detection. However, our prior research involved processing images or video data. Further, lidar point cloud data is a distinct type of data and can be challenging to handle or process as it is sparse and irregular. Therefore Complex-YOLO [100] was chosen as the initial framework for 3D object detection due to prior familiarity with 2D object detection using YOLO architecture. In addition, the bird-eye-view method is employed to represent lidar point cloud data to train and validate the model performance.

3.2.1 Complex-YOLOv3

Pilot Study-3: The preliminary work on 3D object detection was split into two phases. The first phase included implementing Complex-YOLO [100] and Tiny-YOLO3D [98] with minor customisation. The second phase included incorporating temporal information into the network employed during the first work phase and comparing the results. Finally, the outcome obtained in both phases assisted in gaining hands-on experience on 3D detection networks and selecting baseline architecture to further explore the research questions in this thesis.

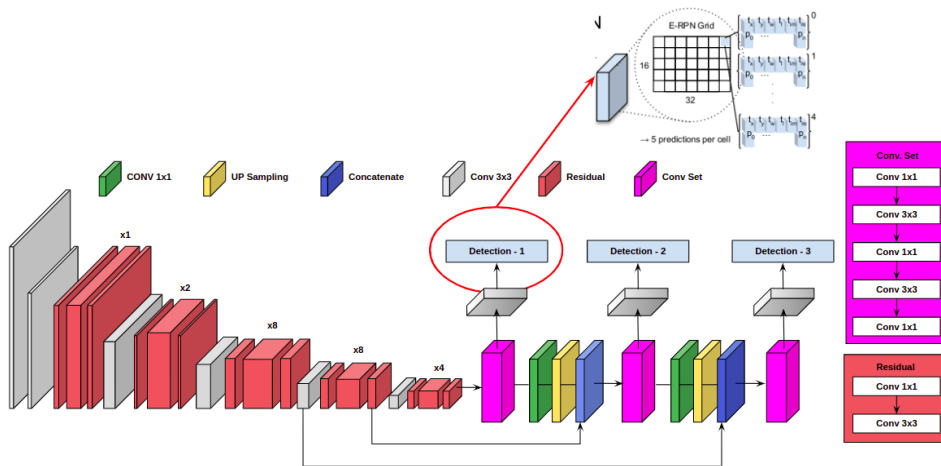


Figure 3.10: 3D Object Detection using Complex-YOLO Architecture

A Complex-YOLO architecture with a YOLOv3-like structure was employed in the first phase. The architecture is presented in Fig. 3.10. The network performs detection at three

different scales and is significantly better at detecting small objects. The network was implemented using PyTorch with TensorFlow as backend on an Intel i7 processor with a single GPU (GeForce GTX TITAN X, 12GB RAM) and trained on the KITTI 3D object detection dataset [138] with three object classes, namely 'car', 'pedestrian' and 'cyclist'. The training data was further split into 80% training and 20% validation sets. The 3D point cloud of a single frame is converted into a birds-eye-view (BEV) RGB map encoded with height, intensity and density values, covering an area of 50mx50m directly in front of the origin of the sensor. The size of the grid map is defined as 608x608 (height x width). The Fig. 3.11. represents input and detection results using lidar data with the bird-eye-view (BEV) representation. For a visual representation of 3d bounding box results, the estimations in the lidar plane are projected onto the RGB image. Both the Complex-YOLO and Tiny-YOLO3D models were trained from scratch for 300 epochs using adam optimiser [106] with a weight decay of 0.0005, momentum=0.9, batchsize=8 and learning rate = 0.001. The KITTI evaluation benchmark was used to measure the model's performance with the IoU thresholds of 0.7 for the 'car' and 0.5 for 'pedestrian' and 'cyclist'. Mean Average Precision (mAP) is employed to measure the accuracy of the model. The results are depicted in Table 3.6. shows an overall mAP of **88.24%** for Complex-YOLOv3 and mAP of **76.38%** for Tiny-yYOLOv3.

3.2.2 Complex-YOLOv3 with convolutional-LSTM

Pilot Study-4: The second phase of work was inspired by El Sallab et al., who proposed YOLO4d [91], incorporating temporal information into the Complex-YOLOv3 and Tiny-YOLOv3 architecture as shown in Fig. 3.12. The objective of integrating temporal information into the network is to retain information from the previous batch of frames the network has seen before and use the learning to make estimations in the current frame. This process flow makes the data preparation task required to train the network critical as the learning depends on the sequences fed to extract the temporal information. A convolutional-LSTM (CLSTM) shown in Fig. 3.12b is used with a Recurrent layer. Similar to an LSTM model, the previous hidden states are passed to the next step of the sequence, and the internal matrix multiplications are exchanged with convolution operations.

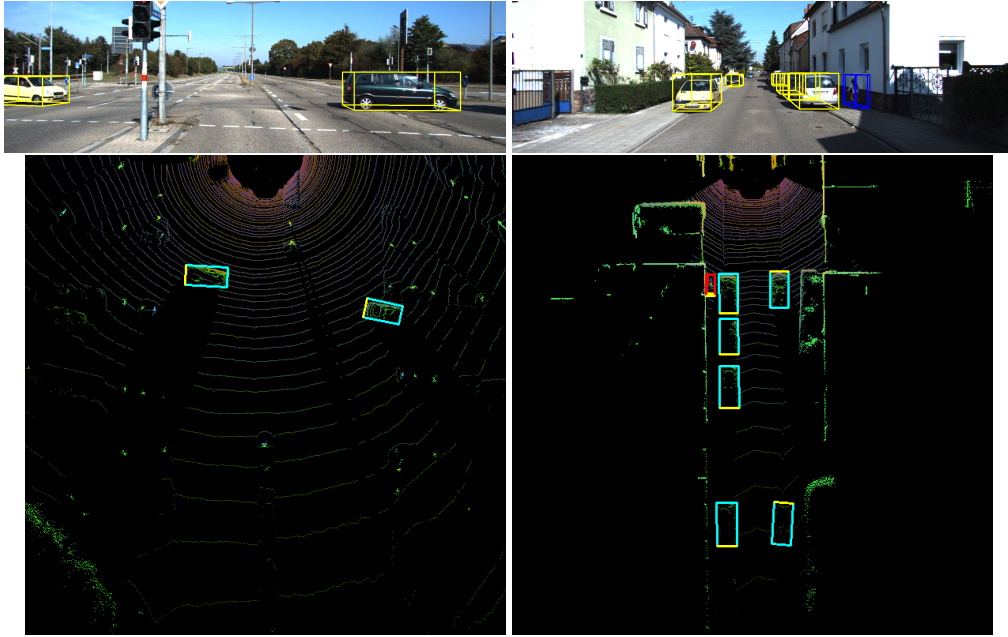
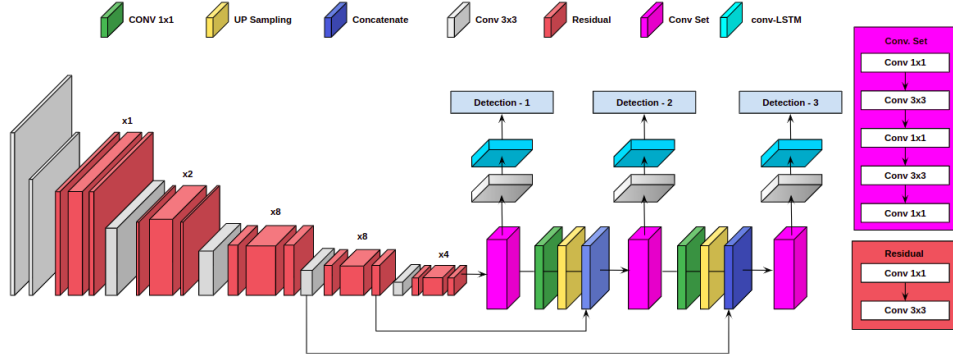
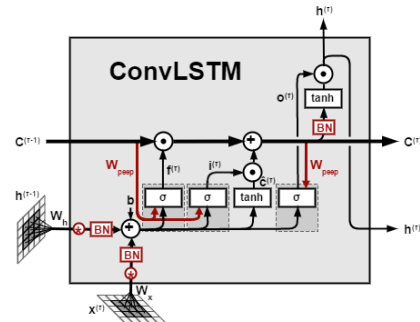


Figure 3.11: Top row: output 3d bounding box projections on RGB image and bottom row: detection results in LiDAR data with BEV representation in KITTI dataset Image 007271 (left) and 007294 (right)



(a) Complex-YOLO with CLSTM layer added between the linear and detection layer



(b) Conv-LSTM Cell

Figure 3.12: Incorporation of temporal information into Complex-YOLO architecture for 3D object detection

A CLSTM layer was added between the linear and detection layer, as shown in Fig 3.12a to extract temporal information. The network was trained on the KITTI 3D object detection dataset without maintaining the sequential coherence in the frames due to the non-availability of sequenced annotated data. The need for sequenced data appeared to be critical while training the CLSTM layer to memorise the spatial and temporal features. Similar training procedures and parameters were employed to train the Complex-YOLOv3-lstm and Tiny-YOLOv3-lstm models. However, the results demonstrated comparable detection accuracy to the original network used in phase 1. Table 3.6 presents the individual class accuracy results obtained with temporal information incorporated into the Complex-YOLOv3 architecture. The performance of the 'pedestrian' was improved by $\sim 7\%$ with Complex-YOLOv3-lstm. However, the performance of 'cyclist' is decreased by $\sim 7\%$. In the case of Tiny-YOLOv3-lstm, the mAP of 'cyclist' is improved by $\sim 4\%$, and for 'pedestrian', the mAP value is dropped by $\sim 12\%$. Furthermore, the results paved the way to outline additional studies to understand the influence of temporal information into the deep detection network and formulation of key tenets in extracting temporal features with multiple frame references from multimodal data and incorporation into the architecture for 3D object detection.

Table 3.6: 3D object detection results on KITTI validation dataset, following short notation are used to indicate the methodology: cYv3 = Complex-YOLO3, cYv3-LSTM = Complex-YOLO3 with CLSTM module, cYv3-tiny = Complex-YOLOv3 tiny version and cYv3-tiny-LSTM = Complex-YOLOv3 tiny version with CLSTM module. Legend: Rcll = Recall, Prcn = Precision, F1 = F1 Score, AP = Average Precision, mAP = mean Average Precision, Best performance in Bold.

Class	Method/ Metric	cYv3	cYv3-lstm	cYv3-tiny	cYv3-tiny-lstm
Car	Rcll	90.86	86.06	90.64	90.13
	Prcn	98.40	98.06	97.44	96.92
	F1	94.40	91.67	93.92	93.40
	AP	97.50	96.60	96.44	95.65
Ped	Rcll	56.89	58.53	53.28	49.11
	Prcn	95.10	93.76	79.64	71.00
	F1	71.19	72.07	63.85	59.04
	AP	77.84	85.66	58.27	46.86
Cyclist	Rcll	72.79	54.15	60.86	61.15
	Prcn	93.04	89.57	87.60	91.00
	F1	81.99	67.50	71.84	73.14
	AP	89.40	82.42	74.44	78.39
mAP		88.24	88.22	76.38	73.63

3.2.3 Baseline Architecture: Frustum-ConvNet

The lidar point cloud data representation in projection-based and voxel-based methods are simple and easily adapted to utilise the existing 2D detection framework. Unfortunately, these methods exhibit a loss of information and suffer geometric misalignment when the data is regularized. The experience gained during the Pilot Study-3 (Section 3.2.1) and Pilot Study-4 (Section 3.2.2) where the bird-eye-view (BEV) representation was employed to address occlusion problem in outdoor scenes by keeping the size of the object constant regardless of height. It was found that the Complex-YOLO architecture used heuristic height prediction based on data statistics which is limited to certain scenarios making it not able to handle occlusion problems in vertical space especially objects like poles or pedestrians, which has fewer points after projection. Interestingly, the introduction of PointNet [115] architecture in 2017 paved the way for the researchers to directly process the unstructured lidar point cloud data without any loss of information. Moreover, the PointNet enabled to capture of local and global point features directly from the lidar point cloud data. Eventually, the PointNet framework was adapted to various 3D-related tasks leading to the proposal of point-based 3D object detection methods [107, 80, 78, 44, 104, 86, 59].

Most recent point-based methods and fusion strategies are discussed in Section 2.2.1, Frustum-ConvNet [85] is adopted as the baseline architecture for addressing the research question mentioned in the Section 1.2 for the following three reasons: Firstly, prior localization of the objects is made using the 2D region proposals which can be obtained from an off-the-shelf object detector. This prior localization reduces the data processing time by avoiding processing entire point cloud data by the network. Secondly, the architecture is simple and follows sequential-based fusion [3]. Any new layer/module can be added or deleted, making it ideal for the current research investigation to segregate the individual modalities for incorporating temporal features and attention mechanisms into the network. Lastly, instead of sequentially fusing the cues from the image (Only the 2D coordinates of the region proposals as in [85]) and lidar data, to generate the joint representation of the modalities and study its impact on model performance fusion strategies [3, 57, 63, 68] are investigated.

In amodal 3D object detection, oriented 3D bounding boxes enclosing the full objects are estimated based on a single perspective of the observed object, which provides partial

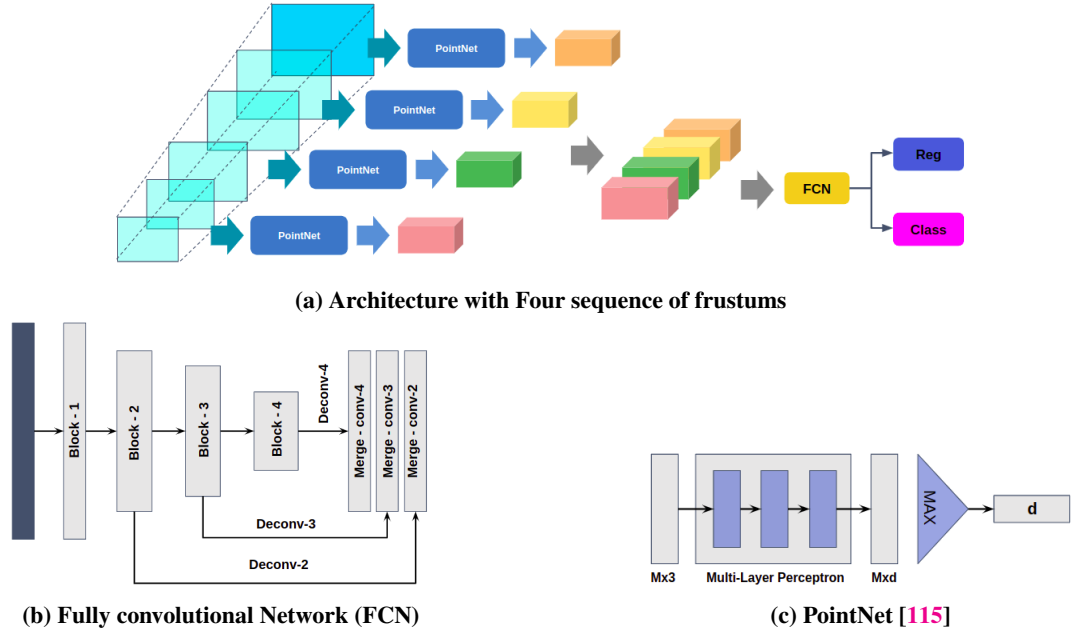


Figure 3.13: Simplified version of Frustum ConvNet Architecture [85]

surface information. Frustum ConvNet (F-convNet) architecture is shown in Fig. 3.13. The architecture supports amodal 3D object detection in an end-to-end learning fashion to estimate the orientated 3D boxes in a continuous 3D space using prior 2D region proposals obtained from the off-the-shelf 2D object detector. The point association with sequences of frustums are obtained by sliding a pair of parallel planes along frustum axes using the coordinate information of these 2D region proposals.

In this research, for developing baseline architecture shown in Fig. 3.13a. following configurations is considered: frustum-level feature are generated using PointNet [115] with a sequence of T ($=4$) frustums of height (u) are generated by sliding a pair of parallel planes along the frustum axis with a stride (s) using 2D region proposal in RGB space represented as $D = \{d_i \mid i = 0, 1, 2, \dots, N\}$, where $d_i \in \mathbb{R}^4$ and N is the number of objects in the scene. The estimation of the 3D bounding oriented boxes is formulated as $B = \{b_i \mid i = 0, 1, 2, \dots, N\}$, where $b_i \in \mathbb{R}^7$ and N is the number of objects in the scene which are parameterised as: box centre coordinates, (x, y, z) ; box dimension, (h, w, l) , representing the height, width and length of each box respectively; and the orientation or heading angle, θ , described using the raw point cloud data, $P = \{p_i \mid i = 0, 1, 2, \dots, N\}$, where $p_i \in \mathbb{R}^{3 \times M}$ and M is the total number of raw points. PointNet is then used to extract

point-wise feature inside each of the sequence frustums to generate the frustum points is represented as $F = \{f_i \mid i = 0, 1, 2, \dots, N\}$, where $f_i \in \mathbb{R}^{3 \times d}$ and d is the number of points inside the frustum. These points are aggregated at early stage as a frustum-level feature vector represented as $\{\mathbf{f}_{i=1}^L\}$, with $\mathbf{f}_i \in \mathbb{R}^d$ are re-formed as 2D feature maps of the size $L \times 3$, which will be used fed to subsequent fully convolutional network (FCN) shown in Fig. 3.13b for a continuous frustum-level feature fusion and 3D box estimation. The estimation includes classification and regression branches added on top of FCN, as shown in Fig. 3.13.

Layer specifications of the FCN module are depicted in Table 3.7. The FCN consists of four convolution blocks and three de-convolution layers corresponding to each block. The outputs of all de-convolution layers are concatenated along the feature dimension obtained from 2D feature maps of fused features generated through convolution and down-sampling operations across the frustums. A hierarchical granularity of frustum is obtained using the feature concatenation from virtual frustums of varying sizes to estimate the 3D boxes of object instances of unknown sizes. Focal loss [112] is used in the classification branch to handle the imbalance of foreground and background samples during training. The formulation of offset between the ground truth (g) and estimation (e) of oriented 3D bounding boxes are denoted as B_g and B_e , respectively. The offset is represented as $B_\phi = \{b_i \mid i = 0, 1, 2, \dots, N\}$, where $\phi \in \{g, e\}$, $b_i \in \mathbb{R}^7$ and N is the number of objects in the scene which are parameterised as: box centre coordinates, (x_ϕ, y_ϕ, z_ϕ) ; box dimension, (h_ϕ, w_ϕ, l_ϕ) , representing the height, width and length of each box respectively; and the orientation or heading angle, θ_ϕ .

$$\begin{aligned} \Delta x &= \frac{x_g - x_e}{x_e}, \Delta y = \frac{y_g - y_e}{y_e}, \Delta z = \frac{z_g - z_e}{z_e} \\ \Delta l &= \frac{l_g - l_e}{l_e}, \Delta w = \frac{w_g - w_e}{w_e}, \Delta h = \frac{h_g - h_e}{h_e} \\ \Delta \theta &= \theta_g - \theta_e \end{aligned} \quad (3.1)$$

similar to 2D object detector, the whole F-ConvNet is trained using a multi-task fashion with Euclidean distance-based regression loss for the box centers, smooth l1 regression

loss for box sizes, and the angle, focal loss is for classification, and corner loss is used to regularize box regression of all parameters.

Table 3.7: Fully convolutional Network (FCN) configuration used in Frustum-ConvNet architecture [85]

Name	Kernel size/ Filter No./Striding/Padding
Block - 1	3 x 128 / 128 / 1 / 1
Block - 2	3 x 128 / 128 / 2 / 1 3 x 128 / 128 / 1 / 1
Block - 3	3 x 128 / 256 / 2 / 1 3 x 256 / 256 / 1 / 1
Block - 4	3 x 256 / 512 / 2 / 1 3 x 512 / 512 / 1 / 1
Deconv - 2	1 x 128 / 256 / 1 / 0
Deconv - 3	2 x 256 / 256 / 2 / 0
Deconv - 4	4 x 512 / 256 / 4 / 0
MergeConv - 2	1 x 256 / 128 / 1 / 0
MergeConv - 3	1 x 512 / 256 / 1 / 0
MergeConv - 4	1 x 1024 / 512 / 1 / 0

The network was trained on KITTI Object dataset described in Section 2.3.1 with 3,712 training and 3,769 validation samples consisting of 'car', 'pedestrian', and 'cyclists' class on a single Nvidia-Titan X GPU with 12GB RAM for 50 epochs using the Adam optimizer [135] with momentum, learning rate and weight decay values set to 0.9, 0.001 and 0.0001 respectively. The learning rate is decreased by 10 times for every 15 epochs and batch size of 16 with the data samples undergoing augmentation, which included $\pm 10\%$ random shift from its centre, flipping in horizontal reference plane and the bounding box size is randomly scaled-up or scaled-down by 2 – 5% of its original bounding box size. The 2D ground truth annotation of the objects is considered as the input region proposal instead of the 2D detector output to cover all the possible objects in the scene which the external detector module may not have detected. All ground truth annotations are augmented before passing to the pipeline to resemble the detector output closely, and regions with less than the minimal lidar points are ignored. The number of sample points for each region proposal is set to have randomly selected 1,024 points. Same parameter initialisation as mentioned in F-ConvNet [85] are considered, the frustum resolutions of [0.5, 1.0, 2.0, 4.0] with corresponding stride value of [0.25, 0.5, 1.0, 2.0] for the 'car' category and frustum resolutions

of [0.2, 0.4, 0.8, 1.6] with corresponding stride value are set to [0.1, 0.2, 0.4, 0.8] for the 'pedestrian' and 'cyclist'. KITTI 3D evaluation benchmark is used to evaluate the model's performance with 3D IoU values of 0.7, 0.5, and 0.5 for the categories of 'car', 'pedestrian', and 'cyclist', respectively. The BEV detection and 3D detection results on the validation set are depicted in Table 3.8.

Table 3.8: Performance of BEV and 3D detection of Frustum-ConvNet on KITTI validation dataset. Legend: 'Easy', 'Mod.' and 'Hard' represent easy, moderate and hard cases respectively

Frustum-ConvNet	Car			Pedestrian			Cyclist		
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
BEV	97.88	87.93	89.72	75.93	76.02	75.99	88.95	89.00	88.99
3D	88.59	87.79	86.98	73.11	73.03	66.39	87.83	88.03	87.92

The exploration of F-ConvNet architecture facilitates the understanding of handling and processing raw lidar point cloud data. Furthermore, prior localization can further aid in optimizing the processing of sparse and unstructured lidar point cloud data. Although F-ConvNet depends on the 2D region proposals highlighting a fatal risk involved when failure of 2D detectors can bound to deteriorate all subsequent pipelines, scope exists to improve and adapt the pipeline to process the image feature instead of just using the coordinate position. The results inferred in this section form the foundation for further investigation and developments to corroborate F-ConvNet architecture, which is discussed in the following chapters.

3.3 Summary

This chapter presents the initial pilot study on 2D object detection using single-stage and two-stage frameworks trained and evaluated on the KITTI dataset. The experience gained with the 2D detector assisted in designing a detection-based tracking framework with a modified probability density filter to track multiple objects in the scene using visual saliency. In addition, the particles are propagated in the direction of the objects using visual saliency to handle occlusion and detection failure scenarios. The adoption of visual saliency into the tracking framework and the learnings from the experiments motivated the research further

to extend the work on 3D object detection, where the estimation of accurate depth information plays a vital role in developing algorithms related to perception systems, specifically for autonomous vehicles. Following are the brief inference of the trials:

1. Complex-YOLO3D with BEV representation of lidar point cloud data was employed to understand the 3D detector for estimating the oriented bounding box positions in the world coordinates. The trial with the Complex-YOLO3D laid the necessary foundation to incorporate convolutional-LSTM for extracting the Spatio-temporal information using convolutional-LSTMs and propagating the extracted temporal information into the network layer, which the final estimation layers will use. This initial investigation on temporal encoding aided the current research in acquiring further insight into the data representation aspect, and analysing the requirements in the dataset for effectively utilising the temporal aspect in the data will be discussed in [Chapter 4](#).
2. Understanding the structural information of an object in the scene is effective when the raw lidar point cloud data is used. However, the complexity of utilising the BEV representation with loss of information led to exploring the Frustum-ConvNet framework. With prior localization of the objects in the scene using the 2D detector, which is later transformed to the 3D space for extracting the lidar point cloud data within the proposal to generate the Frustum level features was fascinating and appeared to be an ideal set-up for exploring the temporal aspects. F-ConvNet was chosen as the baseline architecture to investigate further the research questions involving temporal cues and fusion methods presented in the following chapters.

Chapter 4

Data Representation and Performance Baselines

In the literature, lidar point cloud data is processed in various ways. Firstly, the data can be fed to the network in its raw form as in [115, 97, 85, 10]. Secondly, methods like [100, 103, 29, 105] project the data onto a 2D plane to generate a bird-eye-view (BEV), Similarly, [96] transformed the data to obtain a front-view image and [30] used range-view (RV). An alternative projection is to create a vertical column-like representation to form pillar-like structures termed as “point pillars” [70]. Finally, [95, 107, 62] have proposed transforming the point cloud into a voxel grid-structured format that can be fed to a CNN. All these methods aim to incorporate the spatial and depth information available from the lidar data in a form that can be utilised by object detection and localization models. This chapter addresses the Research Question: *How best to represent or prepare multi-modal data for training an object detection architecture?*

The knowledge gained on the available perception dataset during the literature review in Chapter 2 and the initial findings presented in Chapter 3, contributes to understanding the role of data representation in this context but require further exploration. Lidar point clouds are sparse relative to images, meaning that any points may not fully represent distant or small objects. Furthermore, since the 3D bounding boxes are derived from point cloud distributions, smaller scale objects like pedestrians or poles, which are visible in images, may lack corresponding 3D bounding boxes [35] due to fewer valid points scanned on them through lidar. Furthermore, these objects appear in various scenes with dynamic and static

background objects like trees, and bushes, making it extremely challenging to recognise the object in complex point cloud distributions.

The Bird-eye-view (BEV) representation used by complex-yolo [88], which was used in the pilot study presented in 3.2.1 and 3.2.2, was simple and easily adapted to utilise the existing 2D detection framework to localize the object and estimate its parameters. Although the representation exhibited loss of information and suffered geometric misalignment after the data transformation, it could handle occlusion problems in outdoor scenes by keeping the object's size constant regardless of height. However, the architecture cannot handle occlusion problems in vertical space, especially objects like poles or pedestrians with fewer points after projection.

The introduction of the PointNet [115] architecture to process the entire point cloud data made a significant difference in handling the lidar point cloud data for detection. First, the raw data is processed to predict the bounding boxes on each point in two stages, namely *abstraction* to extract the contextual feature after down-sampling and *feature propagation* to upsample and broadcast the feature to points that are discarded during down-sampling. Then, a 3D region proposal network (RPN) is employed to obtain the final predictions, followed by a refinement network. For instance, point pillars addressed the complexity of processing the point cloud by localizing the region using the prior 2D detection, as shown in Fig. 4.1-(c). The figure also presents the voxel representation where the lidar point cloud data is discretized into a volumetric 3D grid as in Fig. 4.1-(b). On the other hand, point-based networks achieved good accuracy but have a higher run time. Furthermore, object proposal generation relies on either image-based 2D object generation or an expensive per-point foreground segmentation network.

