

Connected Autonomous Electromobility

Visual Navigation and Charging Analytic Frameworks



Kai Li Lim

BEng (Hons) *Nott.*, MSc *Lanc.*

Supervisor: Prof. Thomas Bräunl

Department of Electrical, Electronic and Computer Engineering
The University of Western Australia

This thesis is presented for the degree of
Doctor of Philosophy

June 2019

Last updated Tuesday 11th June, 2019

©2019 Kai Li Lim



This work is licensed under a Creative Commons
Attribution-NonCommercial-ShareAlike 4.0 International License.
The details of this license are available on the Creative Commons website:
<https://creativecommons.org/licenses/by-nc-sa/4.0/>.

In loving memory of my a-má and guā-kong ...

Declaration

I hereby certify that:

- This thesis has been substantially accomplished during enrolment in this degree.
- This thesis does not contain material which has been submitted for the award of any other degree or diploma in my name, in any university or other tertiary institution. In the future, no part of this thesis will be used in a submission in my name, for any other degree or diploma in any university or other tertiary institution without the prior approval of The University of Western Australia.
- This thesis does not contain any material previously published or written by another person, except where due reference has been made in the text and, where relevant, in the Authorship Declaration that follows.
- This thesis does not violate or infringe any copyright, trademark, patent, or other rights whatsoever of any person.

This thesis contains published work and/or work prepared for publication, some of which has been co-authored.

Kai Li Lim
June 2019

Acknowledgements

And I would like to acknowledge ...

This research was supported by an Australian Government Research Training Program (RTP) Scholarship.

This research was funded by [insert name of grant and grant identification numbers].

Technical assistance was kindly provided by [insert name of assistant(s)] for [insert description of assistance] that is described in [insert location in thesis].

Authorship Declaration

In accordance with the regulations of the Graduate Research School, this thesis contains published work and/or work prepared for publications that are co-authored. The bibliographical details of the work and where it appears in the thesis are outlined below.

1. **K. L. Lim** and T. Bräunl, "A Methodological Review of Visual Road Recognition Procedures for Autonomous Driving Applications," *arXiv:1905.01635 [cs.CV]*, May. 2019.

Paper presented as **Chapter 2**.

Estimated percentage of candidate's contribution is **90%**.

2. **K. L. Lim** and T. Bräunl, "A Review of Recent Visual Odometry Methods and Its Applications for Autonomous Driving,"

Paper presented as **Chapter 3**.

Estimated percentage of candidate's contribution is **90%**.

3. R. G. Reid, **K. L. Lim** and T. Bräunl, "Cooperative Multi-Robot Navigation — SLAM, Visual Odometry and Semantic Segmentation," in *Cooperative Localization and Navigation: Theory, Research, and Practice*, C. Gao, Ed. Boca Raton: CRC Press, 2019, [in press].

Paper presented as **Chapter 4**.

Estimated percentage of candidate's contribution is **30%**.

4. **K. L. Lim**, T. Drage and T. Bräunl, "Implementation of semantic segmentation for road and lane detection on an autonomous ground vehicle with LIDAR," in *2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, Daegu, 2017, pp. 429–434. doi: 10.1109/MFI.2017.8170358

Paper presented as **Chapter 5**.

Estimated percentage of candidate's contribution is **70%**.

5. **K. L. Lim**, T. Drage, R. Podolski, G. Meyer-Lee, S. Evans-Thompson, J. Y.-T. Lin, G. Channon, M. Poole, and T. Bräunl, "A Modular Software Framework for Autonomous Vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, Changshu, 2018, pp. 1780–1785. doi: 10.1109/IVS.2018.8500474

Paper presented as **Chapter 6**.
Estimated percentage of candidate's contribution is **35%**.
6. **K. L. Lim**, T. Drage, C. Zhang, C. Brogle, W. W. L. Lai, T. Kelliher, M. Adina-Zada and T. Bräunl, "Evolution of a Reliable and Extensible High-Level Control System for an Autonomous Car," *IEEE Transactions on Intelligent Vehicles*. doi: 10.1109/TIV.2019.2919459 [in press].

Paper presented as **Chapter 7**.
Estimated percentage of candidate's contribution is **25%**.
7. C. Brogle, C. Zhang, **K. L. Lim** and T. Bräunl, "Hardware-in-the-Loop Autonomous Driving Simulation without Real-Time Constraints," *IEEE Transactions on Intelligent Vehicles*. doi: 10.1109/TIV.2019.2919457 [in press].

Paper presented as **Chapter 8**.
Estimated percentage of candidate's contribution is **20%**.
8. **K. L. Lim**, S. Speidel and T. Bräunl, "REView: A Unified Telemetry Platform for Electric Vehicles and Charging Infrastructures," in *Connected Vehicles in the Internet of Things: Concepts, Technologies and Frameworks for the IoV*, Z. Mahmood, Ed. New York: Springer, 2019, [under review].