The data representations in the KITTI and nuScenes datasets for training and testing the object detection network are briefly described in Section 4.1. Visual comparison of lidar and image data to understand the temporal information is provided in Section 4.2. The inference obtained from the Chapter 3 has led to further exploration of F-ConvNet, to handle the point cloud data to extract the features at multiple scales leading to the following experiments presented in Section 4.3:

- Understanding the influence of region proposals from the 2D detector (4.3.1),

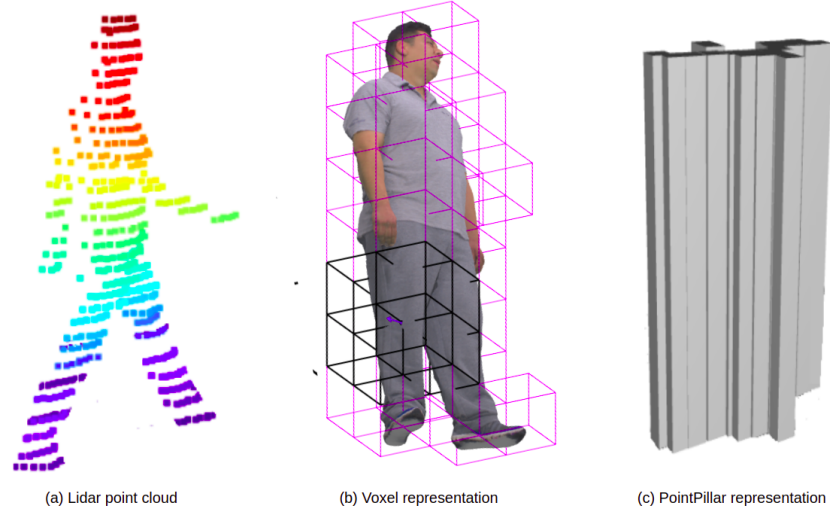


Figure 4.1: Illustration of lidar point cloud, Voxel [107] and PointPillar [70] data Representation for person [8]

- Determining the minimum sequence length for extracting temporal information from the image sequence (4.3.2),
- Investigating the effect of varying the feature vector size of the input data on performance of the baseline model (4.3.3),
- Evaluating the impact of utilising multi-sweep lidar point cloud data available in nuScenes dataset on performance of baseline model (4.3.4) and
- Evaluating individual class performance when the model is trained with all class categories together and separately (4.3.5).

4.1 Dataset

This section provides an overview of the KITTI and nuScenes datasets used for designing the pipeline and the evaluations to address the Research Question 1 and 2 mentioned in Section 1.2.

4.1.1 KITTI

The KITTI 3D object detection dataset is described more fully in Section 2.3.1. The dataset consists of 7,481 training images and 7,518 test images with corresponding point clouds,

comprising a total of 80,256 labeled objects for 8 classes: ‘Car’, ‘Van’, ‘Truck’, ‘Pedestrian’, ‘Person (sitting)’, ‘Cyclist’, ‘Tram’ and ‘Misc’. In the current research, only ‘Car’, ‘Pedestrian’, and ‘Cyclist’ classes appearing in the training samples are considered, which are randomly split into a training set comprising of 3,712 images with $\approx 17K$ annotations and a validation set comprised of 3,769 images with $\approx 17K$ annotations. The number of per-class annotations or instances appearing in the training and validation sets is depicted in Table.4.1.

Table 4.1: Number of instances in the KITTI train and validation datasets

Object Class	Train	Val	Total	(%)
car	14385	14357	28742	72.59
cyclist	893	734	1627	4.11
pedestrian	2280	2207	4487	11.33
person_sitting	166	56	222	0.56
tram	287	224	511	1.29
truck	606	488	1094	2.76
van	1617	1297	2914	7.36

KITTI was used to understand the data representation of image and lidar point cloud data, for initial pilot studies and for the development of the architecture to incorporate temporal information using F-ConvNet as base network. Fig. 4.2 illustrates the representation of image and lidar data with 3D annotation of the object appearing in the scene and Fig. 4.2-(c) shows lidar point clouds projected on to the image plane with each projected point representing depth information corresponding to each of pixel position in the image. Fig. 4.3 illustrates the projection-based representation of the lidar data in terms of the depth map, height map, reflectance map, and bird-eye-view map which are employed to estimate the 3D parameters directly in lidar space [100, 103, 96, 74]. The height maps are obtained by dividing the point clouds into several slices. The intensity directly represents the reflectance map measured by the lidar on a grid.

4.1.2 nuScenes

The nuScenes dataset comprises 1,000 scenes, distributed as training (700), validation (150) and test (150) sets of scenes. The dataset is packaged into ten archives each containing 85

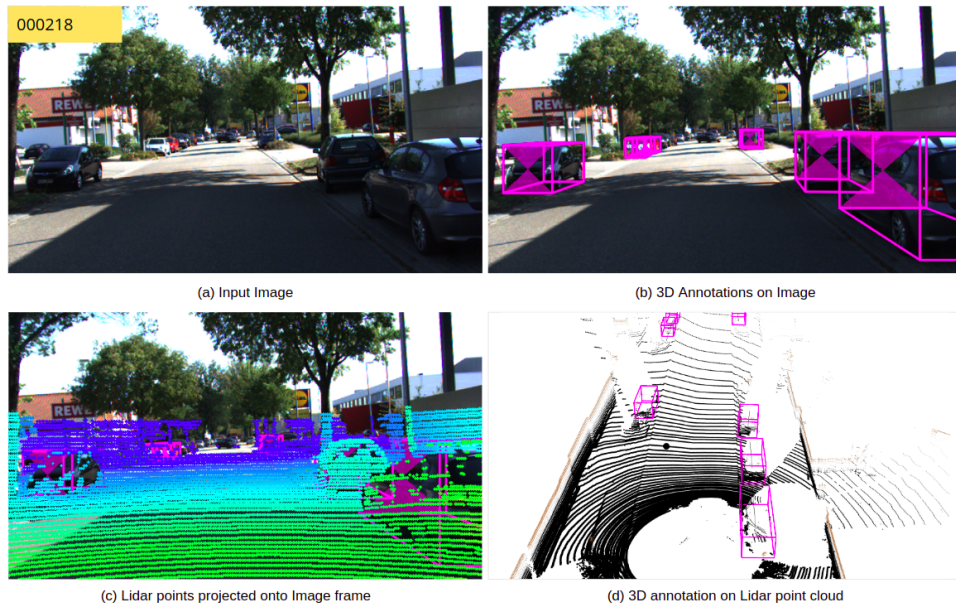


Figure 4.2: Image and Lidar Data Representation of the Scene:000218 in KITTI dataset

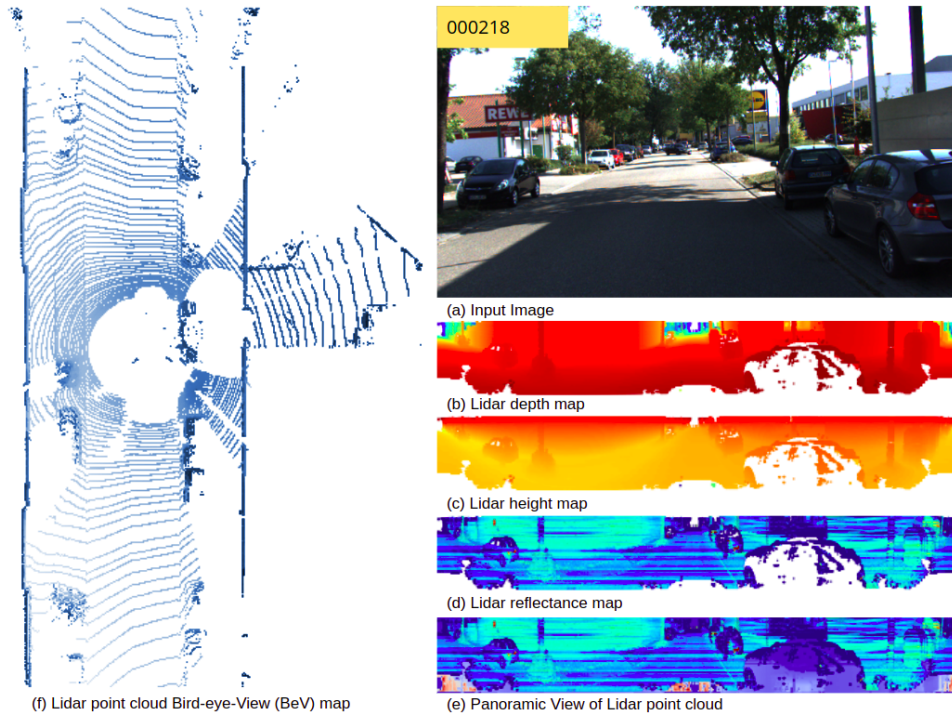


Figure 4.3: Front-View and Bird-eye-View Representation of lidar point cloud data for Image 000218 of KITTI training set

scenes with ≈ 40 samples per scene. For 3D object detection, the dataset provides complete 3D annotation of ≈ 1.4 billion, containing ten classes with eight attributes.

This study investigated the influence of temporal information and multimodal data on the detection network using the nuScenes dataset. The experimental setup had a single GPU with 12GB of RAM on an Intel i7 for training and testing the model. While training the F-ConvNet model on the KITTI dataset with $\approx 17K$ annotation with 3X augmentation, time for a single epoch took around 15-18 minutes. With the limited computing power training the network with ≈ 1.4 billion annotations was complex and time-consuming. Therefore, five of the available ten archives released in the nuScenes dataset repository comprising 425 scenes were selected. The first three archives were chosen as the training set comprising 255 scenes with 60,720 samples and $\approx 322K$ annotations, and the following two subsets for the test set comprised 170 scenes with 41,124 samples and $\approx 202K$ annotations. Further, the training dataset was divided into training and validation with 43,080 ($\approx 199K$ annotations) and 17,640 ($\approx 123K$ annotations) samples, respectively. Table. 4.2 provides the number of instances in each class in the subset of the nuScenes dataset.

Table 4.2: Number of instances in the subset of nuScenes dataset

Object Class	(%)	Train	Val	Test	No. of Instance
barrier	12.75	26168	15157	25572	66897
bicycle	1.32	2405	1680	2854	6939
bus	1.08	2044	1408	2239	5691
car	42.97	97475	40732	87167	225374
construction vehicle	1.47	2668	1065	3967	7700
motorcycle	1.14	2406	902	2695	6003
pedestrian	21.04	29261	40331	40777	110369
traffic cone	7.93	13452	11084	17060	41596
trailer	2.18	5145	1915	4373	11433
truck	8.11	18029	8683	15838	42550

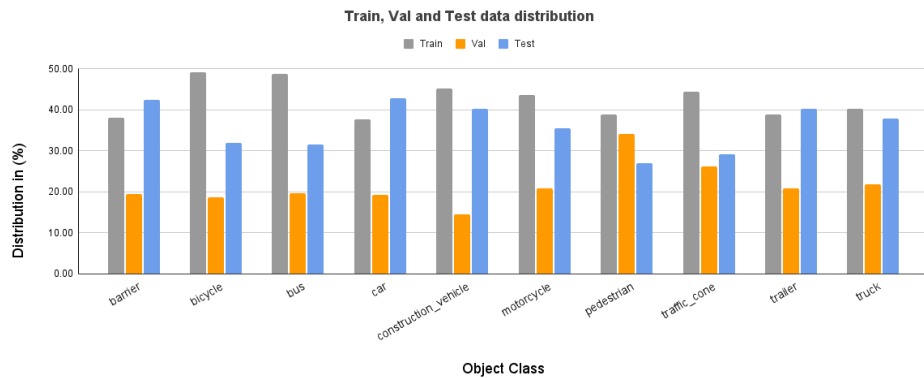
Table. 4.3 depicts the model’s performance for the individual class on the subset of the nuScenes dataset with all the classes. The customised nuScenes dataset with $\approx 322K$ annotations consumed more than 15 hours for training and validation of the F-ConvNet model for a single epoch and took around 15 days to finish 30 epochs. With this extended training time, the realisation of the various research questions would be difficult and drawing any inference from the investigation would not be feasible.

Based on the experience gained with the model training for all the ten classes with

Table 4.3: nuScenes detection metric of F-ConvNet for all the object class appearing in all the six cameras in the custom nuScenes test dataset. Trained only for 30 epochs

Object Class	AP	ATE	ASE	AOE
barrier	1	0.138	0.335	0.412
bicycle	0.85	0.098	0.286	0.659
bus	0.839	0.427	0.3	1.556
car	0.919	0.29	0.192	1.114
construction vehicle	0.817	1.282	0.44	0.96
motorcycle	0.858	0.208	0.372	2.287
pedestrian	0.953	0.134	0.339	0.198
traffic cone	0.944	0.156	0.403	-
trailer	0.561	0.558	0.408	1.032
truck	0.872	0.351	0.161	2.158
NDS	mAP	mATE	mASE	mAOE
0.7534	0.8521	0.3873	0.3148	0.3340

$\approx 322K$ annotations. To handle this problem of long model training time, it would be interesting to investigate this research on footage from the front camera only and evaluate the feasibility of utilising the inference gained on this subset of the nuScenes dataset. The new subset of the customised nuScenes dataset comprises the training set consisting of 255 scenes with 10,120 samples and $\approx 64K$ annotations, and the test set consists of 170 scenes with 6,854 samples and $\approx 38K$ annotations. The training data is further divided into training and validation with 7,180 ($\approx 40K$ annotations) and 2,940 ($\approx 24K$ annotations) samples, respectively.

**Figure 4.4: Sample distribution in the custom subset of nuScenes dataset**

The number of individual class annotations is depicted in Table. 4.4 and the class distribution in percentage (%) between the train, val and test sets are presented in Fig. 4.4. Furthermore, for consistency and ease of comparison with KITTI-like data, all annotations

are converted to the KITTI 3D object detection format [139] using the development toolkit provided in the nuScenes repository. Table. 4.5. depicts the model’s performance for the individual class on the customised subset of the nuScenes dataset of frontal camera view with all the classes.

Table 4.4: Number of instances in each of the classes in the subset of nuScenes dataset considered for training and testing

Object Class	Train	Val	Test	No. of Instance
car	16002	8199	18176	42377
pedestrian	8443	7409	5838	21690
barrier	5247	2698	5857	13802
bicycle	606	231	394	1231
bus	691	278	447	1416
construction vehicle	853	273	760	1886
motorcycle	493	235	400	1128
traffic cone	3609	2129	2371	8109
trailer	856	456	884	2196
truck	3622	1974	3414	9010

Table 4.5: nuScenes detection metric of F-ConvNet for all classes in the custom nuScenes test dataset

Object Class	AP	ATE	ASE	AOE
barrier	0.983	0.18	0.35	0.48
bicycle	0.975	0.118	0.246	2.299
bus	0.739	0.954	0.171	1.483
car	0.956	0.215	0.190	0.896
construction vehicle	0.856	0.962	0.499	2.840
motorcycle	0.975	0.179	0.411	2.543
pedestrian	0.981	0.109	0.284	0.24
traffic cone	0.989	0.103	0.455	-
trailer	0.447	0.989	0.669	1.421
truck	0.881	0.702	0.321	2.233
NDS	mAP	mATE	mASE	mAOE
0.7416	0.8782	0.4511	0.3596	0.3743

The results obtained on both customised nuScenes dataset cases yielded similar results, and the inference drawn can be applied to the complete set of the nuScenes dataset. However, it would be further beneficial to focus this research on ‘car’ and ‘pedestrian’ classes and footage from the front camera instead of considering all the classes and images from all six camera views. The prevalence of ‘car’ and ‘pedestrian’ in autonomous driving scenarios and their contrasting physical properties (shape, scale, speed) ensure these classes will

adequately test the proposed frameworks.

Fig. 4.5 presents a snapshot of the object's 3D annotations in the image and lidar plane. Fig. 4.5-(c) illustrates the projection of lidar point cloud data onto the image frame for visual understanding of lidar points distribution depicting the depth information corresponding to pixels enclosed within the bounding box representing the objects in the scene. Fig. 4.6 depicts the cumulative representation of a lidar point cloud distribution of a scene in nuScenes dataset containing an object for every ten frames. The bird-eye-view image is generated by aggregating the point clouds from multiple sweeps to get a denser point cloud for every ten frames. From top to bottom, left to right, starting at the first instance ($t=0$), the object contains fewer sparse points distribution, and as the AV approaches towards the object, represented by instance $t = 10, 20, 30, 40, 50, 60$. The sensor can scan more points to collectively analyse the features related to the object in the scene. The temporal reference of the object can benefit the deep detection network to learn the distribution of the object and eventually understand its structural information from different viewpoints, which are explored in Chapter 5.

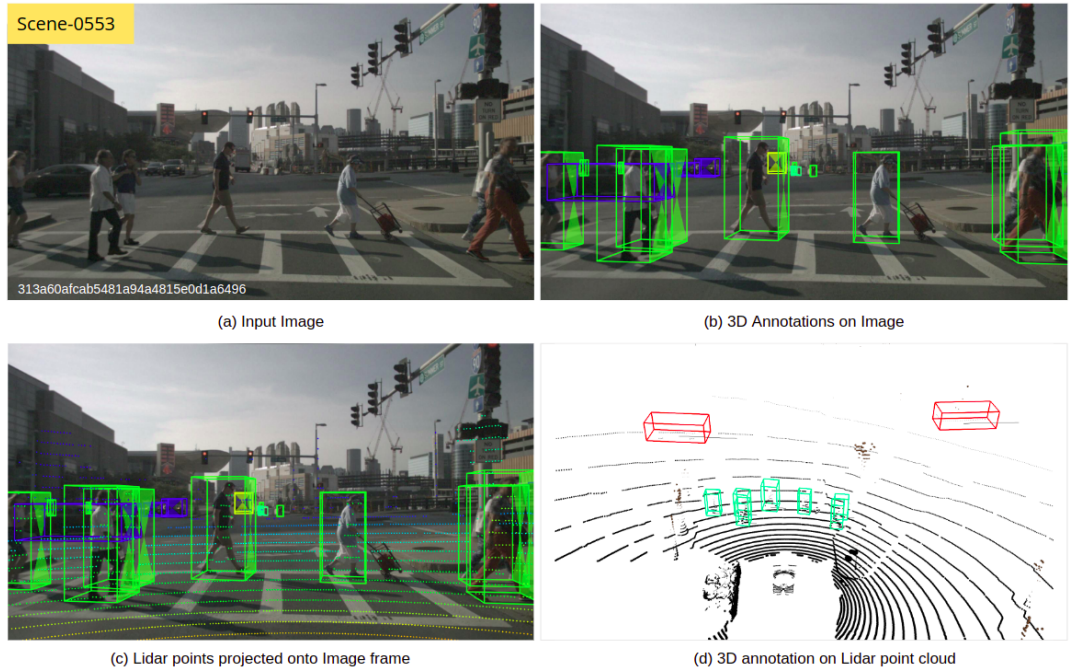


Figure 4.5: Image and Lidar Data Representation of the Scene:0553 in nuScenes dataset

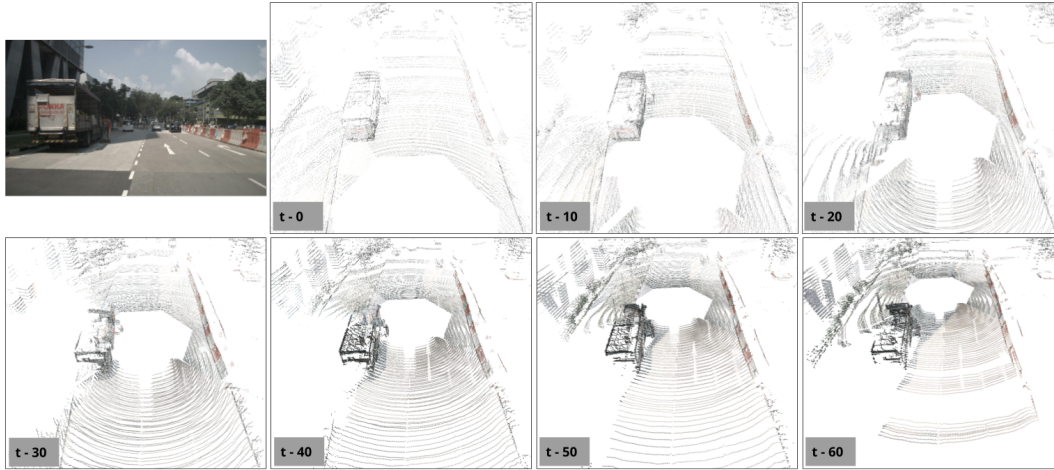


Figure 4.6: Cumulative representation of lidar point cloud data over time

4.2 Understanding Temporal information

The utilisation of spatial and temporal features plays a vital role in many computer vision-related tasks like 2D object detection and tracking, path prediction, and human pose estimation. In these applications, the features from the prior observation aid the framework in understanding the coherence in the current observation to derive new feature maps with reduced errors during parameter estimation. Most computer vision tasks employ image space to define temporal features, containing dense, rich, textural, and color information. However, image-based object detection methods have limitations in extracting the depth information, specifically in the case of single-frame (monocular) or stereo-based depth estimation methodologies, which cannot achieve the accuracy expected by the perception system. The image-based depth estimation can have low performance due to changes in lighting, weather conditions and limited knowledge about the scene. It requires a complex and computationally expensive process to determine the depth of the objects by only using the 2D image. Although, recent monocular image-based methods like DFR-Net [28], MonoDIS [82], M3D-RPN [64] have been proposed that use an RGB image to predict objects on the 2D plane and then perform 2D to 3D lifting to create 3D object detection results with lesser computational cost. However, the performance is still far from satisfactory due to the lack of reliable depth prior and the variance of the object scale caused by perspective projection [28]. Lidar and Radar utilize other modalities and offer an alternative to address the challenge of depth

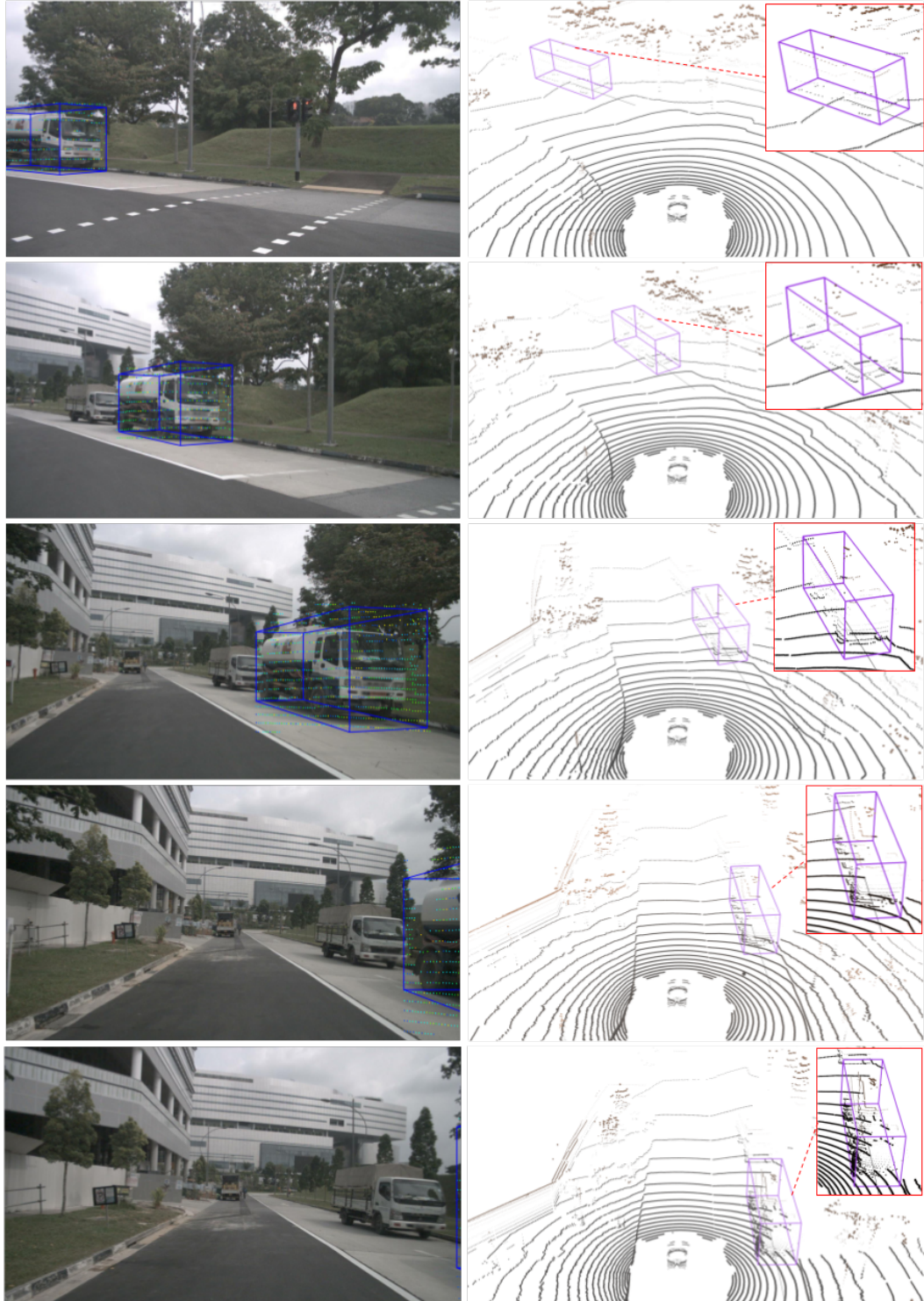


Figure 4.7: Illustration of Lidar data enclosed within the ROI of the truck on Image and Lidar frame for temporal understanding. The bounding box of the object is obtained from the ground truth annotations in the nuScenes dataset.

perception. In contrast, 3D object detection using lidar produces more accurate depth estimation than image based. The lidar data is sparse and contains the spatial-depth information for each point scanned on the object. Although it lacks textural and color information about the object, point cloud distributions are used for extracting spatial features.

In this section, the importance of tracking change in the point cloud data distribution as the AV acquires the data of a scene containing the object of interest is illustrated through Fig. 4.7 with the help of an instance where both the image and lidar frames are employed to understand the coherence of temporal information. The spatio-temporal information can be extracted in the sequenced frames to understand the structural and spatial information of relative locations to achieve better geometric alignment. In the Fig. 4.7, from top-to-bottom, presents the lidar point cloud data projected onto the image plane enclosed within the 3D bounding box of the object to collate the variation in both the plane and helps in establishing a visual correspondence between the image and lidar data of a scene as seen by the AV. The 3D bounding boxes projected on the image and lidar data are obtained from the ground truth annotation of the nuScenes dataset. Utilising the prior localization of the objects optimises the computational complexity when processing the entire point cloud data as the points within this prior regions can be considered as segmented points of the objects which are achieved as free-of-charge [60]. The sequenced data exhibits temporal continuity and the objects appearing in successive frames will drift slightly allowing the utilization of detection and temporal information from earlier frames to refine estimations in the current frame.

4.3 Exploratory Trials

This section presents experimental trials employing KITTI and nuScenes dataset to acquire inference on customised nuScenes subset data, understand spatio-temporal feature extraction using multiple frame reference, effect of region proposals on 3D estimations, effect of feature vector length of the lidar data used for representing the object class, impact of employing single-sweep lidar data versus multiple-sweep lidar data, and lastly, to study the behaviour of model performance when the object classes are trained individually and together.

4.3.1 Influence of 2D proposal on 3D estimations

The F-ConvNet uses an off-the-shelf 2D detector for prior localization of objects in the image space. These coordinate positions are then used to extract the ROI's of the object in the lidar space. This section presents the investigation to understand the influence of 2D region proposals for generating the frustum region to extract the point features by the F-ConvNet architecture. For the generation of region proposals, YOLOv5 detector trained on KITTI dataset for 'cars', 'pedestrian' and 'cyclist' class detailed in Section 3.1.1 of Chapter 3 is employed. The intersection over union (IoU) values and object confidence score are varied to generate the detection results that are passed as input region proposals to the F-ConvNet pipeline. Table. 4.6 depicts the number of 2D region generated represented as '2D(r)' obtained by varying the object confidence ('Conf.') score and IoU threshold ('Th.') value. The corresponding number of 3D estimation represented as '3D(e)' are estimated using these 2D region proposals. The IoU threshold values is varied in the range between 0.5 to 0.95 with an increment of 0.05 for all the object confidence score greater than 0.1, 0.01, 0.001, and 0.0001. In Table. 4.6, a maximum of 221,182 proposals are generated when $Th. \geq 0.5$ and the Confidence score is greater than 0.0001. similarly, a minimum of 17,830 proposals are generated when $Th. \geq 0.95$ and the Confidence score is greater than 0.1. From the Table. 4.7, it can be concluded that the computational complexity of the network increases with more prior region proposals to process. Further, Fig. 4.8 presents the graph representing the average computation time per frame taken by the F-ConvNet to estimate the 3D bounding box parameters of the objects and it can be concluded that the processing time decrease when the IoU threshold value (Th.) is increased which generates fewer region as depicted in Table. 4.6. In Table. 4.6, the trial with $Conf. \geq 0.5$ the region proposal obtained with $Th. \geq 0.95$ was initially used to generate the region proposals which is reported and was considered as a baseline to analyse the influence of varying the confidence score and threshold value.

The performance of BEV detection results for 'car', 'pedestrian', and 'cyclist' are depicted in Table. 4.8, Table. 4.9, and Table. 4.10 respectively. The mAP values get better as the IoU threshold value is increased and the confidence values are decreased. The 'car' attains the best mAP of **89.69 (%)** and **89.65 (%)** with $Conf. \geq 0.1$ and $Th. \geq 0.95$ for

Table 4.6: Number of Region proposals generated ‘2D(r)’ from 2D detector by varying IoU threshold (‘Th.’) and Confidence Score (‘Conf.’) and corresponding number of estimation by 3D detector generated ‘3D(e)’ on KITTI validation set.

Conf./ Th.	0.1		0.01		0.001		0.0001		0.5	
	2D(r)	3D(e)	2D(r)	3D(e)	2D(r)	3D(e)	2D(r)	3D(e)	2D(r)	3D(e)
0.50	48,730	15,319	74,065	16,491	117,592	18,281	221,182	24,371	-	-
0.55	29,056	14,703	42,797	15,823	72,402	17,821	151,894	24,476	-	-
0.60	23,906	14,487	31,787	15,379	51,924	17,283	114,296	24,065	-	-
0.65	21,758	14,350	26,808	15,068	40,710	17,001	90,165	23,561	-	-
0.70	20,571	14,312	24,099	15,097	34,094	16,616	73,808	23,113	-	-
0.75	19,721	14,169	22,385	14,783	29,709	16,249	61,656	22,631	-	-
0.80	19,129	13,903	21,244	14,725	26,726	16,059	52,474	22,110	-	-
0.85	18,663	13,868	20,388	14,400	24,629	15,777	45,466	21,332	-	-
0.90	18,248	13,762	19,701	14,338	23,085	15,538	39,855	20,572	-	-
0.95	17,830	13,619	19,063	14,097	21,825	15,282	35,496	19,691	17,266	15,552

Table 4.7: Average computation time per frame in milliseconds (msec) on KITTI validation set by varying the 2D detector confidence value (Conf.) and IoU threshold value (Th.).

Conf./ Th.	0.1	0.01	0.001	0.0001	0.5
0.50	38.79	58.95	93.60	176.05	-
0.55	23.13	34.07	57.63	120.90	-
0.60	19.03	25.30	41.33	90.98	-
0.65	17.32	21.34	32.40	71.77	-
0.70	16.37	19.18	27.14	58.75	-
0.75	15.70	17.82	23.65	49.08	-
0.80	15.23	16.91	21.27	41.77	-
0.85	14.86	16.23	19.60	36.19	-
0.90	14.52	15.68	18.37	31.72	-
0.95	14.19	15.17	17.37	28.25	13.74

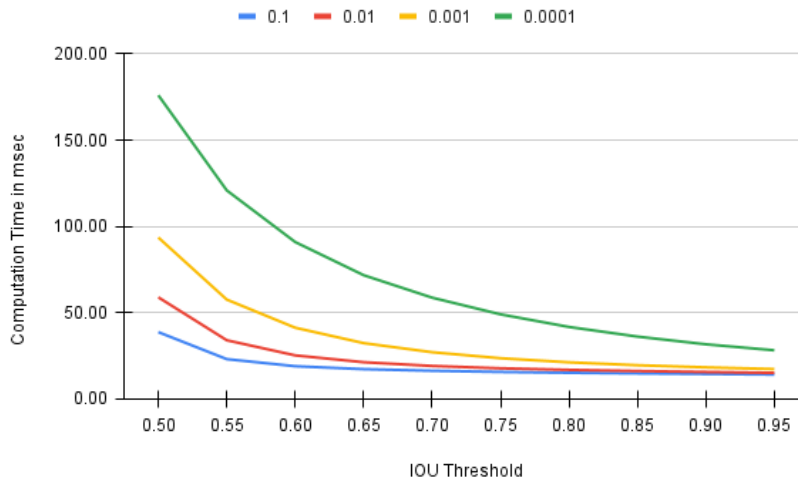


Figure 4.8: Average computation time per frame taken by the detector for 3D estimations depending on the proposal confidence and IoU threshold

Table 4.8: BEV Detection in AP (%) for ‘Car’ Class depending on the regional proposal generated from the YOLOv5 2D detector by varying the confidence (Conf.) and IoU threshold value (Th.). Best performance in bold.

Conf./ Th.	0.1			0.01			0.001			0.0001		
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
0.50	81.60	81.11	80.93	81.42	80.95	80.77	81.30	80.80	80.57	89.39	86.91	80.12
0.55	81.61	81.14	80.98	81.43	80.93	80.78	81.41	80.83	80.61	89.89	86.64	86.67
0.60	81.51	81.07	80.92	81.44	80.86	80.70	89.18	80.74	80.49	89.50	85.98	86.18
0.65	81.54	81.08	80.95	81.39	80.90	80.75	88.66	80.68	80.45	89.70	87.24	86.37
0.70	81.59	81.19	81.03	81.40	80.92	80.78	88.69	88.05	80.43	89.68	87.82	86.69
0.75	81.55	81.12	80.97	89.07	88.85	80.84	88.40	87.83	88.07	89.39	88.13	87.21
0.80	81.53	81.09	80.95	81.38	80.83	80.67	89.97	87.85	88.12	89.36	88.18	87.35
0.85	89.48	81.15	81.05	88.87	88.67	88.88	89.90	88.43	88.41	89.52	88.47	87.83
0.90	89.35	89.17	89.34	90.25	88.88	89.07	90.01	89.17	88.87	89.39	88.58	87.95
0.95	90.56	89.69	89.65	90.26	89.59	89.44	90.19	89.50	89.19	90.70	88.82	88.34

Table 4.9: BEV Detection in AP (%) for ‘Pedestrian’ Class depending on the regional proposal generated from the YOLOv5 2D detector by varying the confidence (Conf.) and IoU threshold value (Th.). Best performance in bold.

Conf./ Th.	0.1			0.01			0.001			0.0001		
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
0.50	71.51	64.83	64.52	62.99	61.96	61.81	67.88	61.02	61.02	63.03	56.60	56.43
0.55	65.08	64.86	64.50	69.00	62.53	62.33	67.76	60.07	60.04	64.51	61.61	57.43
0.60	71.56	64.86	64.74	61.93	61.46	61.20	66.06	58.95	59.16	64.34	60.97	57.18
0.65	63.74	63.65	56.91	67.83	61.55	61.81	65.17	57.98	58.28	61.56	55.11	55.31
0.70	69.69	63.44	63.44	67.45	62.23	62.05	66.94	60.15	60.22	65.47	61.80	58.30
0.75	70.95	65.01	64.87	69.31	62.31	62.60	67.24	65.10	60.39	63.36	61.08	57.13
0.80	70.35	64.20	64.44	68.63	62.46	62.69	67.89	66.09	61.01	65.82	63.70	59.01
0.85	71.46	64.67	64.78	69.61	68.87	63.05	68.85	67.34	66.93	67.47	65.58	64.05
0.90	71.47	71.31	64.99	69.63	68.18	62.86	70.48	68.86	68.04	72.44	66.46	64.84
0.95	71.86	71.92	64.86	71.47	71.04	70.55	75.23	69.06	67.84	67.58	65.78	63.99

Table 4.10: BEV Detection in AP (%) for ‘Cyclist’ depending on the regional proposal generated from the YOLOv5 2D detector by varying the confidence (Conf.) and IoU threshold value (Th.). Best performance in bold.

Conf./ Th.	0.1			0.01			0.001			0.0001		
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
0.50	77.90	77.06	76.85	87.00	76.16	75.84	86.96	81.96	75.16	84.86	79.49	72.71
0.55	79.81	77.17	76.86	87.29	83.71	76.41	86.50	82.38	74.93	82.83	79.47	72.94
0.60	87.73	77.25	77.06	87.08	83.13	75.78	85.58	81.76	80.98	83.01	77.68	71.53
0.65	79.61	76.30	75.89	88.25	83.76	76.19	86.45	82.29	74.86	82.91	78.36	71.67
0.70	87.92	84.48	76.67	87.66	83.96	76.24	84.37	75.00	74.58	84.11	78.35	71.69
0.75	88.58	85.40	77.27	86.63	83.88	76.57	84.98	81.32	74.58	83.42	79.28	78.24
0.80	87.65	84.51	76.65	86.86	83.61	75.72	85.11	81.88	80.66	82.93	79.34	78.20
0.85	88.34	84.69	76.96	87.71	83.48	76.22	87.87	83.22	82.32	85.77	80.55	79.84
0.90	88.40	84.43	76.82	88.60	84.79	83.81	88.09	83.65	82.91	85.80	80.99	80.44
0.95	88.69	85.66	85.28	89.23	85.52	85.06	88.10	84.17	83.64	85.89	82.03	81.11

Table 4.11: 3D Detection in AP(%) for ‘Car’ depending on the regional proposal generated from the YOLOv5 2D detector by varying the confidence (Conf.) and IoU threshold value (Th.). Best performance in bold.

Conf./ Th.	0.1			0.01			0.001			0.0001		
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
0.50	80.61	79.40	79.07	80.63	79.36	78.97	80.54	79.26	78.84	85.77	78.79	78.01
0.55	80.59	79.47	79.18	80.36	79.26	78.94	80.46	79.28	78.89	88.12	78.66	77.87
0.60	80.66	79.47	79.12	80.42	79.26	78.93	80.42	79.22	78.18	85.43	78.17	77.36
0.65	80.71	79.45	79.19	80.69	79.45	79.16	80.06	78.96	78.60	87.15	78.36	71.67
0.70	80.64	79.57	79.30	80.53	79.46	79.10	80.15	78.91	78.49	86.68	77.60	76.73
0.75	80.69	79.57	79.31	80.45	79.36	79.13	80.33	78.96	78.53	86.87	77.57	76.77
0.80	80.51	79.47	79.22	80.56	79.33	79.06	86.44	78.82	78.55	86.98	77.73	77.04
0.85	80.69	79.68	79.50	87.02	79.24	79.05	86.51	78.83	78.57	87.19	82.83	77.29
0.90	80.61	79.56	79.43	87.58	79.50	79.36	88.28	85.46	78.67	87.03	82.59	82.79
0.95	89.04	79.90	79.75	88.92	87.09	87.01	88.33	86.89	86.03	87.12	85.23	83.25

Table 4.12: 3D Detection in AP(%) for ‘Pedestrian’ depending on the regional proposal generated from the YOLOv5 2D detector by varying the confidence (Conf.) and IoU threshold value (Th.). Best performance in bold.

Conf./ Th.	0.1			0.01			0.001			0.0001		
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
0.50	62.42	61.91	55.51	61.26	59.64	53.97	59.66	58.33	57.92	55.64	53.57	52.62
0.55	62.75	62.08	55.88	60.91	60.09	54.21	64.33	57.69	57.30	60.28	54.67	53.82
0.60	62.81	62.56	55.95	60.07	59.27	53.15	58.65	56.83	56.57	59.65	54.44	53.36
0.65	62.05	61.59	55.53	59.51	59.11	59.16	61.99	55.78	58.28	56.17	51.62	50.96
0.70	61.57	61.56	55.18	59.81	60.12	59.77	63.54	57.69	57.36	59.10	54.79	54.37
0.75	62.87	63.31	62.89	66.12	60.00	59.74	63.20	57.58	57.68	57.96	53.62	53.75
0.80	67.67	61.94	62.03	60.87	60.26	60.24	64.45	58.45	58.48	61.01	55.80	55.47
0.85	68.62	62.68	62.81	67.18	61.29	61.19	66.44	64.47	59.82	63.03	60.56	57.06
0.90	69.00	63.17	63.33	67.01	60.54	60.77	67.97	65.83	60.81	65.00	61.72	58.03
0.95	69.76	69.74	63.37	69.02	68.00	62.47	68.43	66.59	61.09	64.82	61.63	57.88

Table 4.13: 3D Detection in AP(%) for ‘Cyclist’ depending on the regional proposal generated from the YOLOv5 2D detector by varying the confidence (Conf.) and IoU threshold value (Th.). Best performance in bold.

Conf./ Th.	0.1			0.01			0.001			0.0001		
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
0.50	79.36	76.30	75.99	79.37	75.40	75.02	86.12	81.10	74.44	84.30	78.16	72.14
0.55	79.07	76.50	76.07	86.35	82.82	75.63	86.27	81.97	74.59	82.82	78.17	72.00
0.60	87.60	76.89	76.59	85.82	81.92	74.63	84.65	80.80	73.47	80.34	76.24	70.14
0.65	79.08	75.58	74.72	87.71	83.33	75.49	85.21	81.15	73.39	81.11	75.51	69.98
0.70	87.26	83.75	75.95	86.92	83.04	75.28	83.15	73.87	73.38	82.77	76.52	70.10
0.75	87.54	84.19	76.25	85.43	75.73	75.24	83.23	79.92	73.41	81.02	76.68	75.16
0.80	86.94	76.14	75.97	85.98	82.68	74.89	84.08	80.78	73.33	80.52	77.39	75.65
0.85	88.08	84.10	76.42	87.18	82.59	75.38	86.72	81.90	81.04	83.78	78.87	77.28
0.90	88.01	83.42	76.01	88.26	84.17	83.38	87.37	83.03	82.17	84.66	79.89	78.35
0.95	88.04	85.16	84.37	90.96	85.52	85.06	88.00	83.98	83.28	84.84	81.08	80.09

Table 4.14: BEV detection and 3D detection AP in (%) on KITTI validation dataset using the regional proposal generated from the YOLOv5 2D detector with confidence (Conf. = 0.5) and IoU threshold value (Th. = 0.95).

Frustum-ConvNet	Car			Pedestrian			Cyclist		
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
BEV	90.07	89.35	89.08	67.82	66.76	65.40	87.35	83.64	82.53
3D	88.61	86.86	85.21	64.47	62.23	58.44	87.21	83.33	82.09

moderate and hard cases respectively. However, the mAP for easy cases is slightly better when $Conf. \geq 0.0001$ and $Th. \geq 0.95$ producing **90.70 (%)** mAP in contrast with $conf. \geq 0.1$. Similarly, the hard cases of ‘pedestrian’ achieves the best mAP of **70.55 (%)** with $Conf. \geq 0.01$ and $Th. \geq 0.95$ and for the easy and moderate cases, it is achieved with $Conf. \geq 0.1$ and $Th. \geq 0.95$. In the case of ‘cyclist’, the best mAP for all the cases is achieved with $Conf. \geq 0.1$ and $Th. \geq 0.95$.

A similar trend was observed in the performance of 3D object detection results for ‘car’, ‘pedestrian’, and ‘cyclist’ depicted in the Table. 4.11, Table. 4.12, and Table. 4.13 respectively. The ‘car’ achieves the best mAP of **87.09 (%)** and **87.01 (%)** with $Conf. \geq 0.01$ and $Th. \geq 0.95$ for moderate and hard cases respectively. However, the mAP for easy cases is slightly better with $Conf. \geq 0.1$ and $Th. \geq 0.95$ producing **89.04 (%)** mAP than with $conf. \geq 0.01$. For ‘pedestrian’ and ‘cyclist’ the best mAP for all the cases is obtained when the $Conf. \geq 0.1$ and $Conf. \geq 0.01$ respectively with $Th. \geq 0.95$. Although the mAP achieves the best value with various confidence scores and IoU threshold values it also increases the computation complexity of the network. A trial with $Conf. \geq 0.5$ and $Th. \geq 0.95$ was conducted where the average processing time was around **13.74 msec** and the results of BEV detection and 3D object detection are depicted in Table. 4.14.

In summary, the prior localization of objects using a 2D detector will have a significant influence on the performance of the 3D detection module. From the obtained results, it can be concluded that more regions around the object increase the network’s proximity to obtain better object localization in 3D space. However, more prior regions create computational overhead, which may be acceptable during experimentation but not deployment in any practical scenario as it provided more regions containing the objects. Further, the selection of the 2D detector module for the generation of the region proposals is essential as it requires the

detector to detect all the objects appearing in the scene with a good confidence score. This could become a major bottleneck during the investigation of the research questions proposed in this research. To address this issue, ground truth annotations available in the KITTI and nuScenes datasets are considered as the input region proposals for further study and investigation of various adaptations to the F-ConvNet architecture. Further, the use of ground truth annotations provides access to all the regions with the objects in the scene, which the external detector module may not have detected. All ground truth annotations are augmented to closely resemble the 2D detector-like output; the objects visible in both modalities are considered, and the regions with 'zero' lidar points are discarded. The 3D estimations for the objects visible in either of the modalities are not addressed in this thesis.

4.3.2 Minimum Sequence for Temporal information

Urban driving environments consist of various road elements like traffic signs, poles, barriers, traffic cones etc., and users like cars, pedestrians, cyclists etc. Object detection is therefore very complex as it requires the perception system [79] to be able to detect the location and depth of these objects in a highly dynamic scene containing multiple varied object classes. The spatial (position and appearance) information of these dynamic targets exhibits variations in the lidar points distribution over time (temporal information) with reference to the scene and the vehicle movement. Using spatio-temporal features in deep learning-based object detection models can improve the estimation of 3D bounding box parameters by mitigating the alignment issues or loss of object information due to perspective changes. This section presents the experimental trials to understand the minimum number of successive frames required by the network to extract the spatial and temporal features of the objects associated with the frame.

Table 4.15: 3D Detection AP (%) of F-ConvNet with convolutional-LSTM (CLSTM) and convolutional-GRU (CGRU) on KITTI validation set. Best performance in bold.

Feature Vector Size	Car			Pedestrian			Cyclist		
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
F-ConvNet	88.59	87.79	86.98	73.11	73.03	66.39	87.83	88.03	87.92
F-ConvNet-clstm	88.22	77.42	77.98	71.48	74.10	63.08	88.26	88.72	88.34
F-ConvNet-cgru	76.28	75.87	76.69	71.43	64.88	65.52	87.56	78.14	70.28

Table 4.16: BEV Detection AP (%) of F-ConvNet with convolutional-LSTM (CLSTM) and convolutional-GRU (CGRU) on KITTI validation set. Best performance in bold.

Feature Vector Size	Car			Pedestrian			Cyclist		
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
F-ConvNet	97.88	97.93	89.72	75.93	76.02	75.99	88.95	89.00	88.99
F-ConvNet-clstm	90.17	89.17	88.60	75.41	73.76	66.38	89.56	89.53	88.87
F-ConvNet-cgru	88.62	80.22	80.34	73.77	66.86	67.31	88.59	79.30	79.13

For the extraction of spatio-temporal features from input lidar data sequence, two popular RNN-based options – convolutional-LSTM (CLSTM) and convolutional-GRU (CGRU) – are considered as potential candidates and were incorporated into the F-ConvNet architecture. The preliminary trials training both the network on KITTI dataset and the hyper parameters were initialised as mentioned in Section 3.2.3. Although the KITTI dataset does not contain sequenced data, the results from this trial provided the initial knowledge about the suitability of CLSTM or CGRU module. The 3D detection (Table. 4.15) and BEV detection results (Table. 4.16) corroborated the need for sequenced annotated data to train the CLSTMs or GRU module for extracting the temporal information of the objects appearing in sequential data.

It was interesting to observe that the performance of the F-ConvNet with CGRU drops in contrast to the CLSTM. This drop in performance could be due to the non-existence of internal memory in CGRU module to remember any information regarding the object that were learnt from the previous sequence to gain more insight about the temporal information for better estimation of the parameters in the sequenced data. Based on the comparative results and observation, F-ConvNet with CLSTM architecture was proposed to investigate the minimum frames required for extracting temporal information on a custom nuScenes dataset discussed in Section 4.1.2.

Table 4.17 presents various input configurations of the nuScenes data which consist of annotation of synchronized keyframes (image and lidar) at 2Hz (2 frames per second) with each scene containing ≈ 40 samples which correspond to approximately ≈ 20 seconds of video. This setup is used to investigate the minimum number of sequenced frames required to extract and learn temporal features of the object for improved estimations with reduced errors. Single frame reference with random shuffle is used as the baseline performance for the analysis of the results obtained using the F-ConvNet with CLSTM architecture. The

input configuration includes varying the number of frames starting with one, three, and five consecutive frames, which are processed together as a batch and the number of objects recurring in the successive frame until it is no longer visible. The maximum number of recurring objects in the successive frames are varied from eight to twelve for all the objects appearing in a frame. This helps in understanding the impact on performance when the objects appear for only one or two instances and then vanishes.

Table 4.17: Detail of the input configurations utilised to investigate the minimum number of sequenced frames required to learn temporal information of an object by F-ConvNet with convolution-LSTMs

No.	Description
Baseline	Single frame (randomly suffled data)
Config-1	3 Consecutive Frames with all objects
Config-2	3 Consecutive frame with max. recurring 8 Objects
Config-3	3 Consecutive frame with max. recurring 12 Objects
Config-4	5 Consecutive frame with max. recurring 8 Objects
Config-5	5 Consecutive frame with max. recurring 12 Objects

Table 4.18: nuScenes Detection Score and mean True Positive metrics computed for various input configurations to understand the minimum number of sequences required to positively impact the temporal information encoding. Best performance in bold.

Configuration	mAP	mATE	mASE	mAOE	NDS
Baseline	0.972	0.155	0.256	0.908	0.766
Config-1	0.969	0.160	0.259	0.763	0.788
Config-2	0.972	0.140	0.264	0.647	0.811
Config-3	0.969	0.147	0.264	0.785	0.786
Config-4	0.972	0.146	0.281	0.638	0.809
Config-5	0.968	0.166	0.280	0.695	0.794

The nuScenes benchmark, described in Section 2.4.2 of Chapter 2, is used to evaluate the performance of the models. The nuScenes detection score (NDS) and the mean true positive metrics including mean average precision (mAP), mean average translation Error (mATE), mean average scale error (mASE), and mean average orientation error (mAOE) are used to measure the overall performance of the model for various input configurations as depicted in Table 4.18. The NDS values of the model show significant improvement of $\sim 4\%$ from its baseline performance with configuration-2 and configuration-4, which had three and five consecutive frames respectively. Both configurations contained a maximum of eight recurring objects.

The individual class performance depicted in Table 4.19 presents the average precision (AP) and true positive metric score namely average translation error (ATE), average scale error (ASE), and average orientation error (AOE) values to analyse the minimum required frames and the influence of object recurrence. The orientation error for ‘pedestrian’ is lowest with configuration-5 but with the same configuration for ‘car’ the orientation error is higher. These observations are contrary to configuration-3 where the orientation error for ‘car’ is lower and ‘pedestrian’ is higher. However, both of these configurations reduce the error compared to the baseline. The cause of this difference in the observed error in various configurations is likely due to the network seeing pedestrian more frequently in successive frames compared to the cars whose velocity is higher than the pedestrians. This helps the network to improve the estimations leading to improvement in the performance of pedestrian when five consecutive frames are considered while the performance drops with three consecutive frame references. A similar conclusion can be drawn on the performance of ‘car’ which moves much faster than ‘pedestrian’; the performance drops when five consecutive frames are considered but improves with three consecutive frames. To handle this difference in individual class performance, configuration-2 was employed where both perform better than the baseline and the errors are improved during investigations presented in Chapter 5.

Table 4.19: True Positive metrics of various input configurations to understand the minimum number of sequences required to positively impact the temporal information encoding

Configuration	Car				Pedestrian			
	AP	ATE	ASE	AOE	AP	ATE	ASE	AOE
Baseline	0.961	0.220	0.191	1.124	0.983	0.090	0.322	0.692
Config-1	0.958	0.220	0.190	0.832	0.981	0.101	0.328	0.693
Config-2	0.961	0.210	0.172	0.734	0.983	0.070	0.355	0.560
Config-3	0.958	0.227	0.188	0.906	0.981	0.066	0.339	0.663
Config-4	0.961	0.204	0.182	0.829	0.983	0.087	0.381	0.447
Config-5	0.956	0.240	0.192	1.002	0.981	0.093	0.367	0.380

4.3.3 Varying the lidar feature vector size

In this section, the length of the input lidar point cloud data is varied to understand the influence on the final performance of the model and study the individual class performance. The length of the feature vector is varied to have 512, 1024, and 2048 lidar points and the

F-ConvNet is trained with these input vector sizes on KITTI and nuScenes dataset and the hyper parameters were initialised as mentioned in Section 3.2.3. The 3D detection and BEV detection results on the KITTI dataset with varying feature vector lengths are depicted in Table.4.20 and Table. 4.21, respectively. Although a slight improvement in mAP values is achieved with a vector length of 512 for few cases, the best mAP values for all the object categories are obtained using the feature vector length of 1024 with a negligible dip in performance. Hence, a feature vector length of 1024 is considered as the number of lidar points to train the network for further investigations using the KITTI dataset.

Table 4.20: 3D Detection AP (%) on KITTI validation set. Best performance in bold.

Feature Vector Size	Car			Pedestrian			Cyclist		
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
512	84.33	84.36	77.49	72.62	73.25	73.18	95.80	87.12	79.12
1024	88.59	87.79	86.98	73.11	73.03	66.39	87.83	88.03	87.92
2048	76.28	75.87	76.69	71.43	64.88	65.52	87.56	78.14	70.28

Table 4.21: BEV Detection AP (%) on KITTI validation set. Best performance in bold.

Feature Vector Size	Car			Pedestrian			Cyclist		
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
512	97.52	89.22	89.22	83.53	76.35	76.36	97.56	89.50	89.25
1024	97.88	97.93	89.72	75.93	76.02	75.99	88.95	89.00	88.99
2048	88.62	80.22	80.34	73.77	66.86	67.31	88.59	79.30	79.13

Similarly, the influence of feature vector length when the model is trained on nuScenes dataset is assessed. The individual class and overall performance of the F-ConvNet obtained on nuScenes dataset by varying the number of sample points (feature vector length) are depicted in Table.4.22 and Table. 4.23, respectively. Although the overall performance shows the feature vector size 1024 produces best NDS values, when individual class performances are considered a different pattern was observed. The best average precision (AP) and orientation error (AOE) for ‘car’ is achieved with feature vector length 2048 but translation (ATE) and scale (ASE) error are slightly higher. In case of ‘pedestrian’, the best AP and ATE is obtained with vector length 512 and the AOE and ASE values are improved with the vector lengths 1024 and 2048. This difference in behaviour motivated to propose to employ two different vector lengths: 1024 for ‘car’, 512 and 1024 for ‘pedestrian’ during network training. The training of the model with the object classes is done separately as discussed in

Section 4.3.5 and further, with the availability of prior regional proposals, adaptation of two different feature vector lengths for the object was feasible during training and inference to best understand the influence of spatial and temporal data on model performance.