Paper presented as **Chapter 9**.
Estimated percentage of candidate's contribution is **65%**.
9. **K. L. Lim**, S. Speidel and T. Bräunl, "A Comparative Study of AC and DC Electric Vehicle Charging Station Usage," *Renewable & Sustainable Energy Reviews*, [under review].

Paper presented as **Chapter 10**.
Estimated percentage of candidate's contribution is **60%**.

The candidate's statement regarding his contribution to each of the works listed above are correct.

<u>Kai Li Lim</u>		
PhD Candidate	Signature	Date

<u>Thomas Drage</u>		
Co-author	Signature	Date

<u>Stuart Speidel</u>		
Co-author	Signature	Date

<u>Robert G. Reid</u>		
Co-author	Signature	Date

<u>Chao Zhang</u>		
Co-author	Signature	Date

<u>Craig Brogle</u>		
Co-author	Signature	Date

I, Thomas Bräunl, hereby certify that the students' statements regarding their contribution to each of the works listed above are correct. As all co-authors' signatures could not be obtained, I hereby authorise the inclusion of co-authored work in the thesis.

Coordinating Supervisor signature:
Date:

Abstract

The growing ubiquity of electric vehicles is often characterised through its increasing autonomy and connectivity. This has led to catalyse the foundations of smart cities and intelligent transportation systems, where the applications of electromobility is often given a pivotal role towards their realisation.

As the title suggests, this thesis presents its investigations into electromobility applications across two key areas — (1) computer vision-based autonomous driving, and (2) data management and analyses of electric vehicle charging stations.

The study into vision-based navigation aims to address the problem of developing an autonomous driving system that predominantly utilises the camera as the vehicle's primary environmental perception sensor. This research gap is attributed to the greater algorithmic complexity in computer vision, as compared to LiDARs or radars. Additionally, the general attainability of cameras, and the diminishing cost of parallel computation has further contributed towards the motivation for this study. To this end, the requirements for visual navigation are centred upon localisation and scene understanding. More specifically, this thesis describes applications pertaining to visual odometry and semantic segmentation following an extensive literature survey. These methods are first tested for its feasibility on datasets and mobile robots, and then verified on an autonomous Formula-SAE electric car as the test bed, enabling the vehicle to perform object detection, lane keeping and dead reckoning in real-time. Experiments were conducted for road scenes and traffic cone drives, yielding fast and accurate results for detections, classifications and odometry.

The electric vehicle charging station network managed by The REV Project comprise of 23 7 kW AC stations and a 50 kW DC fast charging station. Each station is connected to a centralised server over the mobile network, perpetually transmitting telemetric data to the server's daemons. The data generated from these stations effectuates the investigation into the charging behaviours across AC and DC stations, leading to the study of electric vehicle trends around Perth. Results from this study comprise of a combination of time series analyses that compares the cycles and energy consumption between AC and DC charges among local stations. A web-based telemetry monitoring platform, REView, is further described in this thesis. In addition to the charging stations, REView consolidates data from

the project's electric vehicle fleet and solar power generation into a unified framework that features on-demand monitoring for connected infrastructures. These are further detailed according to its back-end processes, encompassing its communication architectures, data management, data visualisation and presentation.

The cumulation of works that are presented here conforms to The REV Project's goal that describe contributions towards the fields of intelligent vehicles and the Internet of Vehicles. These contributions are exemplified in this thesis through the successful application of visual autonomous driving, and the analyses towards the electric vehicle trends in Perth, which should subsequently encourage further developments in this area.

Table of contents

List of figures	xxiii
List of tables	xxix
Nomenclature	xxxii
1 Introduction	1
1.1 Autonomous Driving	1
1.2 Electromobility	5
1.3 Connected Mobility	8
1.4 Contributions	10
1.5 Thesis Outline	11
2 Visual Road Recognition Review	15
2.1 Introduction	15
2.2 Conventional Methods	17
2.2.1 Horizon Detection	18
2.2.2 Vanishing Point Detection	19
2.2.3 Region of Interest Isolation	20
2.2.4 Image Classification	21
2.2.5 Model Fitting	22
2.3 Learning Methods	23
2.4 Commercial Implementations	27
2.5 Recent Works	29
2.6 Conclusion	30
3 Visual Odometry Review	33
3.1 Introduction	33
3.2 Monocular Visual Odometry	36