Table 4.22: Performance of Frustum-ConvNet for Car and Pedestrian class on custom nuScenes test set by varying the feature vector size of lidar point cloud data. Best performance in bold.

Feature Vector Size	Car				Pedestrian			
	AP	ATE	ASE	AOE	AP	ATE	ASE	AOE
512	0.958	0.206	0.190	0.991	0.983	0.101	0.346	0.137
1024	0.961	0.203	0.184	0.819	0.981	0.118	0.327	0.264
2048	0.964	0.210	0.191	0.801	0.981	0.215	0.390	0.126

Table 4.23: Overall Performance of Frustum-ConvNet on custom nuScenes test set by varying the feature vector size of lidar point cloud data. Best performance in bold.

Feature Vector Size	mAP	mATE	mASE	mAOE	NDS
512	0.971	0.153	0.267	0.564	0.821
1024	0.971	0.160	0.255	0.541	0.826
2048	0.972	0.212	0.290	0.463	0.825

4.3.4 Lidar Single-Sweep Versus Multi-Sweep

After varying the feature vector size, the next interesting aspect considered was employing single sweep or multiple sweeps lidar point cloud data for representing a scene. The nuScenes dataset consists of single sweep and multiple sweep lidar data that can be accessed and saved to the local disk using the development toolkit published in the nuScenes repository¹. Fig. 4.9 illustrates the representation of a scene using single-sweep and multi-sweep lidar pointclouds. The multiple sweep lidar data is provided as an additional feature option to represent a scene by previous lidar sweeps by moving all pointclouds to the coordinate system of the keyframe and appending a scalar time-stamp to each point indicating the time delta in seconds from the keyframe. Technically, Multiple sweeps can be considered temporal data. However, in the nuScenes dataset, the annotation is only available for key frames the other lidar timestamp data is accumulated together and assigned to the timestamp which requires multiple sweep representation. This causes the overpopulation of the point outside

¹<https://github.com/nutonomy/nuscenes-devkit>

the ground truth annotation which affects the training process and hence the estimation will be erroneous for a few cases.

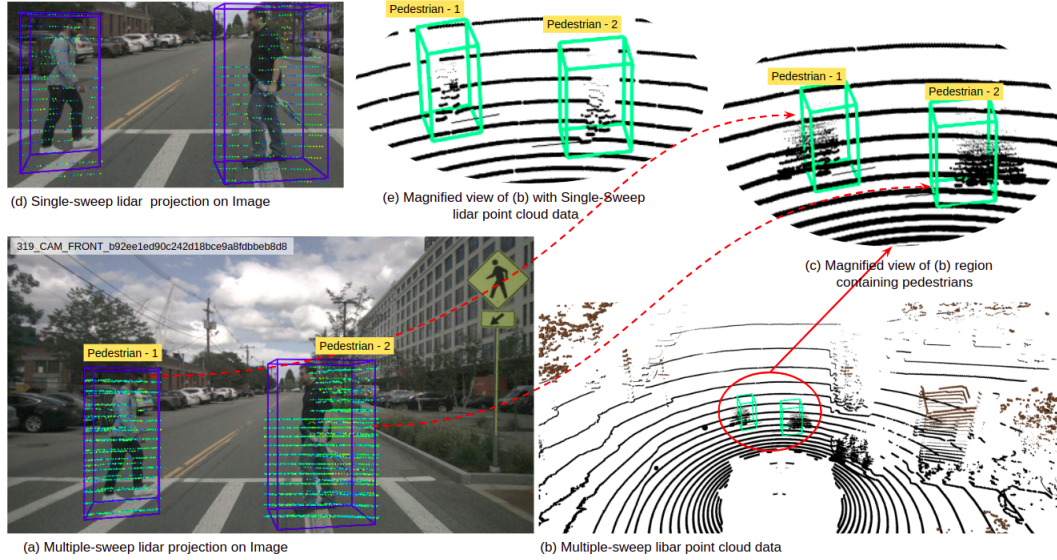


Figure 4.9: Visual Comparison of single-sweep vs multi-sweep lidar point cloud data

Although the use of multi-sweep lidar data encloses more points within the region proposal, there are points that are scattered beyond the regions as shown in Fig. 4.9-(c). Due to this out-of-bound point distribution, representing the object causes the detector to accumulate geometric misalignment issues, particularly for pedestrians, making the estimation process ineffective compared to single-sweep, which has the point distribution within the region proposal (see Fig. 4.9-(e)). Table 4.24 shows the performance of F-ConvNet for the Car and Pedestrian class on the custom nuScenes test dataset and the overall performance is presented in Table. 4.25. From the obtained results, it can be concluded that the overall performance and mean true positive metrics scores of the detector with multiple-sweep lidar data produce inferior results and are depicted in Table. 4.23. However, when the individual class performance, depicted in Table 4.24 and Table. 4.22, are compared, the performance of class ‘Car’ is improved in contrast to ‘Pedestrians’ as it contains the point distribution pattern within the region proposal and has larger surface coverage compared to Pedestrian.

Based on the obtained results, utilising multi-sweep lidar point cloud for representation of the object is not effective for the current study. Further, multi-sweep lidar data consumes more memory compared to single-sweep, which can be a bottleneck during the design of architectures to incorporate temporal information and apply data fusion strategies to fuse

multimodal information. Hence, a single-sweep lidar point cloud is considered for further exploration in this study.

Table 4.24: Performance of F-ConvNet for Car and Pedestrian class on custom nuScenes test set by varying the feature vector size of multi-sweep lidar point cloud data. Best performance in bold.

Feature Vector Size	Car				Pedestrian			
	AP	ATE	ASE	AOE	AP	ATE	ASE	AOE
512	0.967	0.181	0.187	0.774	0.983	0.124	0.311	2.747
1024	0.964	0.186	0.187	0.711	0.981	0.138	0.283	0.732

Table 4.25: Overall Performance of Frustum-ConvNet on custom nuScenes test set by varying the feature vector size of multi-sweep lidar point cloud data. Best performance in bold.

Feature Vector Size	mAP	mATE	mASE	mAOE	NDS
512	0.975	0.152	0.249	1.760	0.754
1024	0.972	0.162	0.235	0.721	0.799

4.3.5 Training: Individual Class vs All-together

Wang and Kui [85] authors of F-ConvNet considered KITTI dataset with ‘car’, ‘pedestrian’ and ‘cyclist’ class the performance were evaluated on validation set as follows. Firstly, the network was trained with only the car class and secondly, the pedestrian and cyclist classes were trained together. While training for cars, 4 frustum resolutions represented as (‘u’) are assigned with values [0.5, 1.0, 2.0, 4.0] and the corresponding scale (‘s’) are set to [0.25, 0.5, 1.0, 2.0] with frustum width (‘d’) equal to [128, 128, 256, 512], frustum level feature vector length (L) is set to 280, and the higher level frustum feature vector length (\tilde{L}) is specified to 140 when features map were upsampled during de-convolution operation. Similarly, while training the network with pedestrian and cyclist class, 4 frustum resolutions: $u = [0.2, 0.4, 0.8, 1.6]$ and $s = [0.1, 0.2, 0.4, 0.8]$ with $d = [128, 128, 256, 512]$, $L = 700$, and $\tilde{L} = 350$. During the investigation the baseline model was trained in two styles: firstly, train with all the classes together and secondly, train the classes as followed in [85]. While training all the classes together, the frustum resolutions were initialized with: $u = [0.2, 0.4, 0.8, 1.6]$ and $s = [0.1, 0.2, 0.4, 0.8]$ with $d = [128, 128, 256, 512]$, $L = 700$, and $\tilde{L} = 350$. The performance of 3D detection and BEV detection in terms of mean average precision (mAP) as percentage

(%) for car, pedestrian, and cyclist classes for easy, moderate, and hard cases are depicted in Table 4.26. The performance of the pedestrian and cyclist classes were improved when these two classes are trained together in contrast to when all three classes are trained together. However, the mAP values of the cyclist for hard cases are better when the classes are trained separately and mAP of car classes is slightly better when the classes are trained together. Based on these observations the object class are trained separately and results are reported.

Table 4.26: 3D Object Detection and BEV Detection in AP (%) when trained classes separately vs together on KITTI validation set. Best performance in bold.

Feature Vector Size	Car			Pedestrian			Cyclist		
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
<i>3D Detection</i>									
Trained-together	88.59	87.79	86.98	73.11	73.03	66.39	87.83	88.03	87.92
Trained-separately	86.08	86.04	86.45	81.32	75.16	75.64	94.02	88.33	86.38
<i>BEV Detection</i>									
Trained-together	97.88	87.93	89.72	75.93	76.02	75.99	88.95	89.00	88.99
Trained-separately	97.77	89.26	89.38	84.89	85.47	78.00	97.19	97.86	89.15

The baseline model was also trained on the nuScenes dataset with car and pedestrian classes. The results are depicted in the Table. 4.27 and the performance of the model is better, with improved true positive metric scores and nuScenes detection metric values when the classes are trained separately. The individual class results are collated for the final evaluation of results compared to the training of the classes together. Based on this inference the model will be trained with classes separately and collate the results to evaluate the individual performance and overall performance of the model. Further investigations related to temporal feature extraction and multi-modal data fusion strategies are therefore carried out using individual class training.

Table 4.27: Overall Performance of Frustum-ConvNet when trained classes separately vs together on custom nuScenes test set. Best performance in bold.

Method	mAP	mATE	mASE	mAOE	NDS
Trained-together	0.971	0.177	0.261	0.598	0.812
Trained-separately	0.971	0.161	0.255	0.541	0.825

4.4 Summary

This chapter explores some widely used representations of the lidar point cloud, such as bird-eye-view, range-view, point-pillars, and voxel format on KITTI and nuScenes datasets. The data representation assisted in understanding the complexity involved in handling the data while training and developing the initial baseline performance on the chosen F-ConvNet architecture. In addition, this chapter presents various investigations carried out to select the training parameters while designing the framework for incorporating temporal information into the 3D detection network. Finally, the experimental results were instrumental for designing pipelines to facilitate the extraction of spatio-temporal information from the raw point cloud data or fused multimodal data. The investigation included:

1. The prior localization of the objects using an off-the-shelf pre-trained 2D detector using the KITTI dataset assisted the research in studying the influence of priors on final accuracy in AP (%). The network achieved maximum mAP with the 2D detector confidence score set between 0.1 to 0.001 and the IoU threshold value equal to 0.95. However, this also increased the average computation time to process the frame as the number of region proposals passed to the network increased. Furthermore, the F-convNet framework relies on these prior region proposals for estimating the objects' final 3D bounding boxes, creating a bottleneck in employing a 2D detector to detect all the objects appearing in the scene to test the final accuracy of the 3D detection framework. To overcome this aspect for further investigations, the ground truth annotations with augmentations are used as preliminary region proposals to cover all the objects appearing in the current and consecutive sequences.
2. Visual analysis of the raw lidar point cloud data in the sequenced frames of the nuScenes dataset assisted in understanding the role of temporal information while training the network to learn the structural information of the object from the point cloud distribution. Furthermore, this helped to understand the benefits of keeping track of variations in the point cloud distribution as the AV acquires the data for estimating the target position in the scene. Tracking the variation will assist the deep

learning models to handle misalignment issues in the data while estimating box parameters.

3. The RNN-based CLSTM and CGRU were incorporated into the detection network to extract the temporal information. The performance of CLSTM was best compared to the CGRU during the evaluation of the KITTI dataset. Therefore, CLSTM was employed to analyse the minimum sequenced data required to effectively extract the temporal information. Furthermore, the analysis showed that the model with three consecutive frames with a maximum of eight recurring objects performs better than other configurations considered in this study. Therefore, the configuration with three consecutive frames with a maximum of eight recurring objects was selected for further investigations detailed in Chapter 5 and Chapter 6.
4. The multi-sweep lidar point cloud data in the nuScenes dataset appeared promising during the visual analysis of the data. However, a comparison of performance between single-sweep and multi-sweep data showed that the true positive error metric values increased, reducing the model's overall performance.
5. The feature vector size used to represent the object class was one of the parameters affecting the model performance. This was analysed by varying the feature vector length of lidar data for the object under consideration in both KITTI and nuScenes datasets. As a result, the 'car' class achieved maximum performance with a feature vector length of 1024, while for 'pedestrian' and 'cyclist', the feature vector length of 512 produced the highest performance.

The experimental results obtained in this chapter were instrumental for designing effective pipelines to facilitate the extraction of spatio-temporal information from the raw point cloud data or fused multi-modal data used in the next chapters.

Chapter 5

Temporal Encoding

The work presented in this chapter is published in Venkatesh G.M, et al. (2022) [4].

Lidar sensors provide the scene's spatial depth that can assist the deep learning model to estimate the parameters related to objects in the scene. Another aspect worth considering is to study the variations in the lidar point cloud distribution as the position and appearance of the object in a scene. The process of understanding and monitoring the object information over a sequence of frames can be referred to as temporal information, which can assist the deep learning-based model during regression tasks. Spatial image features along with temporal information (i.e., timestamp, video frame number, spatial feature variation over time) are extensively used in computer vision applications such as 2D object detection and tracking in videos, for human pose estimation and activity recognition [46], flow predictions [119], video compression and 2D detection based sliding window for path prediction [34].

Most state-of-the-art 3D object detection methods are based on a single-frame reference, and limited focus has been given to utilising temporal aspects in the data for 3D object detection. This limited focus was due to the non-availability of a large-scale, sequentially annotated, data sequence required for extracting temporal information from multiple frame references and various modalities. The recent release of datasets like nuScenes [32] and waymo [53] has drawn considerable interest from researchers to employ multi-frames to leverage both spatial and temporal information to establish coherence between the object and scene and more accurately estimate the depth and other attributes of the object. However, as discussed in Section 2.2.2 of Chapter 2 there is still limited research about extracting temporal information using lidar-only data and its potential use in understanding depth estimation for improved localisation and detection accuracy [16, 38]. The dynamic nature of objects

encountered during complex driving environments has motivated to propose the pipeline incorporating both spatial (lidar data) and temporal feature extraction using convolutional-LSTMs, along with an attention mechanism into the object detection architecture, to gain valuable insight into the influence of spatio-temporal feature during 3D estimations. The results obtained in [4] show that the inclusion of temporal feature extracted from the sequenced lidar point clouds considerably improves the true positive metric, specifically the orientation error of the 3D bounding box from 0.819 to 0.463 and 0.294 to 0.107 for ‘cars’ and ‘pedestrian’ classes, respectively, on the customised subset of nuScenes training dataset. The overall nuScenes detection score (NDS) was also improved by $\sim 4\%$ compared to the reference baseline.

The chapter is structured as follows: firstly, a brief description of the proposed approach for exploring the influence of temporal information on object detection is presented in Section 5.1 followed by the experimental setup, extension to the work presented in [4] and analysis of the results are discussed in Section 5.2. Finally, the chapter concludes (Section 5.3) with the significance of utilising temporal information during the estimation of oriented 3D boxes and its potential usage in further investigations using multiple modalities to train and infer the network for 3D object detection.

5.1 Incorporation of Spatio-temporal Information

The proposed architecture shown in Fig. 5.1 is inspired by methods from Wang et al. [85] and Erçelik, Yurtsever & Knoll [10]. These methods use data representation to handle sparse point cloud data processing by localising the region from the 2D proposal (drawn from image space using 2D detector) and then employed to generate a sequence of frustum by sliding along the frustum axis as shown in Fig. 5.2 and the process is repeated for each of the proposals in the RGB image. The formulation to estimate the 3D bounding boxes, $B = \{b_i \mid i = 0, 1, 2, \dots, N\}$, where $b_i \in \mathbb{R}^7$ and N is the number of objects in the scene, are parameterised as: box centre coordinates, (x, y, z) ; box dimension, (h, w, l) , representing the height, width and length of each box respectively; and the orientation or heading angle, θ , described using the raw point cloud data, $P = \{p_i \mid i = 0, 1, 2, \dots, N\}$, where $p_i \in \mathbb{R}^{3 \times M}$ and M is the total number of raw points. The 2D proposal from the image space, $D =$

$\{d_i \mid i = 0, 1, 2, \dots, N\}$, where $d_i \in \mathbb{R}^4$ and N is the number of objects in the scene are used to generate the Frustum Point, $F = \{f_i \mid i = 0, 1, 2, \dots, N\}$, where $f_i \in \mathbb{R}^{3 \times L}$ and L is the number of points inside the frustum.

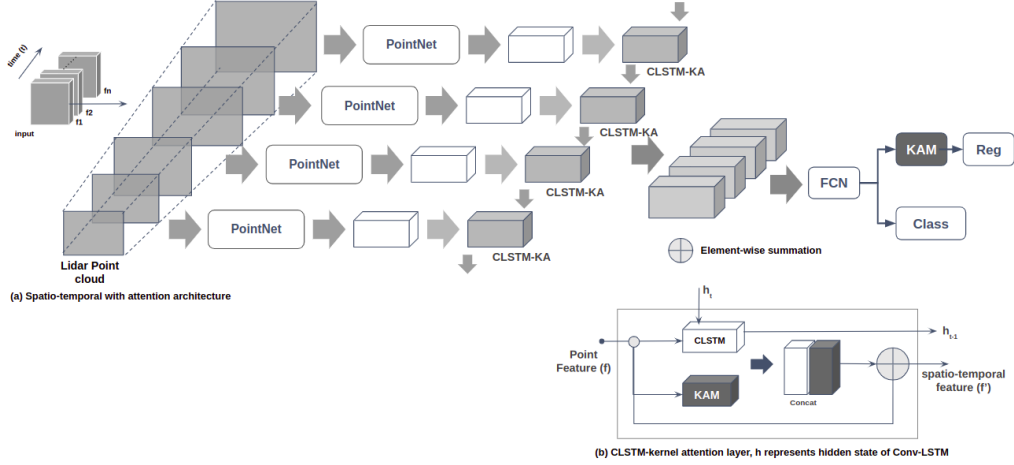


Figure 5.1: Proposed architecture to incorporate spatio-temporal feature into F-ConvNet architecture

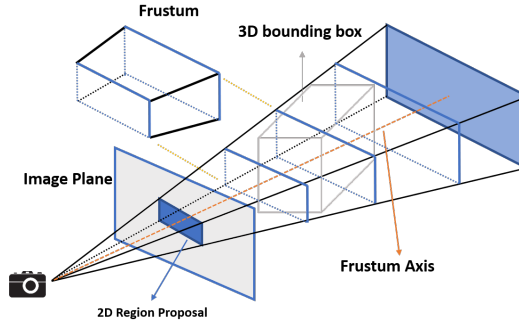


Figure 5.2: Sequence of frustums generated for a region proposal in F-ConvNet architecture [85]

The data takes a whole new dimension when associated over time or with the sequence length of an object. The sequentially referencing sensor data and the object from its first appearance until its exit (“sequence death”) enable the extraction of temporal aspects in the data. Frustum-ConvNet [85] is employed as the baseline architecture and is detailed in Section 3.2.3. The network is trained with a single-frame reference on a subset of the nuScenes data described in 5.2.1. Same loss functions as in the original Frustum-ConvNet architecture are employed, where Euclidean distance-based regression loss is used for the box centres, and smooth l1 regression loss is used for box sizes and the angle. The focal loss is for the object classification and corner loss is used to regularize box regression of

all parameters. The remainder of this section describes the baseline architecture and the methods used to encode temporal information – frame stacking (5.1.1) and Convolution-LSTM (5.1.2) – and finally the Kernel Attention Module (5.1.3) for capturing attention.

5.1.1 Frame Stacking

Frame stacking is the most straightforward approach employed to capture the temporal information where the sequence of frames is stacked without shuffling with the intention that the network is directly exposed to learn the variations in context and the objects as it appears in the data. All the frustums containing the objects extracted using 2D region proposals are arranged as they appear in the successive frames and presented as a single input sequence as a batch to the detection network during training. The network is trained using the same loss function employed in the baseline architecture. Mostly, while training deep learning models the input data is shuffled to reduce overfitting and variance. In the frame stacking approach, the sequence of the frames is maintained as it appears in the scene only the object sequence tracks are shuffled after applying augmentation. This approach relies entirely on the network to encode the temporal information based on the changes it sees in the object information in successive frames over time.

5.1.2 Convolution-LSTM

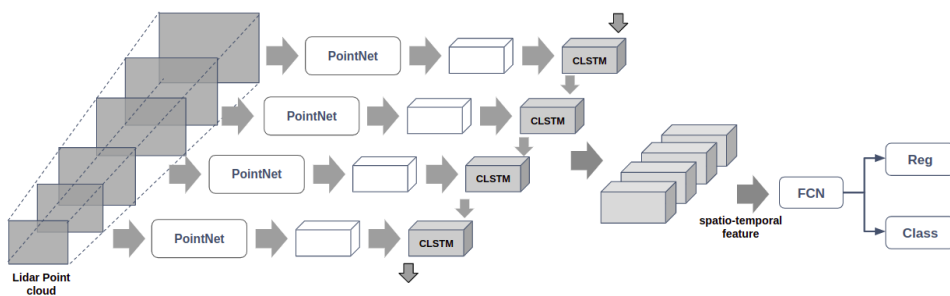


Figure 5.3: Spatio-temporal encoding using convolutional-LSTMs

The network architecture to encode temporal information using convolutional-LSTMs (CLSTM) is shown in Fig. 5.3. A CLSTM layer is placed between the PointNet and fully convolutional network (FCN) block to learn spatial and temporal information from the input sequence. The CLSTM module consists of one input layer and five hidden layers with

dimensions similar to the output feature map size of the four frustum resolutions of the PointNet architecture, having values 128, 128, 256, and 512. The output from the CLSTM module is passed to the FCN in two different configurations: firstly, directly passing the final output of the CLSTM module and secondly, concatenating the output from the first hidden layer with the last one (similar to a skip connection). The network is trained using the same data sequence as the frame stacking approach. The extracted temporal feature and output of the CLSTM are shared between the frames appearing in later sequences to compensate for any loss of information during learning due to occlusion or perspective changes.

Table 5.1: Fully convolutional Network (FCN) configuration used in Frustum-ConvNet-CLSTM configuration 2 architecture

Name	Kernel size/ Filter No./Striding/Padding
Block - 1	3 x 128 / 256 / 1 / 1
Block - 2	3 x 128 / 128 / 2 / 1 3 x 128 / 128 / 1 / 1
Block - 3	3 x 128 / 256 / 2 / 1 3 x 256 / 256 / 1 / 1
Block - 4	3 x 256 / 512 / 2 / 1 3 x 512 / 512 / 1 / 1
Deconv - 2	1 x 128 / 256 / 1 / 0
Deconv - 3	2 x 256 / 256 / 2 / 0
Deconv - 4	4 x 512 / 256 / 4 / 0
MergeConv - 2	1 x 384 / 128 / 1 / 0
MergeConv - 3	1 x 768 / 256 / 1 / 0
MergeConv - 4	1 x 1536 / 512 / 1 / 0

The results obtained in the initial trials using the first configuration of CLSTM are published in Venkatesh G.M et al.(2022) [4]. In the trials involving the second configuration of CLSTM, the FCN layers are initialized with the values presented in Table. 5.1. The true positive metric scores obtained using both CLSTM configurations for individual classes are depicted in Table. 5.2 and overall performance of the models are depicted in Table. 5.3. The depicted values in the tables show the performance of the ‘car’ and ‘pedestrian’ with the second configuration of the CLSTM model reduced the mean true positive error for both the ‘car’ and ‘pedestrian’ classes. The nuScenes detection metrics are captured in Table. 5.3 also draws the same conclusion as the NDS value obtained with the second configuration of CLSTM improved by $\sim 4\%$. The results of the proposed spatio-temporal architectures using

the second configuration of the CLSTM module are presented at the end of this chapter. The same configuration is employed during the investigation of the fusion strategies presented in Chapter 6.

Table 5.2: True positive metric scores for Cars and Pedestrians with multi-frames reference using CLSTM on the subset of nuScenes data. Best performance in bold.

Method	Cars				Pedestrians			
	<i>AP</i>	<i>ATE</i>	<i>ASE</i>	<i>AOE</i>	<i>AP</i>	<i>ATE</i>	<i>ASE</i>	<i>AOE</i>
CLSTM-Config-1	0.958	0.220	0.190	0.832	0.981	0.101	0.328	0.693
CLSTM-Config-2	0.958	0.129	0.164	0.463	0.983	0.101	0.297	0.692

Table 5.3: nuScenes metric computed for CLSTM configurations with multi-frames reference on subset nuScenes data. Best performance in bold.

Method	<i>mAP</i>	<i>mATE</i>	<i>mASE</i>	<i>mAOE</i>	<i>NDS</i>
CLSTM-Config-1	0.9669	0.1601	0.2590	0.7634	0.7887
CLSTM-Config-2	0.9705	0.1150	0.2305	0.5775	0.8314

5.1.3 Kernel Attention Mechanism Module

The purpose of an attention mechanism is to aid the network in focusing on the salient regions of the point cloud. A kernel attention mechanism (KAM) shown in Fig. 5.4 with three configurations focusing on the placement of the attention mechanism module into the detection network presented in Fig. 5.5 is employed.

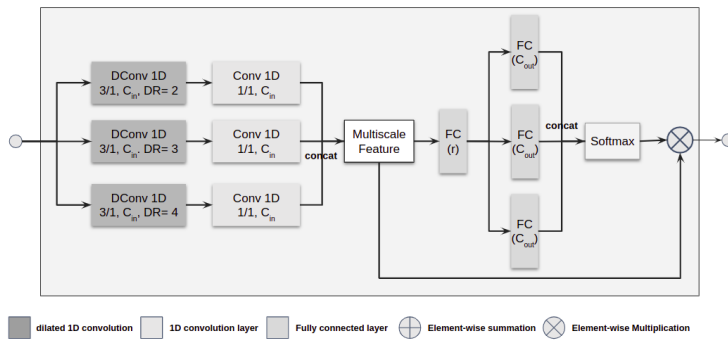


Figure 5.4: Kernel Attention Mechanism (KAM) module

These configurations were considered to investigate the effect of the inclusion of an attention mechanism to focus on the salient lidar points inside the prior region proposal and any resulting improvement in network performance. Firstly, the KAM module is placed just before the regression branch, as shown in Fig. 5.5a, where the features from the FCN module are passed to the KAM module before passing to the regression module. Secondly, the KAM module is placed before the PointNet module as shown in Fig. 5.5b where the input is directly passed to the KAM to learn the salient raw point distributions and another KAM module is placed at the regression branch similar to configuration-1. Lastly, the KAM is placed between the PointNet and FCN, as shown in Fig. 5.5c. Here, the processed data from the PointNet is passed to the KAM to learn the salient features within the feature map contributing to the final estimation process as followed in configurations 1 and 2; a KAM module is placed in the regression branch.

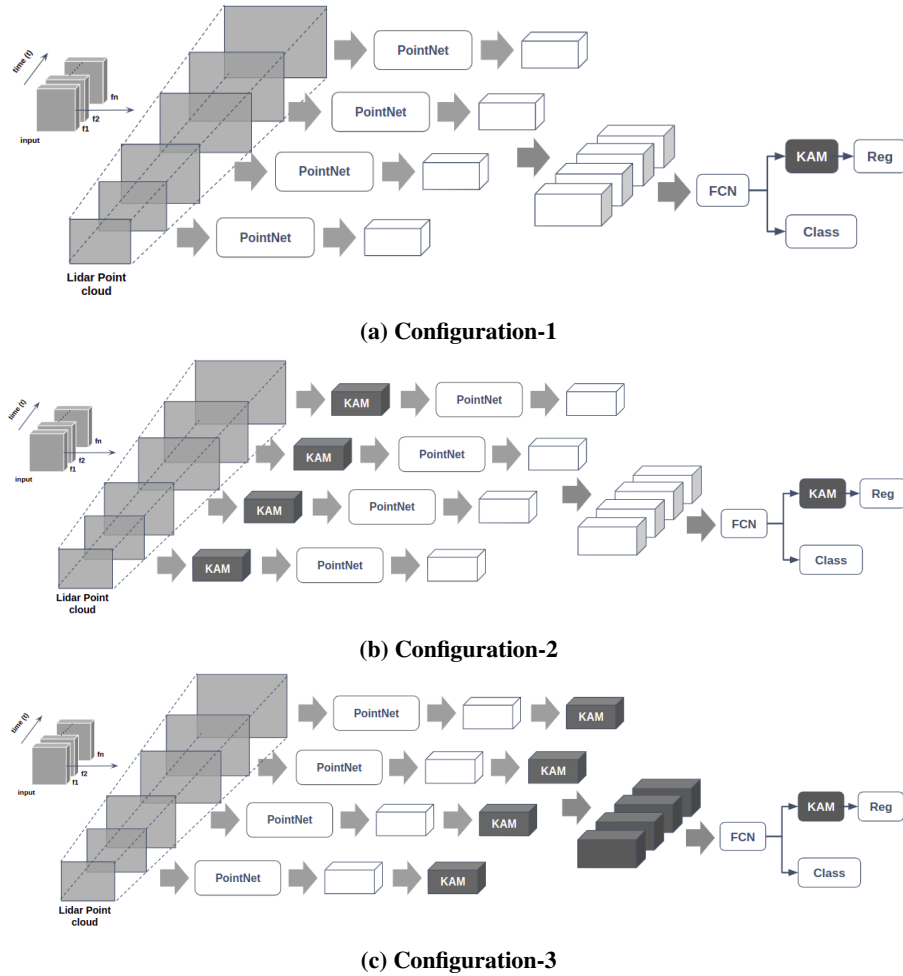


Figure 5.5: Architecture to extract spatio-temporal feature using KAM module

The input vector passed to the KAM module is represented as $f \in \mathbb{R}^{C_{in} \times d}$, where C is the number of channels and d is the length of the vector. Three new tensors are generated namely D_1 , D_2 and D_3 as in equation (5.1) using three 1D dilated convolution (DConv) layers with kernel size = 3, stride length = 1 and different dilation rates of 2, 3 and 4 respectively. Each of these outputs is then passed to the 1D convolution (Conv) to obtain a spatial feature vector that is then concatenated to form a multi-scale feature vector, $f_m \in \mathbb{R}^{C_{in} \times d}$ as in equation (5.2).

$$\begin{aligned} D1 &= \text{Conv}_1(\text{DConv}_1(f)) \\ D2 &= \text{Conv}_2(\text{DConv}_2(f)) \\ D3 &= \text{Conv}_3(\text{DConv}_3(f)) \end{aligned} \tag{5.1}$$

$$f_m = \text{Concat}(D1, D2, D3) \tag{5.2}$$

f_m is fed to a fully connected (FC) layer to obtain vector $f_m' \in \mathbb{R}^{r \times C_{in}}$. To generate a feature vector namely S1, S2 and S3 for each of the channels in f_m' the dimension of these channels is reduced to learn the relationship among features with different dilation rates. Each of the channel vectors, computed as in equation (5.4) of f_m' , is passed to an FC layer, as in equation (5.3), to generate an attention vector S_A across the channels $S \in \mathbb{R}^{C_{out} \times r}$ whose weights are computed by a softmax function as shown in equation (5.5).

$$f_m' = \text{FC}_r(f_m) \tag{5.3}$$

$$\begin{aligned} S1 &= \text{FC}_1(f_m') \\ S2 &= \text{FC}_2(f_m') \\ S3 &= \text{FC}_3(f_m') \end{aligned} \tag{5.4}$$

$$S_A = \text{softmax}\{\text{Concat}(S1, S2, S3)\} \tag{5.5}$$

Finally, the refined feature map are obtained as $F_{KA} \in R^{C_{out} \times r}$ of channel attention by multiplying S_A and f_m :

$$f_{KA} = S_A \otimes f_m \quad (5.6)$$

Table 5.4: Performance of the Car and Pedestrians class using multi-frames with Kernel Attention Mechanism (KAM) on subset of nuScenes data. Best performance in bold.

Method	Cars				Pedestrians			
	<i>AP</i>	<i>ATE</i>	<i>ASE</i>	<i>AOE</i>	<i>AP</i>	<i>ATE</i>	<i>ASE</i>	<i>AOE</i>
KAM-Config-1	0.681	0.677	0.215	1.154	0.478	0.878	0.343	1.156
KAM-Config-2	0.931	0.203	0.179	0.920	0.972	0.113	0.327	2.640
KAM-Config-3	0.956	0.232	0.200	0.975	0.972	0.097	0.360	0.416

Table 5.5: Performance of the KAM configurations using multi-frames reference on subset nuScenes data. Best performance in bold.

Method	<i>mAP</i>	<i>mATE</i>	<i>mASE</i>	<i>mAOE</i>	<i>NDS</i>
KAM-Config-1	0.5792	0.7776	0.2792	1.3588	0.4468
KAM-Config-2	0.9513	0.1579	0.2532	1.7801	0.7404
KAM-Config-3	0.9639	0.1645	0.2800	0.6957	0.7919

The extracted point-wise features inside each of the sequence frustums to generate the frustum points is represented as $F = \{f_i \mid i = 0, 1, 2, \dots, N\}$, where $f_i \in \mathbb{R}^{3 \times d}$. The number of points (d) inside each of the sequence frustum is aggregated to extract point-wise features to form a frustum-level feature vector which is represented as $\{\mathbf{f}_{i=1}^L\}$, with $\mathbf{f}_i \in \mathbb{R}^d$. These feature vectors, re-formed as 2D feature maps of the size $L \times 3$, are used as input to the subsequent fully convolutional network (FCN) shown in Fig. 3.13b for continuous frustum-level feature fusion. The detection block includes classification and regression branches added on top of FCN, as shown in Fig. 5.1 and Fig. 5.5. A KAM module is added in the regression branch to learn the most salient feature map to further aid the model while estimating the oriented 3D box parameters.

The network architecture with kernel attention only is shown in Fig. 5.5 and Fig. 5.1 presents the proposed spatio-temporal architecture using CLSTM with the attention mechanism to aid the network in propagating temporal information extracted from the input data

sequence. The model further benefits from using an attention mechanism to focus on spatially distributed points defining the object and the context associated with the scene. Both architectures are trained with the same sequences of data as employed in the frame stacking approach.

5.2 Experiments

5.2.1 Dataset

To study the influence of temporal information on the detection network, the nuScenes[32] dataset described in Section 4.1.2 of Chapter 4 is considered. Five of the available ten subsets in the nuScenes dataset are considered. The first three subsets, comprising 255 scenes with 10,120 samples, are considered the training set and the following two subsets as the test set, which comprises 170 scenes with 6,854 samples. The training data is further divided into training and validation with 7,180 and 2,940 samples, respectively. The experimental trials focused on ‘car’ and ‘pedestrian’ classes and footage from the front camera and considered lidar-only data with $\approx 64K$ annotations. Furthermore, for consistency and ease of comparison to KITTI-like data, all annotations are converted to the KITTI 3D object detection format [139] using the development toolkit provided by the nuScenes repository.

5.2.2 Setup

The models were trained on a single Nvidia-TitanX GPU with 12GB RAM for 75 epochs using the adam optimizer[106] from scratch without using any pre-trained model weights. The learning rate and weight decay values were set to 0.001 and 0.0001, respectively, with a batch size of 16 for single-frame reference mode. During multi-frame reference mode, three consecutive frames and a batch size of 4 without shuffling the input sequence are considered for training the model.

The region proposals are obtained from the ground-truth annotation and are augmented with a translation and scaling factor during training. The number of sample points for each region proposal is set to 1,024 and 512 for ‘car’ and ‘pedestrian’, respectively, whose values are selected randomly depending on the trial configurations. Same as [85], 4 frustum

resolutions of [0.2, 0.5, 1.0, 2.0] and stride value set to [0.1, 0.25, 0.5, 1.0] for the ‘car’ category and 4 frustum resolutions of [0.2, 0.4, 0.8, 1.6] and stride value set to [0.1, 0.2, 0.4, 0.8] for the ‘pedestrian’ are considered. During the inference, the performance of the methods is evaluated using the nuScenes metric discussed in Section 5.2.3 on the custom nuScenes test data. Both single-frame and multiple-frame reference modes are considered to understand their effect on the model performance.

5.2.3 Metrics

The performance of the model is evaluated using nuScenes metrics [32] which include: Average Precision (AP) (equation (5.7)) and True Positive (TP) (equation (5.8)) for car and pedestrian classes on the custom test data. While computing the TP metric, Average Translation Error (ATE), Average Scale Error (ASE) and Average Orientation Error (AOE) are reported. While converting the nuScenes annotation to KITTI, the velocity component in the data is not captured, making the Average Velocity Error (AVE) computation not feasible with the current setup. Since ground truth 2D bounding boxes are used as the region proposals, Average Attribute Error (AAE) is also ignored as the network will have prior knowledge about the class information and since all the estimation is made for every input frame all the objects (in motion or static) are considered for estimation. nuScenes detection score (NDS) is calculated using equation(5.9) after computing mAP and mTP using equation(5.7) and equation (5.8) respectively. In equation (5.7) and equation (5.8), D represents distance threshold in meters, $\{0.5m, 1m, 2m, 4m\}$ and C is the set of classes $\{cars, pedestrians\}$ considered for evaluation.

$$\text{mAP} = \frac{1}{|C||D|} \sum_{c \in C} \sum_{d \in D} AP_{c,d} \quad (5.7)$$

$$\text{mTP} = \frac{1}{|C|} \sum_{c \in C} TP_c \quad (5.8)$$

$$\text{NDS} = \frac{1}{6} [3\text{mAP} + \sum_{\text{mTP} \in \mathbb{TP}} (1 - \min(1, \text{mTP}))] \quad (5.9)$$

Here mAP is mean Average Precision, defined in equation (5.7), and \mathbb{TP} is the set of three mean True Positive metrics presented in equation (5.8) obtained by averaging mATE, mASE, and mAOE with all the metric values are bound between the range 0 to 1. A weight of 3 to mAP and 1 to each of the three TP scores are assigned to obtain the normalized sum value before computing the nuScenes Detection Metric (NDS) defined in equation (5.9).

Mean Average Precision (mAP), as used in the KITTI benchmark, can only provide a general understanding of the model’s performance and fails to capture the other aspects causing and affecting the model’s performance. However, nuScenes metrics aid in analysing the impact of encoding temporal information and the use of the attention mechanism. To propose a pipeline capable of giving accurate 3D positioning and orientation of dynamic objects, the orientation error (mAOE) is also considered for analysis. NDS also provides a more nuanced metric for comparing the methods.

5.2.4 Results and Discussion

The performance of the baseline and the proposed methods incorporating temporal information through frame stacking (FS), convolutional-LSTMs (CLSTM) and kernel attention mechanism (KAM) models evaluated for ‘car’, and ‘pedestrian’ classes on the subset of the nuScenes test data are presented in Table. 5.6. Table. 5.7 presents the performance in terms of mean average errors and the nuScenes detection score (NDS) of all models employed in this study.

The results show that temporal information does play an important role in influencing the model while estimating the 3D box parameters. While the mAP values are slightly lower, the mAOE and NDS values show improvements when temporal information (FS or CLSTM) are included and increase again when the kernel attention Mechanism (KAM) module is incorporated. Note the $\sim 17\%$ improvement over the baseline in the orientation error (mAOE) when CLSTM and KAM are applied with the multi-frame reference method. The Frustum-ConvNet model incorporating CLSTM and CLSTM-KAM modules was trained again with the multi-frame references method using the second configuration of CLSTM modules described in Section 5.1.2. The true positive metric score of the ‘car’ was improved and boosted the model’s overall performance depicted in Table. 5.8 and Table. 5.9. The NDS values of

Table 5.6: True positive metric scores for cars and pedestrians on single frame ('s') and multi-frames ('m') reference using frame stacking (FS), convolutional-LSTMs (CLSTM) and Kernel Attention Mechanism (KAM) with the Baseline Frustum-ConvNet method on subset of nuScenes data.

Model	Cars				Pedestrians			
	<i>AP</i>	<i>ATE</i>	<i>ASE</i>	<i>AOE</i>	<i>AP</i>	<i>ATE</i>	<i>ASE</i>	<i>AOE</i>
Baseline-s	0.961	0.203	0.184	0.819	0.981	0.118	0.327	0.264
Baseline-m	0.961	0.203	0.184	0.819	0.975	0.118	0.326	0.294
FS-s	0.956	0.227	0.199	0.989	0.983	0.116	0.306	1.494
FS-m	0.950	0.227	0.197	0.970	0.983	0.116	0.306	1.494
CLSTM-s	0.958	0.220	0.190	0.832	0.981	0.101	0.328	0.693
CLSTM-m	0.958	0.213	0.184	0.744	0.981	0.104	0.328	0.692
FS-KAM-s	0.961	0.210	0.182	0.924	0.983	0.116	0.297	0.862
FS-KAM-m	0.956	0.232	0.200	0.975	0.972	0.097	0.360	0.416
CLSTM-KAM-s	0.958	0.220	0.191	0.840	0.978	0.107	0.351	0.143
CLSTM-KAM-m	0.958	0.219	0.188	0.784	0.975	0.107	0.351	0.111

Table 5.7: nuScenes metric computed on single frame ('s') and multi-frames ('m') references using frame stacking (FS), convolutional-LSTMs (CLSTM) and Kernel Attention Mechanism (KAM) with the Baseline Frustum-ConvNet method on subset nuScenes data. Best performance in bold.

Method	<i>mAP</i>	<i>mATE</i>	<i>mASE</i>	<i>mAOE</i>	<i>NDS</i>
Baseline-s	0.9708	0.1606	0.2554	0.5415	0.8258
Baseline-m	0.9681	0.1607	0.2552	0.5567	0.8219
FS-s	0.9694	0.1714	0.2525	1.2414	0.7474
FS-m	0.9667	0.1713	0.2514	1.2321	0.7462
CLSTM-s	0.9694	0.1604	0.2589	0.7625	0.7878
CLSTM-m	0.9694	0.1569	0.2560	0.7079	0.7979
FS-KAM-s	0.9722	0.1631	0.2393	0.8929	0.7702
FS-KAM-m	0.9639	0.1645	0.2800	0.6957	0.7919
CLSTM-KAM-s	0.9681	0.1637	0.2708	0.4916	0.8297
CLSTM-KAM-m	0.9681	0.1629	0.2697	0.4474	0.8378

the model were improved by $\sim 5\%$ for convolutional-LSTM and $\sim 4\%$ for convolutional-LSTM and kernel attention mechanism compared to the initial configuration performance presented in Table. 5.7.

Table 5.8: True positive metric scores for cars and pedestrians on multi-frames ('m') reference incorporating CLSTM-config-2 and Kernel Attention Mechanism (KAM) on subset of nuScenes data. Best performance in bold

Method	Cars				Pedestrians			
	AP	ATE	ASE	AOE	AP	ATE	ASE	AOE
CLSTM-2-m	0.958	0.129	0.164	0.463	0.983	0.101	0.297	0.692
CLSTM-2-KAM-m	0.958	0.119	0.161	0.563	0.983	0.103	0.287	0.107

Table 5.9: nuScenes metric computed with multi-frames ('m') reference incorporating CLSTM-config-2 and Kernel Attention Mechanism (KAM) on subset nuScenes data. Best performance in bold.

Method	mAP	mATE	mASE	mAOE	NDS
CLSTM-2-m	0.971	0.115	0.231	0.578	0.8314
CLSTM-2-KAM-m	0.971	0.111	0.224	0.335	0.8736

The trials involving varying the feature vector size are presented in Section. 4.3.3. The performance of the 'pedestrian' showed significant improvement when the region proposals with a feature vector length of 512. Based on this inference, the F-ConvNet incorporating CLSTM-2 and CLSTM-2-KAM modules were re-trained for 'pedestrian' with a feature vector size of 512. The results were collated with the 'car', which was trained with a feature vector size of 1,024. The true positive metric score and nuScenes detection metrics are depicted in Table. 5.10 and Table. 5.11. This further boosted the NDS values by $\sim 3\%$ (**0.8314** \rightarrow **0.8666**) for convolutional-LSTM and $\sim 1.5\%$ (**0.8736** \rightarrow **0.8805**) for convolutional-LSTM with kernel attention mechanism compared the results achieved by using the same feature vector size for both the class presented in Table. 5.9.

A visual illustration of the effectiveness of estimating the oriented 3D bounding box using the proposed spatio-temporal method compared with the ground truth and estimations obtained using the F-ConvNet (baseline) model is shown in Fig. 5.6. In the figure, the blobs highlighted in red represent the instances where the model mispredicts the heading

Table 5.10: True positive metric scores for cars and pedestrians with feature vector size 1,024 and 512 respectively on multi-frames ('m') reference incorporating CLSTM-config-2 and Kernel Attention Mechanism (KAM) on subset of nuScenes data. Best performance in bold

Method	Cars				Pedestrians			
	<i>AP</i>	<i>ATE</i>	<i>ASE</i>	<i>AOE</i>	<i>AP</i>	<i>ATE</i>	<i>ASE</i>	<i>AOE</i>
CLSTM-2-m	0.958	0.129	0.164	0.463	0.983	0.104	0.290	0.274
CLSTM-2-KAM-m	0.958	0.119	0.161	0.563	0.983	0.097	0.243	0.074

Table 5.11: nuScenes metric computed with multi-frames ('m') reference for cars and pedestrians with feature vector size 1,024 and 512 respectively incorporating CLSTM-config-2 and Kernel Attention Mechanism (KAM) on subset nuScenes data. Best performance in bold.

Method	<i>mAP</i>	<i>mATE</i>	<i>mASE</i>	<i>mAOE</i>	<i>NDS</i>
CLSTM-2-m	0.971	0.117	0.227	0.369	0.8666
CLSTM-2-KAM-m	0.971	0.108	0.202	0.319	0.8805

angle. The deep learning model takes advantage of the spatio-temporal information extracted from the multi-frame reference during oriented bounding box estimation, which eventually improves as the model sees more object instances in the input sequence.

5.3 Summary

This chapter addresses the research question of investigating a suitable approach to incorporating spatio-temporal information into the F-ConvNet architecture. Three approaches were proposed to incorporate temporal information. Firstly, a simple frame stacking approach without shuffling the input data sequences. Secondly, a convolutional-LSTM module was added to the F-ConvNet architecture to learn the temporal information of the objects from the region proposals generated using the input-sequenced data. Thirdly, the attention mechanism concept is employed in the frame stacking approach to aid the network in focusing on the points representing the objects within the region proposals. At the same time, it learns to extract temporal information. The architectures were trained and evaluated using single-frame and multiple-frame reference aspects on the subset of the nuScenes training dataset

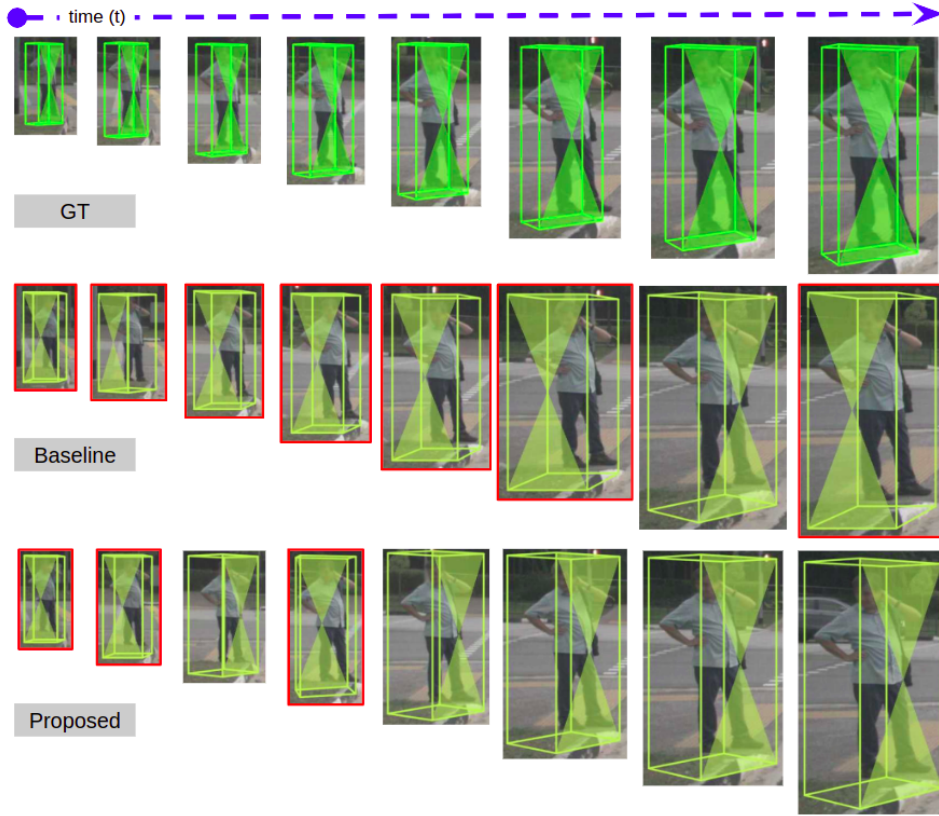


Figure 5.6: Comparison of the spatio-temporal 3D object detection results with the baseline and ground truth. The top row presents the ground truth (GT) annotation, middle row presents the Baseline results and bottom row presents the results obtained using the proposed spatio-temporal architecture. The orientation of the object is presented with a side having a cross mark on the 3D bounding box and all the GT annotated are obtained from the nuScenes dataset.

considered in this research work. The following are the conclusions made from these trials:

1. The frame stacking approach, being the simplest of all the methods proposed in this chapter, failed to show any positive impact on the model's performance.
2. Two configurations of CLSTM were considered to investigate their effectiveness in the final performance of the module. The second configuration, which concatenates the output from the first hidden layer with the last one, improved the nuScenes detection score by $\sim 4\%$ and also reduced the true positive metric error of both 'car' and 'pedestrian'. Hence, the second configuration of the CLSTM module was chosen to evaluate the performance of the F-ConvNet with CLSTM architecture.
3. Since, the experimental set-up involved customised nuScenes dataset, the performance

of each of the proposed architectures is compared with the baseline results only. Obtaining the SOTA models and running the experiments on these models was not possible as it required code changes for training on the customised dataset.

4. Three configurations of the architecture focusing on the placement of the KAM module were explored to study the influence of the attention mechanism to enhance model focus on the input's salient region. The third configuration, where the PointNet features are passed to the KAM, fed to the FCN and then to the regression branch with the KAM module to predict the 3D bounding boxes, improved the NDS value by $\sim 2\%$ with single-frame reference and $\sim 4\%$ with multiple-frame reference compared to the frame stacking approach.
5. The results from this study show that the inclusion of temporal features extracted from the lidar point cloud data significantly reduced the true positive metric, specifically the orientation error of the 3D bounding box from 0.819 to 0.463 and 0.294 to 0.107 for 'cars' and 'pedestrian' classes respectively, on the customized subset of nuScenes training data. The overall nuScenes detection score (NDS) was also improved by $\sim 5\%$ ($0.822 \rightarrow 0.8736$) compared to the reference baseline.
6. The performance on the 'pedestrian' class, for both configurations, showed significant improvement when the region proposals were represented with a feature vector length equal to 512. These newly obtained 'pedestrian' results were collated with the 'car', which was trained with a feature vector size of 1,024. This further boosted the NDS values by $\sim 3\%$ ($0.8314 \rightarrow 0.8666$) for convolutional-LSTM and $\sim 1.5\%$ ($0.8736 \rightarrow 0.8805$) for convolutional-LSTM with kernel attention mechanism compared the results achieved by using the feature vector size of 1,024 for both the classes. The mean true positive metric scores were also improved compared to the baseline or other configurations considered in this chapter.

Incorporating spatio-temporal feature coupled with attention mechanism extracted from the lidar point cloud data into the 3D detection architecture improved 3D detection accuracy (mAP) with reduced true positive error metrics. The results obtained in this study have expanded the understanding of how temporal information and attention mechanisms in a deep

network can improve overall performance, specifically for dynamic objects (e.g., pedestrians). Further utilisation of spatio-temporal features can improve network design for 3D object detection by employing multi-frame, multi-modal data fusion techniques for effective and targeted utilisation of sensor data from instrumented vehicles, which is explored in the next chapter.

Chapter 6

Multimodal Data Fusion and Temporal Encoding

Each of the various sensors like camera, and lidar data employed by the perception system to realise the computer vision task has certain limitations. For example, standard camera-based methods only provide texture information, no depth information. On the other hand, lidar-based processing methods provide depth information but lack texture information. However, the availability of new lidar technologies can provide texture information. Most of the SOTA methods have considered KITTI, nuScenes and recently waymo. These are open datasets where annotations for synchronised sensor data are made available. Although, the ego vehicle used by these data providers consists of newer sensors annotations are not made available for the public for training or testing the 3D detection models. Hence, when these datasets are employed the high-level features extracted from the image data play a vital role in object detection and classification, while the lidar data provides the depth information which is critical for accurately estimating the 3D location and object size. Furthermore, as the distance from the sensor increases, the point cloud density decreases proportionally. Since texture and depth modalities are essential in 3D object detection, most fusion-based methods for object detection available in the literature discussed in Sections 2.2.1 of Chapter 2 employs one of the fusion schemes illustrated in Fig 6.1 to fuse the representation derived from these data.

The exploration of the F-ConvNet architecture facilitated an understanding of handling and processing raw lidar point cloud data and how the prior localisation can further aid in

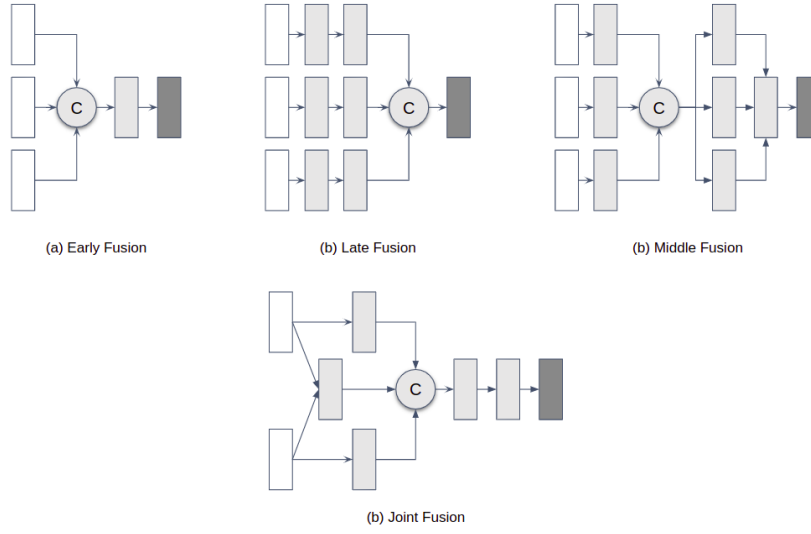


Figure 6.1: Illustration of fusion schemas available in the literature [61, 7, 68]

optimising the processing of sparse and unstructured data. However, F-ConvNet depends on the 2D region proposals, highlighting a fatal risk involved when the failure of 2D detectors will deteriorate all subsequent pipelines. Scope exists to improve and adapt the pipeline to process the image features instead of just using the coordinate position. F-ConvNet follows a sequential data fusion schema that can be categorised as a form of early data fusion. The 2D region information of the objects obtained using the image space is used to generate the frustum features used by the subsequent modules in the pipeline to estimate oriented 3D bounding box parameters.

The utilisation of coordinate information about the region proposal containing the object will help in optimising the data processing of the sparse lidar point cloud information. However, the rich texture and contextual image feature information are entirely ignored, which can be a potential candidate to improve the performance of the detection architecture. Using high-level features from the image space and depth information from the lidar space can aid the current research to learn and understand the significance of including temporal aspects into the deep learning network for perception-related tasks specifically for object detection. High-level features extracted in the image and lidar space, along with the spatio-temporal features, will be able to mitigate the error in the 3D-oriented box estimation. The errors are caused due to the perspective changes in the object and autonomous vehicle as the vehicle drifts w.r.t the object or can have relative motion between the vehicle and the object, which

is illustrated in Section 4.2 of Chapter 4.

This chapter addresses Research Question 3: the methodologies that can aid the fusion of multiple modalities and incorporation of spatio-temporal information into object detection networks. A brief overview of the fusion schema to combine lidar and image data are discussed in Section 6.1. Making use of these fusion schemas, Section 6.2 presents the two frameworks for estimation of 3D oriented bounding boxes: (1) F-ConvNet with multimodal data fusion architecture (Section 6.2.1) and (2) Incorporation of spatio-temporal information into the F-ConvNet using convolutional-long short term memory (CLSTM) and kernel attention mechanism (KAM), discussed in Section 5.1, along with the multimodal data fusion (Section 6.2.2). Section 6.3 provided the details about the dataset, experimental setup, and the metrics employed for training, testing and evaluation of the proposed pipeline, followed by results and discussion in Section 6.4.

6.1 Fusion Strategies

The extraction of features from the image is achieved using the off-the-shelf PSPNet [118] proposed by Zhou, Jingchun, et al., which is pre-trained on the ImageNet [140] dataset. The PSPNet architecture is shown in Fig. 6.2. First, ResNet18 is used as the backbone network to extract the feature map from the input image, followed by a pyramid pooling module to accumulate the contextual information at four pyramid levels with bin sizes of 1x1, 2x2, 3x3, and 6x6 by covering the whole, one-half and small portions of the images. The accumulated contextual features are then fused to form a global contextual feature that is concatenated with the original feature map and fed to a convolutional layer to generate the final feature representatives. This section presents a brief overview of the fusion methods: global multimodal fusion (GMF), intermediate multimodal fusion (IMF), and joint multimodal fusion (JMF) adopted for combining the lidar and image data.

6.1.1 Global Multimodal Fusion (GMF)

Global multimodal Fusion (GMF), as shown in Fig. 6.3 is the simplest of fusion methods employed to capture the correlation across modalities. The high-level representations of the image data generated by the PSPNet is fused into the PointNet architecture by concatenating

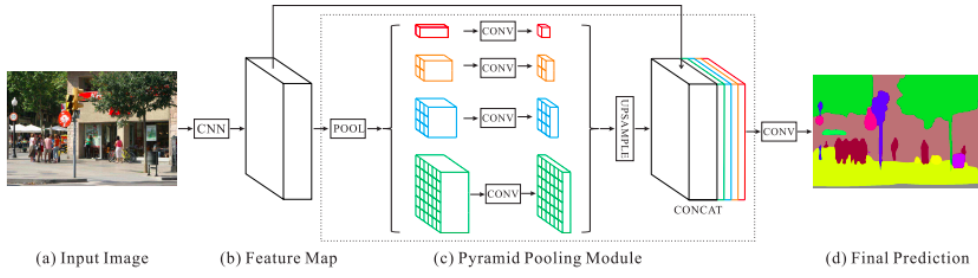


Figure 6.2: Pyramid Scene Parsing Network (PSPNet) Architecture [118]

with the point feature followed by a convolutional layer for applying a non-linear transformation to generate the joint-representation of the feature map before passing the feature vector to the max operator. Although the GMF increased the dimensionality of both the representation, it lacks the ability to capture the complex correlation and semantic information across the modalities [141], as only the error of the shared layer can be propagated to each modality, making it difficult for the network to extract the features which can correlate between the modalities, in later stages of the network [68].

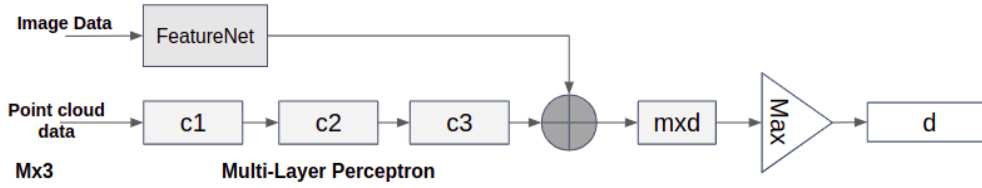


Figure 6.3: Global multimodal fusion schema of Lidar and Image data representation. c1, c2, and c3 represent the multi-layer perceptron layers (mlp) of the PointNet Module

6.1.2 Intermediate Multimodal Fusion (IMF)

Intermediate multimodal fusion (IMF) addresses the issues related to the GMF as the data representation takes advantage of the semantic information from the shared layers extracted from the individual modalities. These semantic features are extracted from the individual modality-specific layers, eventually improving the network's ability to capture the correlations between the layers. The Intermediate multimodal fusion (IMF) schema is shown in Fig. 6.4. In IMF, the high-level image representations generated using PSPNet are scaled accordingly to the layer level of the PointNet architecture before concatenating with the point feature. These features are then used to generate the joint representation by applying

multiple non-linear transformations. Then the feature maps are passed to the max operator to obtain the final fused features of the lidar and image data. Although the IMF can correlate the similarities between the modalities to learn the contextual information existing in the individual modalities representing the object or the scene, it also contains only one path for learning correlations between the modalities as the specific weights can only be optimised with respect to the correlation in the top layer.

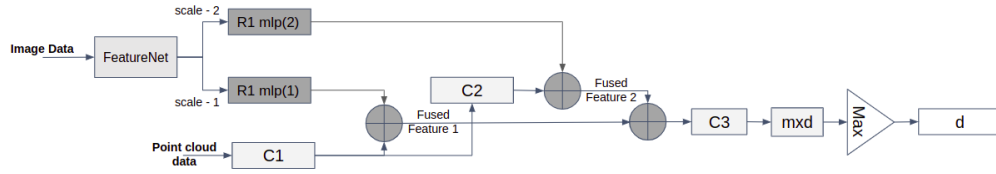


Figure 6.4: Intermediate multimodal fusion of Lidar and Image data representation. c1, c2, and c3 represent the multi-layer perceptron layers (mlp) of the PointNet Module

6.1.3 Joint Multimodal Fusion (JMF)

Joint multimodal fusion (JMF) [68], as shown in Fig. 6.5 provided multiple representations for each of the modalities at different levels of the network after applying non-linear transformations to each of these modalities containing spatial and semantic information. The JMF shares features at multiple layers and passes the partially fused information for further processing to estimate the inter-modality and intra-modality representations. The shared layers in the JMF are densely stacked, making it feasible to correlate between different modalities in each of the representation level and modality specific network layers. Each shared layer will be able to capture the correlation at the current level but also learn the dependency among the correlations. Thus, the JMF is the most suitable fusion strategy in the current context of the investigation to capture the structural variation in lidar representation and correlate it with the rich textural representation of the image. This aids the module in generating the composite hierarchical correlation between the modalities in contrast to the global or intermediate multimodal fusion method.

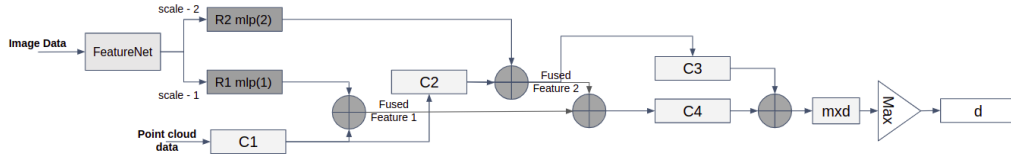


Figure 6.5: Joint multimodal fusion of Lidar and Image data representation. c1, c2, c3 and c4 represent the multi-layer perceptron layers (mlp) of the PointNet Module

6.2 Pipeline

This section presents the two frameworks employing fusion methods discussed in the previous section and incorporating spatio-temporal information into the F-ConvNet architecture along with the kernel attention mechanism module described in Section 5.1.

6.2.1 Multimodal 3D Object detection

The F-ConvNet architecture is discussed in Section 3.2.3. Fig. 6.6 presents the multimodal 3D object detection architecture utilising the fusion methods discussed in Section 6.1 for fusing the lidar and image data. The original F-ConvNet utilises the early fusion strategy where only the 2D coordinate position of the region proposals is used to generate the frustums. In this section, the initial region proposals, along with the image features extracted using the PSPNet, are fused with the layers of the PointNet architecture using fusion methods: GMF, IMF, and JMF for generating the joint representation of the modalities at different frustum resolution. These multi-resolution joint representations are passed to the FCN block, followed by a regression branch for the final estimation of the box parameters.

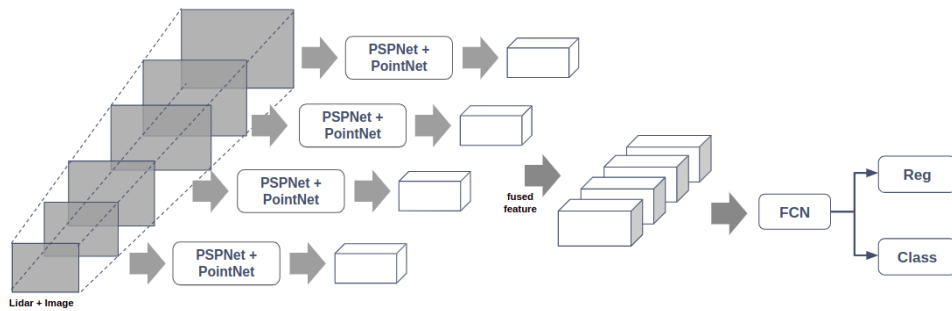


Figure 6.6: Multimodal data fusion architecture

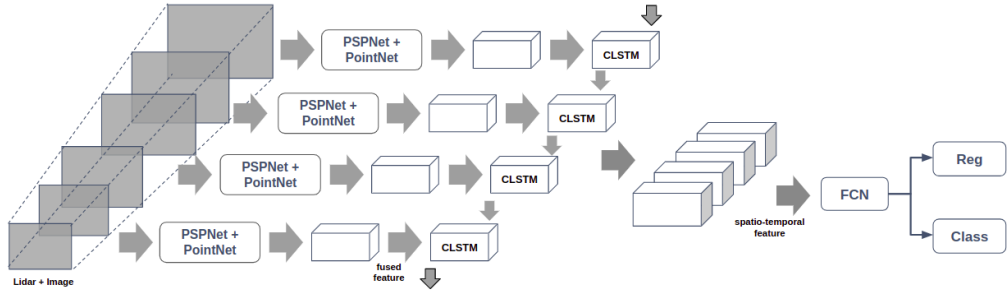


Figure 6.7: Multi-modal data fusion architecture incorporating spatio-temporal using CLSTM.

6.2.2 Multimodal Spatio-Temporal 3D object detection

The network architecture shown in Fig. 6.7 fuses the multiple modalities employing the strategies discussed in Section 6.1. The architecture extracts the temporal features using convolutional-LSTMs, as discussed in Section 5.1.2, using the joint representation obtained from the fusion model. A convolutional-LSTM module is placed between the multimodal data fusion block consisting of PointNet and PSPNet, and a fully convolutional network (FCN) block to learn spatial and temporal feature from the fused modalities of the input sequence. The CLSTM module consists of one input layer and five hidden layers with dimensions size same as the output feature map size of the four frustum resolutions of the PointNet architecture having values 128, 128, 256, and 512. The downwards-facing arrow in Fig. 6.7 indicates the previous hidden states information of the Conv-LSTM while extracting the current state information. Based on the learnings from Chapter 5, the second configuration of the CLSTM module and FCN layers presented in Table 5.1 were employed.

The inference drawn from the trials involving KAM module discussed in Section 5.1, assisted in designing the architecture presented in Fig. 6.8. The architecture includes CLSTM and the kernel attention mechanism (KAM) module to extract the spatio-temporal features from the joint representations from the multiple modalities. The spatio-temporal feature aid the network by propagating temporal information to learn the structural and texture features of the objects to minimize the error during the final parameter estimation. The downwards-facing arrow in Fig. 6.8 indicates the previous hidden states information of the Conv-LSTM while extracting the current state information. Both the architectures are trained with the same sequences of data as employed in Section 5.1.

The true positive metric scores of the CLSTM configurations for individual classes are depicted in Table. 6.5 and overall performance of the models are depicted in Table. 6.6.

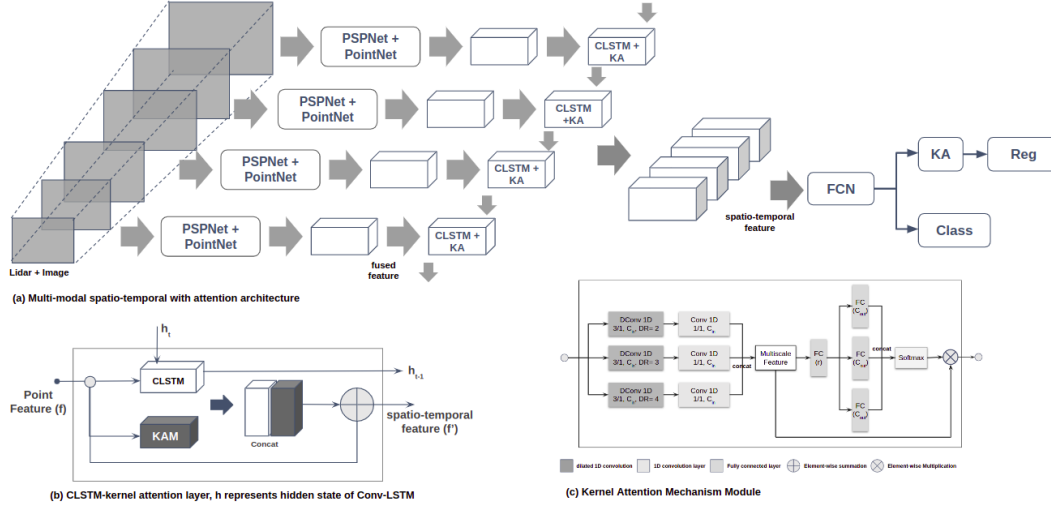


Figure 6.8: Multi-modal data fusion architecture incorporating spatio-temporal using CLSTM and kernel attention module.

6.3 Experimental Setup

The KITTI and nuScenes datasets were employed to train and evaluate the performance of the proposed multimodal 3D detection architecture. The KITTI 3D object detection dataset [139, 138] and its distribution of samples for training and validation are described in Section 2.3.1 and Section 3.2.3. The nuScenes[32] dataset is described in Section 2.3.4 and the customization of the data for training and testing is discussed in Section 5.2.1.

The models were trained on a single Nvidia-TitanX GPU with 12GB RAM for 50 epochs using the adam optimizer[106] from scratch and an off-the-shelf PSPNet [118] (PSPNet-18) model pre-trained on ImageNet [140] is used to extract the image features. The learning rate and weight decay values were set to 0.001 and 0.0001, respectively, with a batch size of 16 for single-frame reference mode, and during multi-frame reference mode, three consecutive frames and a batch size of four without shuffling the input sequence is considered. The region proposals are obtained from the ground-truth annotation and are augmented with a translation and scaling factor during training and testing. The number of sample points for

each region proposal is set to 1,024 and 512 for ‘car’ and ‘pedestrian’, respectively, whose values are selected randomly depending on the trial configurations.

Same as [85], while training on the nuScenes dataset, 4 frustum resolutions of [0.2, 0.5, 1.0, 2.0] and stride values of [0.1, 0.25, 0.5, 1.0] for the ‘car’ category and 4 frustum resolutions of [0.2, 0.4, 0.8, 1.6] and stride values of [0.1, 0.2, 0.4, 0.8] for the ‘pedestrian’. For the KITTI dataset, 4 frustum resolutions of [0.2, 0.5, 1.0, 2.0] and stride values of [0.1, 0.25, 0.5, 1.0] for the ‘car’ category and 4 frustum resolutions of [0.2, 0.4, 0.8, 1.6] and stride value are set to [0.1, 0.2, 0.4, 0.8] for ‘people’ categories (‘pedestrian’ and ‘cyclist’).

The performance of the models is evaluated using the KITTI benchmark metric, mean Average Precision (mAP), as described in Section 2.4.1 for easy, moderate and hard cases of the classes appearing in the dataset. The nuScenes dataset is evaluated using the metric [32], which includes: the Average Precision (AP) metric and True Positive (TP) metric for the ‘car’ and ‘pedestrian’ class on the custom test data. Section 5.2.3 details the AP and TP metric. Single-frame reference mode was considered for KITTI due to the absence of sequenced data and multiple-frame reference modes for nuScenes to understand the model performance under various configurations.

6.4 Results and Discussion

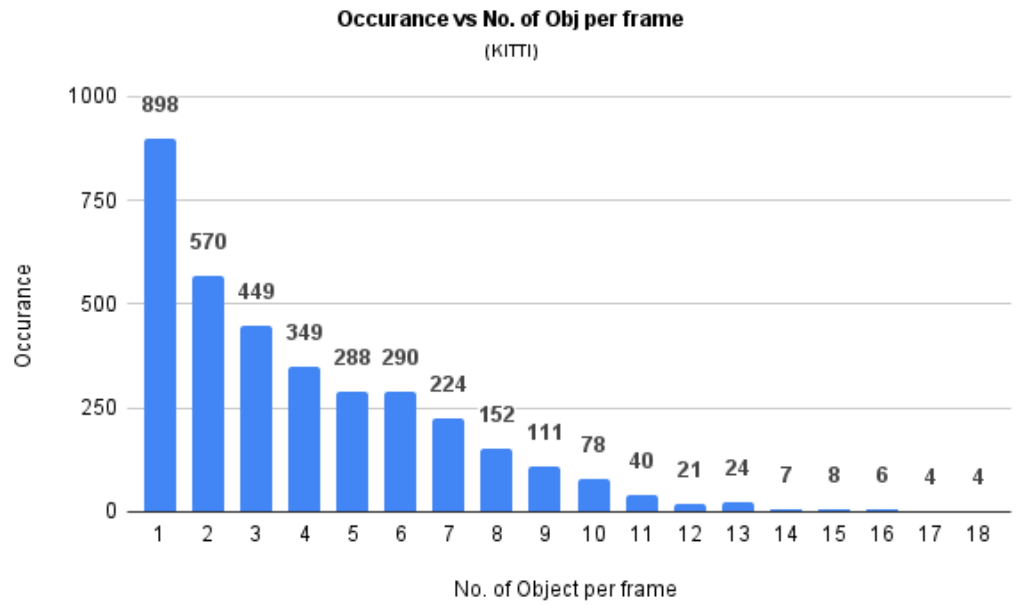
The 3D detection and BEV detection results of the F-ConvNet model employing lidar and image data on the KITTI dataset are depicted in Table 6.1 and Table 6.2 respectively. The 3D detection results with the joint multimodal fusion (JMF) strategy achieve an improvement of $\sim 2\text{-}3\%$ over the baseline for most cases except for the moderate case of the ‘pedestrian’ class, where improvement is around $\sim 7\%$. In the case of BEV detection, improvements of $\sim 1\%$ for ‘car’, using intermediate multimodal fusion (IMF), and $\sim 1\%$ for ‘pedestrian’ and ‘cyclist’, using JMF, are observed except for the easy cases of the ‘pedestrian’ class, which achieved an improvement of $\sim 2\%$ using the global multimodal fusion (GMF) strategy. Overall, the performance of the model is improved by fusing the image and lidar data to estimate the 3D bounding box.

Table 6.1: Comparison of 3D Detection AP (%) results of the proposed architecture with the state-of-the-art methods, where global (gmf), intermediate (imf) and joint multimodal fusion (jmf) schema employed in the F-ConvNet (Arch) to fuse the Lidar (L) and Image (I) data. Best performance in bold.

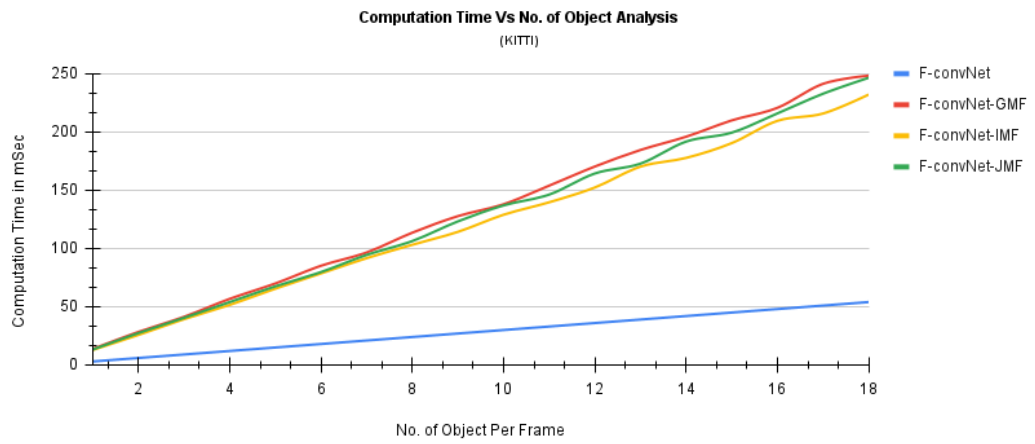
Method	Type	Car			Pedestrian			Cyclist		
		Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
VoxelNet [107]	L	81.97	65.46	62.85	-	-	-	-	-	-
SECOND [102]	L	87.43	76.48	69.10	-	-	-	-	-	-
F-ConvNet [85]	LI	86.08	86.04	86.45	81.32	75.16	75.64	94.02	88.33	86.38
MV3D [65]	LI	71.29	62.68	56.56	-	-	-	-	-	-
PointFusion [101]	LI	77.92	63.00	53.27	-	-	-	-	-	-
F-PointNet [97]	LI	83.76	70.92	63.65	-	-	-	-	-	-
Cont. Fuse [94]	LI	86.32	73.25	67.81	-	-	-	-	-	-
AVOD-FPN [65]	LI	84.41	74.44	68.65	-	-	-	-	-	-
IPOD [104]	LI	84.10	76.40	75.30	-	-	-	-	-	-
MMF [71]	LI	87.90	77.86	75.57	-	-	-	-	-	-
AA3D [54]	LI	86.77	76.84	75.92	-	-	-	-	-	-
Arch-gmf	LI	84.55	84.89	77.82	83.95	76.33	76.22	89.86	87.38	79.38
Arch-imf	LI	87.45	87.15	87.19	82.70	76.01	76.16	92.84	84.51	86.49
Arch-jmf	LI	87.89	88.36	88.19	84.10	83.23	78.75	96.88	88.91	87.00

Table 6.2: Comparison of BEV Detection AP (%) results of the proposed architecture with the state-of-the-art methods on KITTI validation set, where global (gmf), intermediate (imf) and joint multimodal fusion (jmf) schema employed in the F-ConvNet (Arch) to fuse the Lidar (L) and Image (I) data. Best performance in bold.

Method	Type	Car			Pedestrian			Cyclist		
		Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
VoxelNet [107]	L	89.60	84.81	78.57	-	-	-	-	-	-
SECOND [102]	L	89.96	87.07	79.66	-	-	-	-	-	-
F-ConvNet [85]	LI	97.77	89.26	89.38	84.89	85.47	78.00	97.19	97.86	89.15
MV3D [65]	LI	86.55	78.10	76.67	-	-	-	-	-	-
PointFusion [101]	LI	87.45	76.13	65.32	-	-	-	-	-	-
F-PointNet [97]	LI	88.16	84.02	76.44	-	-	-	-	-	-
Cont. Fuse [94]	LI	95.44	87.34	82.43	-	-	-	-	-	-
AVOD-FPN [65]	LI	-	-	-	-	-	-	-	-	-
IPOD [104]	LI	88.30	86.40	84.60	-	-	-	-	-	-
MMF [71]	LI	96.66	88.25	79.60	-	-	-	-	-	-
AA3D [54]	LI	89.95	87.70	86.95	-	-	-	-	-	-
Arch-gmf	LI	97.30	89.24	89.26	86.76	78.55	78.44	97.20	88.02	79.83
Arch-imf	LI	98.45	89.98	89.95	85.94	86.21	78.40	93.46	85.51	85.43
Arch-jmf	LI	97.43	89.44	89.49	85.35	86.76	79.18	97.77	98.83	89.46



(a) No. of objects per frame analysis



(b) Per frame computation analysis

Figure 6.9: Computation time analysis of models employed on KITTI dataset

The computational complexity of the F-ConvNet architecture with various fusion strategies on the KITTI dataset is presented in Fig. 6.9. The fusion methods have a higher per-frame computation time than the F-ConvNet, increasing almost five-fold. However, it satisfies the objective of this research to investigate the fusion strategies for fusing lidar and image data. Similarly, the F-ConvNet model was also trained on the customised nuScenes dataset using both image and lidar data. The true positive metric of the ‘car’ and ‘pedestrian’ classes evaluated on the subset of the nuScenes test data are depicted in Table. 6.3. Table. 6.4 presents the overall nuScenes detection metric score of the F-ConvNet model with different fusion strategies employed in this study. The results show that the mAP values are slightly lower than the baseline. However, the mATE, mAOE, mASE and NDS values, which measure more nuanced performance, show improvements when multiple modalities are used for estimating the 3D box parameters. The JMF method achieved maximum improvement of $\sim 14\%$, $\sim 22\%$, $\sim 3\%$ in the mATE, mAOE and NDS values. The GMF significantly reduced the mASE value from 0.2554 to 0.1376. Similarly, IMF and JMF fusion strategies also yielded reduced mASE values of 0.2414 and 0.2350 respectively. Overall, the performance of the model in terms of nuScenes detection metrics was improved by $\sim 2\text{-}3\%$ when both image and lidar data are fused to estimate the 3D bounding box parameters.

Table 6.3: True positive metric scores for cars and pedestrians on nuScenes custom test set, with global (gmf), intermediate (imf) and joint multimodal fusion (jmf) schema employed in the F-ConvNet (Arch) to fuse the Lidar (L) and Image (I) data. Best performance in bold.

Method	Type	Cars				Pedestrians			
		AP	ATE	ASE	AOE	AP	ATE	ASE	AOE
F-ConvNet	L	0.961	0.203	0.184	0.819	0.981	0.118	0.327	0.264
Arch-gmf	LI	0.956	0.123	0.109	0.869	0.981	0.162	0.166	0.290
Arch-imf	LI	0.958	0.156	0.213	0.633	0.978	0.131	0.270	0.267
Arch-jmf	LI	0.958	0.139	0.225	0.586	0.981	0.136	0.245	0.258

The F-ConvNet model incorporating the spatio-temporal features and the multimodal data was also trained on the customised nuScenes dataset using both image and lidar data. The true positive metric of the ‘car’ and ‘pedestrian’ classes evaluated on the subset of the nuScenes test data are depicted in Table. 6.5. Table. 6.6 presents the overall nuScenes detection metric score of the F-ConvNet model incorporating spatio-temporal features with

Table 6.4: nuScenes metric computed for cars and pedestrians on nuScenes custom test set, with global (gmf), intermediate (imf) and joint multimodal fusion (jmf) schema employed in the F-ConvNet (Arch) to fuse the Lidar (L) and Image (I) data. Best performance in bold.

Method	Type	<i>mAP</i>	<i>mATE</i>	<i>mASE</i>	<i>mAOE</i>	<i>NDS</i>
F-ConvNet	L	0.9708	0.1606	0.2554	0.5415	0.8258
Arch-gmf	LI	0.9681	0.1426	0.1376	0.5794	0.8408
Arch-imf	LI	0.9681	0.1435	0.2414	0.4500	0.8449
Arch-jmf	LI	0.9695	0.1375	0.2350	0.4220	0.8523

the different fusion strategies employed in this study. The results show that the *mAP* values are slightly lower than the baseline. The *mATE*, *mAOE*, *mASE* and *NDS* values, which measure more nuanced performance, showed improvements when spatio-temporal features learnt from the multiple modalities was used for estimating the 3D box parameters. The JMF method achieved the highest improvement from 0.227 to 0.167 for *mATE*, 0.369 to 0.215 for *mAOE*, and *NDS* improved $\sim 2\%$. However, the *mATE* value obtained using the baseline model was better than all the fusion strategies. The error score and *NDS* showed further improvement with the inclusion of the attention mechanism. Overall, the performance of the model according to the nuScenes detection metrics was improved by $\sim 3\%$ for F-ConvNet with CLSTM and $\sim 4\%$ for F-ConvNet with CLSTM and KAM module with both image and lidar data fused to estimate the 3D bounding box parameters.

Table 6.5: True positive metric scores for cars and pedestrians on nuScenes custom test set, with global (gmf), intermediate (imf) and joint multimodal fusion (jmf) schema employed in the modified F-ConvNet (Arch) incorporating spatio-temporal encoding through CLSTM and KAM to fuse the Lidar (L) and Image (I) data. Best performance in bold.

Method	Type	Cars				Pedestrians			
		<i>AP</i>	<i>ATE</i>	<i>ASE</i>	<i>AOE</i>	<i>AP</i>	<i>ATE</i>	<i>ASE</i>	<i>AOE</i>
Arch-clstm-2	L	0.958	0.129	0.164	0.463	0.983	0.104	0.290	0.274
Arch-clstm-2-ka	L	0.958	0.119	0.161	0.563	0.983	0.097	0.243	0.074
Arch-clstm-2-gmf	LI	0.942	0.175	0.187	0.452	0.964	0.217	0.196	0.136
Arch-clstm-2-ka-gmf	LI	0.942	0.165	0.177	0.422	0.964	0.185	0.193	0.113
Arch-clstm-2-imf	LI	0.944	0.187	0.260	0.409	0.958	0.109	0.186	0.230
Arch-clstm-2-ka-imf	LI	0.947	0.090	0.174	0.496	0.958	0.139	0.170	0.151
Arch-clstm-2-jmf	LI	0.951	0.149	0.191	0.289	0.968	0.180	0.144	0.141
Arch-clstm-2-ka-jmf	LI	0.951	0.116	0.140	0.159	0.968	0.105	0.098	0.040

The resource utilization and computation time of the various models used in this study are depicted in Table 6.7. The resource utilization of the network is measured based on the modalities (lidar, image) and the number of parameters (in millions) used to train the

Table 6.6: nuScenes metric computed for cars and pedestrians on nuScenes custom test set, with global (gmf), intermediate (imf) and joint multimodal fusion (jmf) schema employed in the modified F-ConvNet (Arch) incorporating spatio-temporal encoding through CLSTM and KAM to fuse the Lidar (L) and Image (I) data. Best performance in bold.

Method	Type	<i>mAP</i>	<i>mATE</i>	<i>mASE</i>	<i>mAOE</i>	<i>NDS</i>
Arch-clstm-2	L	0.971	0.117	0.227	0.369	0.8666
Arch-clstm-2-ka	L	0.971	0.108	0.202	0.319	0.8805
Arch-clstm-2-gmf	LI	0.953	0.196	0.191	0.294	0.8629
Arch-clstm-2-ka-gmf	LI	0.953	0.175	0.185	0.267	0.8719
Arch-clstm-2-imf	LI	0.951	0.147	0.223	0.319	0.8607
Arch-clstm-2-ka-imf	LI	0.952	0.114	0.171	0.323	0.8747
Arch-clstm-2-jmf	LI	0.960	0.164	0.167	0.215	0.8886
Arch-clstm-2-ka-jmf	LI	0.960	0.110	0.119	0.100	0.9159

model. The computation time presented in the Fig. 6.10 represents the average time the model takes to process a frame during inference. The table shows that computation time (training and testing) is directly proportional to the number of trainable parameters in the network. The fusion methods have almost a five-fold higher per-frame computation time than the F-ConvNet.

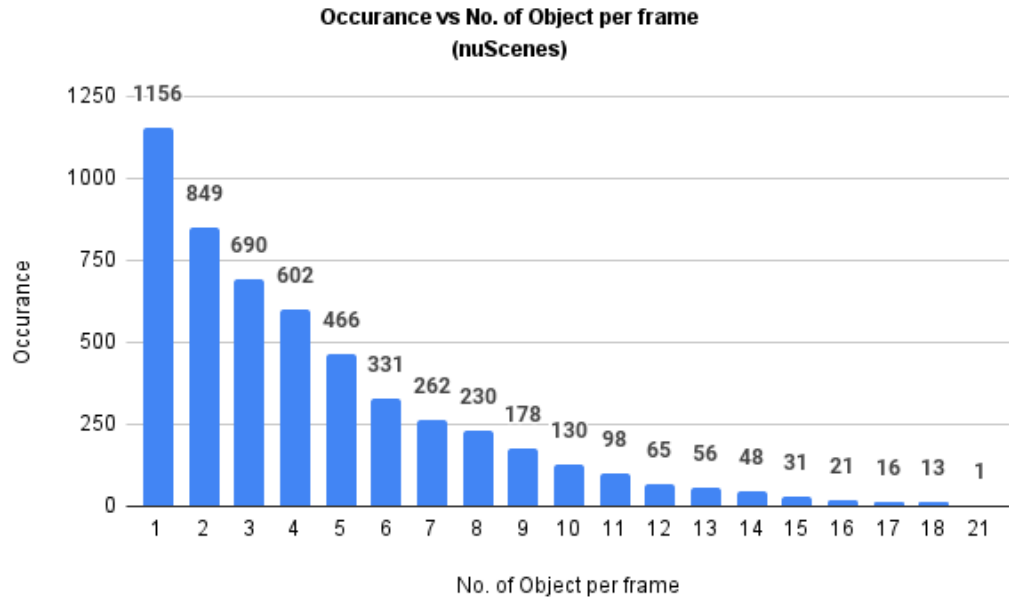
Table 6.7: Overview of Number of parameters and Average per frame computation time of the various models employed in this thesis work, with global (gmf), intermediate (imf) and joint multimodal fusion (jmf) schema employed in the modified F-ConvNet (Arch) incorporating spatio-temporal encoding through CLSTM and KAM to fuse the Lidar (L) and Image (I) data.

Model	Modality	Parameters (M)	Time (msec)	
			kitti	nuScenes
F-ConvNet	L	03.32	13.5	10.5
Arch-clstm-1	L	12.07	46.8	36.4
Arch-clstm-2	L	12.46	50.9	39.6
Arch-clstm-2-ka	L	14.86	57.6	44.8
Arch-gmf	LI	23.13	61.2	47.6
Arch-clstm-2-gmf	LI	32.36	66.6	51.8
Arch-clstm-2-ka-gmf	LI	35.36	70.7	55.0
Arch-imf	LI	22.89	56.3	43.8
Arch-clstm-2-imf	LI	32.12	65.3	50.8
Arch-clstm-2-ka-imf	LI	35.12	69.3	53.9
Arch-jmf	LI	23.07	59.0	45.9
Arch-clstm-2-jmf	LI	33.04	65.3	50.8
Arch-clstm-2-ka-jmf	LI	35.30	69.8	54.3

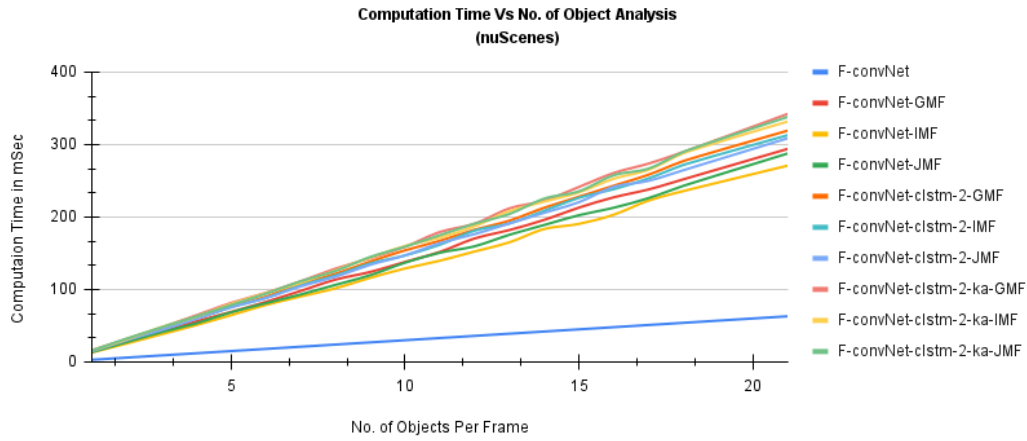
6.5 Summary

This chapter addresses the research question investigating suitable fusion methodologies to incorporate multiple modalities and spatio-temporal features into object detection networks. The three fusion schemas – global multimodal fusion, intermediate multimodal fusion and joint multimodal data fusion – are evaluated to combine lidar and image data. Two frameworks to fuse the multimodal data by employing these three fusion schemes are proposed for estimating the 3D-oriented bounding boxes. The first framework includes F-ConvNet with multimodal data fusion architecture. The second includes incorporating spatio-temporal features into the F-ConvNet using convolutional long short-term memory (CLSTM) and kernel attention mechanism (KAM) extracted from the multimodal data. The proposed architectures were trained and evaluated using multiple-frame references on the subset of the nuScenes training and testing dataset considered in this research work. The following conclusions are presented from these evaluations:

1. For the F-convNet trained on the KITTI dataset with lidar and image data, the 3D detection results with the joint multimodal fusion (JMF) strategy achieve an improvement of $\sim 2\text{-}3\%$ over the baseline for most cases except for the moderate cases of the ‘pedestrian’ class, where improvement is around $\sim 7\%$. In the case of BEV detection, results are an improvement of $\sim 1\%$ for ‘car’ using intermediate multimodal fusion (IMF) and $\sim 1\%$ for ‘pedestrian’ and ‘cyclist’ is observed with JMF strategy except for the easy cases of ‘pedestrian’ class which achieved an improvement of $\sim 2\%$ using global multimodal fusion (GMF) strategy. **Overall, the performance of the model is improved by fusing the image and lidar data to estimate the 3D bounding box.**
2. For the F-ConvNet model incorporating the spatio-temporal features and the multimodal data trained on the customised nuScenes dataset using both image and lidar data, the results show that the mAP values are slightly lower than the baseline. In contrast, the mATE, mAOE, mASE and NDS values, more nuanced metrics, show improvements when spatio-temporal information learnt from the multiple modalities was used for estimating the 3D box parameters. The JMF method improved the NDS by $\sim 2\%$. However, the mATE value obtained using the baseline model was better



(a) No. of objects per frame



(b) Per frame computation analysis

Figure 6.10: Computation time analysis of models employed on nuScenes dataset

than the fusion strategies. The true positive error score and NDS showed further improvement with the inclusion of the attention mechanism. **Overall, the performance of the model in terms of nuScenes detection metrics was improved by $\sim 3\%$ for F-ConvNet with CLSTM and $\sim 4\%$ for F-ConvNet with CLSTM and KAM module with both image and lidar data are fused to estimate the 3D bounding box parameters.**

3. The resource utilization and computation time analysis of the various models used in this study showed that the average per-frame computation time is directly proportional to the number of trainable parameters in the network. The fusion methods increased per-frame computation time by five-fold compared with F-ConvNet. **However, the inclusion of this complexity also improved the model performance by almost $\sim 8\%$ with reduced true-positive metric scores.**

The Fusion of multiple modalities exploits the individual characteristics of the data assisting the deep networks in learning a better representation of the data resulting in reduced errors and improved accuracy. Further, the understanding of the structural information of the object using a lidar point cloud is enhanced by learning the temporal aspect of the sequenced data. However, using lidar and image data in the real-world scenario is challenging as the sensor operates at different frequencies leading to synchronization problems. The advancement in the underlying hardware technology has improved the optimization of deep learning framework to handle and processing of multiple modalities operating at different frequencies effectively. For instance, in an object detection framework consider a use-case scenario where the camera sensor operates at 40 Hz and the lidar at 20 Hz. The system acquires one lidar point cloud data for every second image frame. If the proposed models presented in this chapter need to be employed for the 3D detection task which takes a maximum computation time of around 60 msec (~ 15 FPS or 16 Hz) for processing single lidar and image data. Instead of dropping the unsynchronized frames, apply a prediction model on these frames to estimate the 3D parameters based on the previous keyframes (synchronized frames). The unsynchronized frames can be used to learn the temporal feature of the object when the other sensor modalities are unavailable. This strategy could make the framework robust and improve the estimation process during sensor synchronization problems.

Chapter 7

Summary and Conclusions

The work presented in this thesis investigates the significance of temporal and spatial features extracted from lidar and image data to improve the detection accuracy of a deep learning-based object detection architecture. The advancement in the sensor and hardware technologies used in the development of autonomous vehicles has increased the competition in the automotive industry. To enable fully autonomous driving and use by humans, these vehicles should be highly reliable and trusted for having a safe driving system on the road with 'zero' casualties. This requirement has raised the expectation of the perception system to have human-like perseverance in the driving environment in determining the precise positions of the objects in the scene. The continuous data acquisition in the AV exhibits temporal continuity, and objects appearing in successive frames will have relative motion due to their own motion or camera motion or relative motion between camera and objects. This allows the utilization of detection results and temporal features of the objects from earlier frames to refine predictions and/or parameter estimations in the current frame. There has been limited research about extracting temporal feature using lidar and image data, and its potential usage during the estimation of object parameters for improved localisation and detection. Further, the availability of large-scale sequenced annotated lidar and image data has paved the way to enhance the knowledge to understand the dynamic nature of objects in an autonomous vehicle driving environment and has motivated the proposal of the pipeline incorporating spatial features from lidar and image.

The proposed 3D object detection framework employs lidar and image data and explores the representations for various object classes considered in KITTI and nuScenes datasets.

During the literature review, State-of-the-art results have used fusion-based techniques, combining multiple input modalities and training the model to utilise individual characteristics of the data for improved detection accuracy. However, as the data acquisition and annotation led to the release of large-scale datasets like nuScenes and waymo datasets. These datasets contain certain unique instances where the current state-of-the-art fails even after achieving good performance on the benchmark evaluation dataset. In addition, there was limited research employing multiple-frame references paving the way for the investigation of the significance of temporal features and fusion of lidar and image data.

Chapter 4 explores some widely used representations of the lidar point cloud, such as bird-eye-view, range-view, point-pillars, and voxel format on KITTI and nuScenes datasets. Changing the data representation assisted in understanding the complexity involved in handling the data and provided the initial baseline performance on the chosen F-ConvNet architecture. In addition, this chapter presents various investigations carried out to select the training parameters while designing the framework for incorporating temporal information into the 3D detection network. Finally, the experimental results were instrumental for designing pipelines to facilitate the extraction of spatio-temporal information from the raw point cloud data or fused multimodal data.

Chapter 5 addresses the research question of identifying a suitable approach to incorporating spatio-temporal information into the F-ConvNet architecture. Three approaches were proposed to incorporate temporal information. Firstly, a simple frame stacking approach without shuffling the input data sequences. Secondly, a convolutional-LSTM module was added to the F-ConvNet architecture to learn the temporal information of the objects from the region proposals generated using the input-sequenced data. Lastly, the attention mechanism concept is employed in the frame stacking approach to aid the network in focusing on the points representing the objects within the region proposals. At the same time, it learns to extract temporal information. The architectures were trained and evaluated using single-frame and multiple-frame reference aspects on the subset of the nuScenes training dataset.

Chapter 6 addresses the research question of identifying suitable fusion methodologies to fuse multiple modalities and incorporate spatio-temporal information into object detection

networks. The three fusion schemas, namely global multimodal fusion, intermediate multimodal fusion and joint multimodal data fusion, are explored to combine lidar and image data. Two frameworks to fuse the multimodal data by employing these three fusion schemes are proposed for estimating the 3D-oriented bounding boxes. The first framework includes F-ConvNet with multimodal data fusion architecture. The second included incorporating spatio-temporal information into the F-ConvNet using convolutional-LSTM (CLSTM) and kernel attention mechanism (KAM) extracted from the multimodal data. The proposed architectures were trained and evaluated using multiple-frame references on a subset of the nuScenes training and testing dataset.

In the next Section 7.1, the research questions formulated in the opening chapter are revisited and addressed from a post-experimental perspective. The final section discusses the feasibility of applying the proposed framework in the context of real-world applications in autonomous vehicles equipped with improved sensor and hardware technologies. Finally, concludes by providing future directions to current research on 3D detection using multiple modalities and how to handle uncertainties that might exist or arise within the data sequence.

7.1 Research Questions and Proposed Solutions

This thesis evaluates 3D detection architecture incorporating convolutional-LSTMs and an attention mechanism for estimating the 3D bounding boxes using spatio-temporal features and the fusion of multiple modalities. The hypothesis and research questions are reviewed here and analysed according to the experimental results.

Hypothesis: *“Multimodal spatio-temporal information can contribute to bridging the gap between 2D image space and 3D metric space by improving the accuracy and reducing the error score of deep learning framework for 3D object detection.”*

To investigate the hypothesis of this research, the following questions were considered:

- **Research Question 1 (RQ1): How best to represent/prepare multimodal data for training the object detection architecture?**

Data representation is crucial while training a deep learning network to extract the high-level features for better representation of raw data into meaningful information,

fed to the prediction layer for estimating the parameters depending on the task requirement. The preliminary findings of the pilot study conducted in Chapter 3 led to the framework selection with the following conclusion.

- Complex-YOLO3D with BEV representation of lidar point cloud data was employed to understand the 3D detector for estimating the oriented bounding box positions in the world coordinates. The trial with the Complex-YOLO3D laid the necessary foundation to incorporate a convolutional-LSTM for extracting the spatio-temporal information and propagating the extracted temporal information into the network layer, which the final estimation layers will use. This initial investigation on temporal encoding aided the current research in acquiring further insight into the data representation aspect and analysing the requirements in the dataset for effectively utilising the temporal aspect in the data are discussed in Chapter 4.
- Understanding the structural information of an object in the scene is effective with the raw lidar point cloud data. However, the complexity of utilising the BEV representation with a loss of information led to exploring the Frustum-ConvNet framework. With prior localisation of the objects in the scene using the 2D detector, transformed to the 3D space for extracting the lidar point cloud data within the proposal to generate the Frustum level features was promising and appeared to be an ideal set-up for exploring the inclusion of temporal aspects. F-ConvNet was chosen as the baseline architecture to investigate research questions RQ2 and RQ3.
- The prior localisation of the objects using an off-the-shelf pre-trained 2D detector using the KITTI dataset assisted the research in studying the influence of priors on model's performance in terms of mAP (%). The network achieved maximum mAP with the 2D detector confidence score set between 0.1 to 0.001 and the IoU threshold value equal to 0.95. However, this also increased the average computation time to process the frame as the number of region proposals passed to the network increased. Furthermore, the F-convNet framework relies on these prior region proposals for estimating the objects' final 3D bounding

boxes, creating a bottleneck in employing a 2D detector to detect all the objects appearing in the scene to test the final accuracy of the 3D detection framework. To overcome this aspect for further investigations, the ground truth annotation with augmentations as preliminary region proposals covers all the objects appearing in the current and consecutive sequences.

- Visual analyses of the raw lidar point cloud data in the sequenced frames of the nuScenes dataset assisted in understanding the role of temporal information while training the network to learn the structural information of the object from the point cloud distribution. Furthermore, the illustration helped to understand the benefit of keeping track of variations in the point cloud distribution as the AV acquires the data for estimating the target position in the scene. Tracking the variation will assist the deep learning models to handle the reduced misalignment issues in the data while estimating box parameters.
 - The multiple-sweep lidar point cloud data in the nuScenes dataset appeared very interesting during the visual analysis of the data. However, investigating the use of single sweep over multiple-sweep lidar point cloud data for training networks showed that the true positive error metric values increased, reducing the model's overall performance.
 - The feature vector size used to represent the object class was one of the parameters affecting the model performance. To understand this aspect, the feature vector length of lidar data is varied for the object under consideration in KITTI and nuScenes datasets. As a result, the 'Car' achieved the maximum with a feature vector length of 1024, and for 'pedestrian' and 'cyclist', the feature vector length of 512 produced the highest performance.
- **Question 2 (RQ2): How can temporal information be extracted from multimodal data?**

Temporal information, along with deep features, can aid the deep learning network to understand data variation in the object/s or scenes by learning the changing appearance when the vehicle moves in a dynamic environment. The preliminary finding

in Chapter 3, related to the temporal understanding, assisted in addressing RQ2 by dividing it further into two aspects:

- *Research Question 2.1 (RQ2.1): What is the minimum amount of sequential data needed to extract and exploit temporal information to positively impact the training and final accuracy?*

The purpose of RQ2.1 is to understand the minimum number of frames containing the object of interest that the network needs to see to extract the temporal information and gain knowledge regarding the structural variations in the object and improve the overall network performance.

In Chapter 4, the RNN-based CLSTM and CGRU were incorporated into the detection network to extract the temporal information. The CLSTM outperformed the CGRU during the evaluation of the KITTI dataset. Therefore, CLSTM was employed to analyse the minimum sequence of data and optimum object density to effectively extract temporal information. The analysis showed that the model with three consecutive frames with a maximum of eight recurring objects performs better than other configurations considered in this study. Therefore, the configuration with three consecutive frames with a maximum of eight recurring objects was selected for the investigation detailed in Chapter 5 and Chapter 6.

- *Research Question 2.2 (RQ2.2): Which is the most suitable approach for including temporal information? Potential candidates include Recurrent Neural Network (RNN) based Long Short Term Memory (LSTM), Gated Recurrent Unit (GRU), Temporal convolution network (TCN) or frame stacking for extracting temporal information from the sequence data.*

The purpose of RQ2.2 is to explore the approaches for extracting temporal information from the input data sequence. The most straightforward approach is ‘frame stacking’, where consecutive frames are stacked and fed to the network without any shuffle. In this approach, the deep detection network is presumed to learn the temporal aspect as it sees through the sequenced data during training. Furthermore, temporal information is extracted using the RNN-based Convolutional-Long Short Term Memory (LSTM) module and adopted into the

proposed framework presented in Chapter 5 and Chapter 6. Finally, the following conclusions are drawn while addressing RQ2.2.

- * The frame stacking approach, being the simplest of all the methods proposed in this chapter, failed to show any positive impact on the model's performance.
- * Two configurations of CLSTM are considered to investigate their effectiveness. The output from the CLSTM module passed to the FCN is configured in two ways. In the first, the final output of the CLSTM module is directly fed to the FCN, while the second concatenates the output from the first hidden layer with the last one. The performance of 'car' and 'pedestrian' in the second configuration of the CLSTM module improved the nuScenes detection score (NDS) by $\sim 4\%$ and also reduced the true positive error metric of both classes. Hence, the second configuration of the CLSTM module was chosen to evaluate the performance of the F-ConvNet with CLSTM architecture.
- * Three configurations of the architecture focusing on the placement of the KAM module were explored to study the influence of the attention mechanism to focus on the input's salient region. The third configuration, where the PointNet features, extracted from the PointNet, are passed to the KAM to learn the salient regions within the feature map, fed to the FCN and then to the regression branch with the KAM module to predict the 3D bounding boxes. The model's performance improved the NDS value by $\sim 2\%$ with single-frame reference and $\sim 4\%$ with multiple-frame reference compared to the frame stacking approach.
- * The performance of the 'pedestrian' showed significant improvement when the region proposals were represented with a feature vector length equal to 512. The second configuration of CLSTM and CLSTM with KAM improved the models' performance for 'pedestrians' using a feature vector size of 512. These newly obtained 'pedestrian' results are collated with the 'car', which was trained with a feature vector size of 1,024. This further boosted

the NDS values by $\sim 3\%$ (**0.8314** \rightarrow **0.8666**) for convolutional-LSTM and $\sim 1.5\%$ (**0.8736** \rightarrow **0.8805**) for convolutional-LSTM with kernel attention mechanism compared the results achieved by using the feature vector size of 1,024 for both the classes. The mean true positive metric scores are also improved.

- * The inclusion of temporal information considerably reduced the true positive metric, specifically the orientation error of the 3D bounding box from **0.819** to **0.463** and **0.294** to **0.107** for ‘cars’ and ‘pedestrian’ classes respectively on the chosen subset of nuScenes training data. The overall nuScenes detection score (NDS) was also improved from **0.822** to **0.8736** compared to the reference baseline.

- **Research Question 3 (RQ3): How best to fuse multimodal and temporal data into object detection networks, and what fusion schemes need to be employed? Where multiple layers are used to incorporate data, how does the inclusion of additional layers impact the complexity (ie., time and memory usage) of the network?**

Combining multiple modalities exploits the individual characteristics of the data employed to train the detection network, which eventually assists the network in learning a better representation of the data resulting in reduced errors and improved accuracy. RQ3 aims to investigate the fusion strategies to fuse the lidar and image data representations into the detection network to improve performance compared with using a single modality. Chapter 6 explore the fusion strategies for combining the lidar and image data for estimating the object parameters. However, any additional functionality or layer added to the framework increases the network’s complexity regarding the number of parameters to be trained, memory usage, per-frame computation time, and model size. Chapter 6 provides a summary of all the network configurations employed in this research to understand the complexity associated with the network depending on the modality employed. Finally, the following conclusions are drawn:

- The F-convNet was trained on the KITTI dataset with lidar and image data. The 3D detection results with the joint multimodal fusion (JMF) strategy achieve an

improvement of $\sim 2\text{-}3\%$ over the baseline for most cases except for the moderate case of the ‘pedestrian’ class, where improvement is around $\sim 7\%$. In the case of BEV detection, results are an improvement of $\sim 1\%$ for ‘car’ using intermediate multimodal fusion (IMF) and $\sim 1\%$ for ‘pedestrian’ and ‘cyclist’ is observed with JMF strategy except for the easy case of ‘pedestrian’ class which achieved an improvement of $\sim 2\%$ using global multimodal fusion (GMF) strategy. Overall the performance of the model is improved by fusing the image and lidar data to estimate the 3D bounding box.

- The F-ConvNet model incorporating the spatio-temporal information and the multimodal data was trained on the customised nuScenes dataset using both image and lidar data. The results show that the mAP values are slightly lower than the baseline. On the other hand, the mATE, mAOE, mASE and NDS values showed improvements when spatio-temporal information learnt from the multiple modalities was used for estimating the 3D box parameters. The JMF method improved the NDS by $\sim 2\%$. However, the mATE value obtained using the baseline model was better than the fusion strategies. The true positive error score and NDS showed further improvement with the inclusion of the attention mechanism. Overall the performance of the model in terms of nuScenes detection metrics was improved by $\sim 3\%$ for F-ConvNet with CLSTM and $\sim 4\%$ for F-ConvNet with CLSTM and KAM module with both image and lidar data are fused to estimate the 3D bounding box parameters.
- The resource utilization and computation time analysis of the various models used in this study showed that the average per-frame computation time is directly proportional to the number of trainable parameters in the network. The fusion methods have a higher per-frame computation time than the F-ConvNet, which increased almost five-fold. However, the inclusion of this complexity also improved the model performance by almost $\sim 8\%$ with reduced true-positive metric scores.

7.2 Research Contributions

The contribution of this research work are listed as follows:

- Recommending a data representation to effectively aid the detection architecture to learn the spatial features of the object and the temporal information from the sequence data.
 - Analysing the influence of prior localisation of the objects using an off-the-shelf pre-trained 2D detector while extracting the lidar point cloud features and its effect on final accuracy in AP (%) on the KITTI dataset.
 - Analysing the minimum number of sequenced frames required to effectively extract the temporal information of the object to propagate into the detector network for understanding the structural variation through lidar point cloud distribution to handle the misalignment problems during the final estimation.
 - Analysing the impact of using single sweep over multiple-sweep lidar point cloud data for training and estimating the 3D bounding box parameters on the nuScenes dataset.
 - Analysing the effect of varying the feature vector length of lidar data representation on the performance of the detection model on the KITTI and nuScenes dataset.
- Proposal of optimum attention mechanism configuration to focus on the salient points of the lidar point cloud distribution within the region proposals representing the object.
- Incorporating spatio-temporal information coupled with attention mechanism into the 3D detection architecture to improve 3D detection accuracy and reduce the true-positive error metrics.
- Evaluation of fusion strategies to fuse image and lidar data to improve the 3D object estimations.
- Analysis of the computational and resource complexity while handling multiple modalities to improve 3D detection accuracy.

7.3 Future Research

Overall, this thesis has established the importance and utility of multi-modal systems for refined 3D object detection and proposed a complex pipeline incorporating spatial, temporal and attention mechanisms to improve specific and general class accuracy demonstrated on key autonomous driving datasets. Further, the understanding of the structural information of the object using a lidar point cloud is enhanced by learning the temporal aspect of the sequenced data. However, the feasibility of applying the proposed framework in the context of real-world applications in autonomous vehicles can be still considered only for altering the driver for possible collision detection while driving on the road without the algorithm taking full control of the steering or vehicle control. Although, for fully autonomous vehicles where the driver has no control over the steering the proposed framework cannot be applied presently as this requires optimisation and adaptation to the new sensor information with higher resolution and perception capabilities.

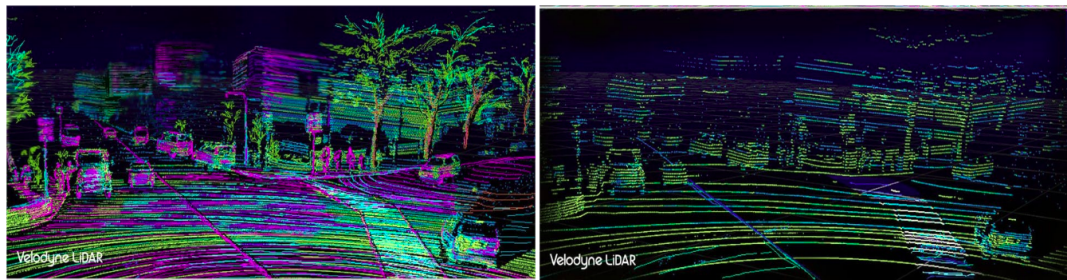


Figure 7.1: Visualisation of Velodyne’s new VLS-128 LiDAR sensor data captured for an urban scene, left, has 10 times the resolution of an image from its previous captured sensor Velodyne’s HDL-64.

The use of a newer sensor will definitely benefit the existing SOTA and proposed framework as the network will be fed with more information for understanding the environment as shown in Fig. 7.1. The Velodyne’s VLS-128 sensors have 10 times higher resolution and the ability to process data without using a vehicle’s onboard computer, thus saving compute power for other vehicle functions compared with the previous model Velodyne HDL-64. However, training the existing or newer framework on this dataset may not be possible at the moment as annotation for this new data is not available. This also requires adaptation of the framework to understand and process rich data as lidar and camera sensor operates

at different frequencies leading to synchronization problems. Another bottleneck that is observed with the proposed framework is that it is not optimised to be able to use in real-world applications, which could be an overhead to the processing and memory budget allocated to the perception system of the autonomous vehicle.

In the current research, the inclusion of an RNN-based convolutional-LSTM module and attention mechanism module has increased the computation and resource complexity in the architecture. However, this inclusion has proven the significance of these additional features extracted from the input data to aid the proposed deep learning-based model to learn the contextual information within the region proposals passed for estimating the 3D bounding boxes with a very low objectiveness score (confidence score). The experiments conducted in this research produced evidence of the following thought process.

- Adopting anchor-free techniques recently proposed in 3DSSD [59], CentrePoint [26], and CentreNet3D [55] methods within the frustum region of the proposed architecture for making the pipeline data and class independent.
- Incorporating a feedback mechanism to keep track of objects lost due to occlusion or detection failure by predicting the location of the target ‘X’ number of frames ahead [34] to enable spatio-temporal context understanding of the associated objects and the scene in a complex driving scenario. The feedback mechanism will also assist the detector module in handling missed detections due to occlusion or regions with low confidence scores.
- A detector module with a feedback mechanism of missed objects and prior estimation of the object location in future sequences would benefit the current research as it employed ground truth annotations. Furthermore, using a 2D detector module will optimise the feature extraction process and improve the model’s overall computation time as the image features can be shared.
- Exploring other fusion strategies to investigate the effectiveness of multiple modalities to corroborate during sensor failure.

Bibliography

- [1] Xuyang Bai et al. “Transfusion: Robust lidar-camera fusion for 3d object detection with transformers”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 1090–1099.
- [2] Junhyung Kang et al. “A Survey of Deep Learning-Based Object Detection Methods and Datasets for Overhead Imagery”. In: *IEEE Access* 10 (2022), pp. 20118–20134.
- [3] Rui Qian, Xin Lai, and Xirong Li. “3D object detection for autonomous driving: a survey”. In: *Pattern Recognition* (2022), p. 108796.
- [4] GM Venkatesh, Noel E O’Connor, and Suzanne Little. “Incorporating Spatio-Temporal Information in Frustum-ConvNet for Improved 3D Object Detection in Instrumented Vehicles”. In: *2022 10th European Workshop on Visual Information Processing (EUVIP)*. IEEE. 2022, pp. 1–6.
- [5] Syed Sahil Abbas Zaidi et al. “A survey of modern deep learning based object detection models”. In: *Digital Signal Processing* (2022), p. 103514.
- [6] Muhammad Ahmed et al. “Survey and performance analysis of deep learning based object detection in challenging environments”. In: *Sensors* 21.15 (2021), p. 5116.
- [7] Deyun Dai et al. “A Review of 3D Object Detection for Autonomous Driving of Electric Vehicles”. In: *World Electric Vehicle Journal* 12.3 (2021), p. 139.
- [8] Selma Deac, Flaviu Vancea, and Sergiu Nedevschi. “MVGNet: 3D object detection using Multi-Volume Grid representation in urban traffic scenarios”. In: *2021 IEEE 17th International Conference on Intelligent Computer Communication and Processing (ICCP)*. IEEE. 2021, pp. 101–108.
- [9] Emeç Erçelik, Ekim Yurtsever, and Alois Knoll. “3D Object Detection With Multi-Frame RGB-Lidar Feature Alignment”. In: *IEEE Access* 9 (2021), pp. 143138–143149.
- [10] Emeç Erçelik, Ekim Yurtsever, and Alois Knoll. “Temp-Frustum Net: 3D object detection with temporal fusion”. In: *2021 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2021, pp. 1095–1101.
- [11] Scott Ettinger et al. “Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 9710–9719.
- [12] Venkatesh GM, Bianca Pereira, and Suzanne Little. “Urban footpath image dataset to assess pedestrian mobility”. In: *Proceedings of the 1st International Workshop on Multimedia Computing for Urban Data*. 2021, pp. 23–30.
- [13] Feiyan Hu et al. “Utilising visual attention cues for vehicle detection and tracking”. In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE. 2021, pp. 5535–5542.

- [14] R. Kesten et al. *Level 5 Perception Dataset 2020*. 2021. URL: <https://level-5.global/data/perception/>.
- [15] Junho Koh et al. “Joint 3D Object Detection and Tracking Using Spatio-Temporal Representation of Camera Image and LiDAR Point Clouds”. In: *arXiv preprint arXiv:2112.07116* (2021).
- [16] Ankit Laddha et al. “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.
- [17] Jiageng Mao et al. “One million scenes for autonomous driving: Once dataset”. In: *arXiv preprint arXiv:2106.11037* (2021).
- [18] Ramin Nabati and Hairong Qi. “Centerfusion: Center-based radar and camera fusion for 3d object detection”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2021, pp. 1527–1536.
- [19] Xinshuo Weng Yunze Man Jinhyung Park, Ye Yuan Dazhi Cheng, and Matthew O’Toole Kris Kitani. “All-In-One Drive: A Comprehensive Perception Dataset with High-Density Long-Range Point Clouds”. In: (2021).
- [20] Matthew Pitropov et al. “Canadian adverse driving conditions dataset”. In: *The International Journal of Robotics Research* 40.4-5 (2021), pp. 681–690.
- [21] Jacob Solawetz. *How to Train YOLOv5 On a Custom Dataset*. Mar. 2021. URL: <https://blog.roboflow.com/how-to-train-yolov5-on-a-custom-dataset/>.
- [22] *ultralytics/yolov5: v4.0 - nn.SiLU() activations, Weights & Biases logging, PyTorch Hub integration* | Zenodo. <https://zenodo.org/record/4418161#.YQXtK6IzZuQ>. (Accessed on 07/25/2021). Jan. 2021.
- [23] Chunwei Wang et al. “PointAugmenting: Cross-Modal Augmentation for 3D Object Detection”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 11794–11803.
- [24] Yi Wu et al. “Multi-modal 3d object detection by 2d-guided precision anchor proposal and multi-layer fusion”. In: *Applied Soft Computing* 108 (2021), p. 107405.
- [25] Renjie Xu et al. “A Forest Fire Detection System Based on Ensemble Learning”. In: *Forests* 12.2 (2021), p. 217.
- [26] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. “Center-based 3d object detection and tracking”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 11784–11793.
- [27] Wu Zheng et al. “Cia-ssd: Confident iou-aware single-stage object detector from point cloud”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 35. 4. 2021, pp. 3555–3562.
- [28] Zhikang Zou et al. “The Devil Is in the Task: Exploiting Reciprocal Appearance-Localization Features for Monocular 3D Object Detection”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 2713–2722.

- [29] Alejandro Barrera et al. “Birdnet+: End-to-end 3d object detection in lidar bird’s eye view”. In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2020, pp. 1–6.
- [30] Alex Bewley et al. “Range conditioned dilated convolutions for scale invariant 3d object detection”. In: *arXiv preprint arXiv:2005.09927* (2020).
- [31] Mario Bijelic et al. “Seeing Through Fog Without Seeing Fog: Deep Multimodal Sensor Fusion in Unseen Adverse Weather”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.
- [32] Holger Caesar et al. “nusenes: A multimodal dataset for autonomous driving”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 11621–11631.
- [33] Di Feng et al. “Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges”. In: *IEEE Transactions on Intelligent Transportation Systems* 22.3 (2020), pp. 1341–1360.
- [34] Jaime B Fernandez, Suzanne Little, and Noel E O’Connor. “Multiple Path Prediction for Traffic Scenes using LSTMs and Mixture Density Models.” In: *VEHITS*. 2020, pp. 481–488.
- [35] Jakob Geyer et al. “A2d2: Audi autonomous driving dataset”. In: *arXiv preprint arXiv:2004.06320* (2020).
- [36] Chenhang He et al. “Structure aware single-stage 3d object detection from point cloud”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 11873–11882.
- [37] John Houston et al. “One thousand and one hours: Self-driving motion prediction dataset”. In: *arXiv preprint arXiv:2006.14480* (2020).
- [38] Rui Huang et al. “An lstm approach to temporal 3d object detection in lidar point clouds”. In: *European Conference on Computer Vision*. Springer. 2020, pp. 266–282.
- [39] Tengting Huang et al. “Epnet: Enhancing point features with image semantics for 3d object detection”. In: *European Conference on Computer Vision*. Springer. 2020, pp. 35–52.
- [40] Jianhao Jiao et al. “MLOD: Awareness of extrinsic perturbation in multi-LiDAR 3D object detection for autonomous driving”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 10556–10563.
- [41] Motaz Khader and Samir Cherian. “An introduction to automotive lidar”. In: *Texas Instruments* (2020).
- [42] Hongwu Kuang et al. “Voxel-FPN: Multi-scale voxel feature aggregation for 3D object detection from LIDAR point clouds”. In: *Sensors* 20.3 (2020), p. 704.
- [43] KS Chidanand Kumar and Samir Al-Stouhi. “Real-time Spatial-temporal Context Approach for 3D Object Detection using LiDAR.” In: *VEHITS*. 2020, pp. 432–439.
- [44] Jiale Li et al. “3D IoU-Net: IoU guided 3D object detector for point clouds”. In: *arXiv preprint arXiv:2004.04962* (2020).
- [45] Li Liu et al. “Deep learning for generic object detection: A survey”. In: *International journal of computer vision* 128.2 (2020), pp. 261–318.

-
- [46] Ruixu Liu et al. “Attention mechanism exploits temporal contexts: Real-time 3d human pose reconstruction”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 5064–5073.
 - [47] Zhe Liu et al. “Tanet: Robust 3d object detection from point clouds with triple attention”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 07. 2020, pp. 11677–11684.
 - [48] Scott McCrae and Avidesh Zakhori. “3D object detection for autonomous driving using temporal LiDAR data”. In: *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2020, pp. 2661–2665.
 - [49] Mircea Paul Muresan, Ion Giosan, and Sergiu Nedevschi. “Stabilization and validation of 3D object position using multimodal sensor fusion and semantic segmentation”. In: *Sensors* 20.4 (2020), p. 1110.
 - [50] Quang-Hieu Pham et al. “A* 3D dataset: Towards autonomous driving in challenging environments”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 2267–2273.
 - [51] Charles R. Qi et al. “ImVoteNet: Boosting 3D Object Detection in Point Clouds With Image Votes”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.
 - [52] Shaoshuai Shi et al. “Pv-rcnn: Point-voxel feature set abstraction for 3d object detection”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 10529–10538.
 - [53] Pei Sun 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.
 - [54] Yonglin Tian et al. “Adaptive and azimuth-aware fusion network of multimodal local features for 3D object detection”. In: *Neurocomputing* 411 (2020), pp. 32–44.
 - [55] Guojun Wang et al. “Centernet3d: An anchor free object detector for autonomous driving”. In: *arXiv preprint arXiv:2007.07214* (2020).
 - [56] Longyin Wen et al. “UA-DETRAC: A New Benchmark and Protocol for Multi-Object Detection and Tracking”. In: *Computer Vision and Image Understanding* (2020).
 - [57] Yutian Wu et al. “Deep 3D object detection networks using LiDAR data: A review”. In: *IEEE Sensors Journal* 21.2 (2020), pp. 1152–1171.
 - [58] Liang Xie et al. “PI-RCNN: An efficient multi-sensor 3D object detector with point-based attentive cont-conv fusion module”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 07. 2020, pp. 12460–12467.
 - [59] Zetong Yang et al. “3dssd: Point-based 3d single stage object detector”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 11040–11048.
 - [60] Hongwei Yi et al. “Segvoxelnet: Exploring semantic context and depth-aware features for 3d vehicle detection from point cloud”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 2274–2280.
 - [61] Ekim Yurtsever et al. “A survey of autonomous driving: Common practices and emerging technologies”. In: *IEEE access* 8 (2020), pp. 58443–58469.

- [62] Yin Zhou et al. “End-to-end multi-view fusion for 3d object detection in lidar point clouds”. In: *Conference on Robot Learning*. PMLR. 2020, pp. 923–932.
- [63] Eduardo Arnold et al. “A survey on 3d object detection methods for autonomous driving applications”. In: *IEEE Transactions on Intelligent Transportation Systems* 20.10 (2019), pp. 3782–3795.
- [64] Garrick Brazil and Xiaoming Liu. “M3d-rpn: Monocular 3d region proposal network for object detection”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 9287–9296.
- [65] Åsmund Brekke, Fredrik Vatsendvik, and Frank Lindseth. “Multimodal 3d object detection from simulated pretraining”. In: *Symposium of the Norwegian AI Society*. Springer. 2019, pp. 102–113.
- [66] Pei Cao et al. “Multi-view frustum pointnet for object detection in autonomous driving”. In: *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2019, pp. 3896–3899.
- [67] Yilun Chen et al. “Fast point r-cnn”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 9775–9784.
- [68] Di Hu et al. “Dense multimodal fusion for hierarchically joint representation”. In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 3941–3945.
- [69] Xinyu Huang et al. “The apolloscape open dataset for autonomous driving and its application”. In: *IEEE transactions on pattern analysis and machine intelligence* 42.10 (2019), pp. 2702–2719.
- [70] Alex H Lang et al. “Pointpillars: Fast encoders for object detection from point clouds”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 12697–12705.
- [71] Ming Liang et al. “Multi-task multi-sensor fusion for 3d object detection”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 7345–7353.
- [72] Zhijian Liu et al. “Point-voxel cnn for efficient 3d deep learning”. In: *Advances in Neural Information Processing Systems* 32 (2019).
- [73] Yuexin Ma et al. “Trafficpredict: Trajectory prediction for heterogeneous traffic-agents”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 6120–6127.
- [74] Gregory P Meyer et al. “Lasernet: An efficient probabilistic 3d object detector for autonomous driving”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 12677–12686.
- [75] Gregory P Meyer et al. “Sensor fusion for joint 3d object detection and semantic segmentation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2019.
- [76] Felix Nobis et al. “A deep learning-based radar and camera sensor fusion architecture for object detection”. In: *2019 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*. IEEE. 2019, pp. 1–7.

- [77] Abhishek Patil et al. “The h3d dataset for full-surround 3d multi-object detection and tracking in crowded urban scenes”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 9552–9557.
- [78] Charles R Qi et al. “Deep hough voting for 3d object detection in point clouds”. In: *proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 9277–9286.
- [79] Francisca Rosique et al. “A systematic review of perception system and simulators for autonomous vehicles research”. In: *Sensors* 19.3 (2019), p. 648.
- [80] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. “Pointcnn: 3d object proposal generation and detection from point cloud”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 770–779.
- [81] Kiwoo Shin, Youngwook Paul Kwon, and Masayoshi Tomizuka. “Roarnet: A robust 3d object detection based on region approximation refinement”. In: *2019 IEEE intelligent vehicles symposium (IV)*. IEEE. 2019, pp. 2510–2515.
- [82] Andrea Simonelli et al. “Disentangling monocular 3d object detection”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 1991–1999.
- [83] Vishwanath A Sindagi, Yin Zhou, and Oncel Tuzel. “Mvx-net: Multimodal voxelnet for 3d object detection”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 7276–7282.
- [84] GM Venkatesh et al. “Saliency guided 2d-object annotation for instrumented vehicles”. In: *2019 International Conference on Content-Based Multimedia Indexing (CBMI)*. IEEE. 2019, pp. 1–7.
- [85] Zhixin Wang and Kui Jia. “Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 1742–1749.
- [86] Zetong Yang et al. “Std: Sparse-to-dense 3d object detector for point cloud”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 1951–1960.
- [87] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. “Objects as points”. In: *arXiv preprint arXiv:1904.07850* (2019).
- [88] Waleed Ali et al. “Yolo3d: End-to-end real-time 3d oriented object bounding box detection from lidar point cloud”. In: *Proceedings of the European conference on computer vision (ECCV) workshops*. 2018.
- [89] Sean Campbell et al. “Sensor technology in autonomous vehicles: A review”. In: *2018 29th Irish Signals and Systems Conference (ISSC)*. IEEE. 2018, pp. 1–4.
- [90] Karanbir Singh Chahal and Kuntal Dey. “A survey of modern object detection literature using deep learning”. In: *arXiv preprint arXiv:1808.07256* (2018).
- [91] Ahmad El Sallab et al. “Yolo4d: A spatio-temporal approach for real-time multi-object detection and classification from lidar point clouds”. In: (2018).

- [92] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. “3d semantic segmentation with submanifold sparse convolutional networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 9224–9232.
- [93] Jason Ku et al. “Joint 3d proposal generation and object detection from view aggregation”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 1–8.
- [94] Ming Liang et al. “Deep continuous fusion for multi-sensor 3d object detection”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 641–656.
- [95] Wenjie Luo, Bin Yang, and Raquel 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.
- [96] Kazuki Minemura et al. “LMNet: Real-time multiclass object detection on CPU using 3D LiDAR”. In: *2018 3rd Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*. IEEE. 2018, pp. 28–34.
- [97] Charles R Qi et al. “Frustum pointnets for 3d object detection from rgb-d data”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 918–927.
- [98] J. Redmon and A. Farhadi. “Tiny-Yolo3D”. In: 2018.
- [99] Joseph Redmon and Ali Farhadi. “Yolov3: An incremental improvement”. In: *arXiv preprint arXiv:1804.02767* (2018).
- [100] Martin Simony et al. “Complex-yolo: An euler-region-proposal for real-time 3d object detection on point clouds”. In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. 2018.
- [101] Danfei Xu, Dragomir Anguelov, and Ashesh Jain. “Pointfusion: Deep sensor fusion for 3d bounding box estimation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 244–253.
- [102] Yan Yan, Yuxing Mao, and Bo Li. “Second: Sparsely embedded convolutional detection”. In: *Sensors* 18.10 (2018), p. 3337.
- [103] Bin Yang, Wenjie Luo, and Raquel 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.
- [104] Zetong Yang et al. “Ipod: Intensive point-based object detector for point cloud”. In: *arXiv preprint arXiv:1812.05276* (2018).
- [105] Yiming Zeng et al. “Rt3d: Real-time 3-d vehicle detection in lidar point cloud for autonomous driving”. In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 3434–3440.
- [106] Zijun Zhang. “Improved adam optimizer for deep neural networks”. In: *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*. Ieee. 2018, pp. 1–2.

-
- [107] Yin Zhou and Oncel Tuzel. “Voxelnet: End-to-end learning for point cloud based 3d object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4490–4499.
 - [108] Abdullahi Daniyan et al. “Kalman-Gain Aided Particle PHD Filter for Multitarget Tracking”. In: *IEEE Transactions on Aerospace and Electronic Systems* 53.5 (2017), pp. 2251–2265.
 - [109] Kaiming He et al. “Mask R-CNN”. In: *Proc of IEEE international conference on computer vision*. 2017, pp. 2961–2969.
 - [110] Bo Li. “3d fully convolutional network for vehicle detection in point cloud”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 1513–1518.
 - [111] Zeming Li et al. “Light-head r-cnn: In defense of two-stage object detector”. In: *arXiv preprint arXiv:1711.07264* (2017).
 - [112] Tsung-Yi Lin et al. “Focal loss for dense object detection”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2980–2988.
 - [113] Junting Pan et al. “Salgan: Visual saliency prediction with generative adversarial networks”. In: *arXiv preprint arXiv:1701.01081* (2017).
 - [114] Scott Drew Pendleton et al. “Perception, planning, control, and coordination for autonomous vehicles”. In: *Machines* 5.1 (2017), p. 6.
 - [115] Charles R Qi et al. “Pointnet: Deep learning on point sets for 3d classification and segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 652–660.
 - [116] Charles Ruizhongtai Qi et al. “Pointnet++: Deep hierarchical feature learning on point sets in a metric space”. In: *Advances in neural information processing systems* 30 (2017).
 - [117] Joseph Redmon and Ali Farhadi. “YOLO9000: better, faster, stronger”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7263–7271.
 - [118] Hengshuang Zhao et al. “Pyramid scene parsing network”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2881–2890.
 - [119] Xizhou Zhu et al. “Flow-guided feature aggregation for video object detection”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 408–417.
 - [120] Alex Bewley et al. “Simple online and realtime tracking”. In: *2016 IEEE international conference on image processing (ICIP)*. IEEE. 2016, pp. 3464–3468.
 - [121] Jifeng Dai et al. “R-fcn: Object detection via region-based fully convolutional networks”. In: *Advances in neural information processing systems* 29 (2016).
 - [122] Bo Li, Tianlei Zhang, and Tian Xia. “Vehicle detection from 3d lidar using fully convolutional network”. In: *arXiv preprint arXiv:1608.07916* (2016).
 - [123] Wei Liu et al. “Ssd: Single shot multibox detector”. In: *European conference on computer vision*. Springer. 2016, pp. 21–37.
 - [124] Anton Milan et al. “MOT16: A benchmark for multi-object tracking”. In: *arXiv preprint arXiv:1603.00831* (2016).

- [125] Joseph Redmon et al. “You only look once: Unified, real-time object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.
- [126] Keshav Bimbraw. “Autonomous cars: Past, present and future a review of the developments in the last century, the present scenario and the expected future of autonomous vehicle technology”. In: *2015 12th international conference on informatics in control, automation and robotics (ICINCO)*. Vol. 1. IEEE. 2015, pp. 191–198.
- [127] Ross Girshick. “Fast r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.
- [128] Kaiming He et al. “Spatial pyramid pooling in deep convolutional networks for visual recognition”. In: *IEEE transactions on pattern analysis and machine intelligence* 37.9 (2015), pp. 1904–1916.
- [129] Shihao Ji et al. “Blackout: Speeding up recurrent neural network language models with very large vocabularies”. In: *arXiv preprint arXiv:1511.06909* (2015).
- [130] Ming Jiang et al. “Salicon: Saliency in context”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1072–1080.
- [131] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully convolutional networks for semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.
- [132] Shaoqing Ren et al. “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: *Advances in neural information processing systems* 28 (2015).
- [133] Xingjian Shi et al. “Convolutional LSTM network: A machine learning approach for precipitation nowcasting”. In: *Advances in neural information processing systems* 28 (2015).
- [134] Ross Girshick et al. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *Proceedings of IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.
- [135] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).
- [136] Tsung-Yi Lin et al. “Microsoft coco: Common objects in context”. In: *European conference on computer vision*. Springer. 2014, pp. 740–755.
- [137] Ronald PS Mahler. *Advances in statistical multisource-multitarget information fusion*. Artech House, 2014.
- [138] Andreas Geiger et al. “Vision meets robotics: The kitti dataset”. In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1231–1237.
- [139] Andreas Geiger, Philip Lenz, and Raquel 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.
- [140] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems* 25 (2012).
- [141] Jiquan Ngiam et al. “Multimodal deep learning”. In: *ICML*. 2011.

-
- [142] Mark Everingham et al. "The pascal visual object classes (voc) challenge". In: *International journal of computer vision* 88.2 (2010), pp. 303–338.
 - [143] Keni Bernardin and Rainer Stiefelhagen. "Evaluating multiple object tracking performance: the clear mot metrics". In: *EURASIP Journal on Image and Video Processing* 2008 (2008), pp. 1–10.
 - [144] Daniel E Clark, Kusha Panta, and Ba-Ngu Vo. "The GM-PHD filter multiple target tracker". In: *2006 9th International Conference on Information Fusion*. IEEE. 2006, pp. 1–8.
 - [145] Ozgur Erdinc, Peter Willett, and Yaakov Bar-Shalom. "Probability hypothesis density filter for multitarget multisensor tracking". In: *2005 7th International Conference on Information Fusion*. Vol. 1. IEEE. 2005, 8–pp.
 - [146] L Chasmer, C Hopkinson, and P Treitz. "Assessing the three-dimensional frequency distribution of airborne and ground-based lidar data for red pine and mixed deciduous forest plots". In: *Int Arch Photogramm Remote Sens Spat Inf Sci* 36.8 (2004), W2.
 - [147] Ba-Ngu Vo, Sumeetpal Singh, Arnaud Doucet, et al. "Sequential Monte Carlo implementation of the PHD filter for multi-target tracking". In: *Proc. Int'l Conf. on Information Fusion*. 2003, pp. 792–799.
 - [148] Eric A Wan and Rudolph Van Der Merwe. "The unscented Kalman filter". In: *Kalman filtering and neural networks* (2001), pp. 221–280.