3.2.1	Related Applications	38
3.3	Stereoscopic Visual Odometry	39
3.3.1	Related Applications	43
3.4	Visual-Inertial Odometry	45
3.5	Discussions	47
3.6	Conclusion	48
4	Cooperative Multi-Robot Navigation	51
4.1	Introduction	51
4.2	Robot Hardware Design	53
4.3	Cooperative Localisation and Navigation	54
4.3.1	Mapping	54
4.3.2	MR-SLAM Architecture	56
4.3.3	SLAM Implementation	59
4.3.4	UGV/GCS Communications	60
4.3.5	Loop Closures	61
4.3.6	SLAM Evaluation	62
4.4	Visual Odometry	65
4.4.1	Visual Odometry Method	65
4.4.2	Visual Odometry Evaluation	66
4.5	Semantic Segmentation	67
4.5.1	Semantic Segmentation Method	68
4.5.2	Semantic Segmentation Evaluation	68
4.6	Conclusion	70
5	Semantic Segmentation for Road and Lane Detection	71
5.1	Introduction	72
5.2	Implementation	73
5.2.1	Application Environment	75
5.2.2	Autonomous Driving Procedures	75
5.3	Testing and Evaluations	76
5.3.1	Methodology	76
5.3.2	Results and Discussions	80
5.4	Conclusion	85
6	A Modular Software Framework for Autonomous Vehicles	87
6.1	Introduction	87

6.2	Autonomous Driving Framework	89
6.2.1	Path Planner	89
6.2.2	Software Communications	92
6.2.3	Localisation	93
6.2.4	Odometry	94
6.2.5	LiDAR	94
6.2.6	Visual Navigation	95
6.2.7	Safety Trip Monitor	97
6.2.8	Controller	97
6.3	Implementation on SAE Vehicle	98
6.4	Results	99
6.5	Conclusion	100
7	Evolution of a Reliable and Extensible High-Level Control System	101
7.1	Introduction	101
7.2	System Overview	104
7.3	Navigation Sensors	106
7.3.1	Odometry	106
7.3.2	Dead Reckoning	107
7.3.3	LiDAR System	108
7.3.4	Camera System	109
7.4	Path Planning	110
7.4.1	Waypoint Driving	110
7.4.2	Cone Driving	111
7.5	Visual Navigation	112
7.5.1	Road and Lane Detection	113
7.5.2	Visual Odometry	115
7.5.3	Cone Detection	116
7.6	Hardware-in-the-Loop Simulation	117
7.7	System Validation	118
7.7.1	Sensor Fusion	118
7.7.2	Waypoint Driving	119
7.7.3	Cone Driving	121
7.7.4	Driving Simulation	124
7.8	Conclusion	126

8	Hardware-in-the-Loop Autonomous Driving Simulation	127
8.1	Introduction	127
8.2	Software Framework	129
8.3	Driving Simulator	131
8.3.1	Performance and Suitability for Wall-clock Time Operation	136
8.3.2	Time Synchronisation	137
8.3.3	Simulation Benefits	137
8.4	Sensors, Navigation and Path Planning	138
8.4.1	LiDAR System	139
8.4.2	Camera System	140
8.4.3	Path Planning	141
8.5	Experiments and Results	142
8.5.1	Vehicle Dynamics	142
8.5.2	LiDAR Cone Detection	143
8.5.3	Visual Cone Detection	146
8.5.4	Compute Hardware Load	148
8.5.5	Response Time	150
8.6	Future Work	150
8.7	Conclusion	151
9	REView	153
9.1	Introduction	154
9.2	Background	156
9.2.1	Local and International Adoption of Electric Vehicles and Charging Stations	156
9.2.2	Importance of Measuring Environmental Impact	157
9.2.3	Telemetry Platforms and Networks	159
9.3	System Design Overview	161
9.4	Charging Infrastructures	162
9.4.1	DC Charging	162
9.4.1.1	Communication Protocols	163
9.4.1.2	User Authentication	164
9.4.1.3	Data Visualisation	165
9.4.2	AC Charging	167
9.4.2.1	Communication Protocols	169
9.4.2.2	Telemetry Parameters	170
9.4.2.3	User Authentication	171

9.4.2.4	Database	172
9.4.2.5	Data Visualisation	175
9.5	Vehicle Monitoring	177
9.5.1	Communication Protocols	177
9.5.2	Database	179
9.5.3	Data Visualisation	180
9.5.3.1	Vehicle Tracking	181
9.5.3.2	Driving Statistics	182
9.5.3.3	Heat Maps	184
9.5.3.4	Journey Logs	186
9.6	EV Charging Power Generation	186
9.6.1	Data Visualisation	187
9.7	Usage Billing	188
9.7.1	User Billing	189
9.7.2	Station Operator Billing	190
9.7.3	Network Overview	191
9.8	Mobile Application	193
9.9	Results	194
9.9.1	Overall Energy Usage	194
9.9.2	Charging Infrastructure Usage	195
9.9.3	Solar PV Monitoring	198
9.9.4	Heat Maps for EV Tracking	198
9.9.5	Charging Infrastructure Usage Forecast	199
9.10	Conclusion	203
10	A Comparative Study of AC and DC Electric Vehicle Charging Station Usage	205
10.1	Introduction	205
10.2	AC and DC Charging Infrastructure	207
10.3	Types of EV Charging	209
10.3.1	Typical Charging Cycle	210
10.3.2	Limitations on Charging Speed	211
10.3.3	Authentication and Billing	211
10.3.4	Driving Efficiency and Battery Size for EV	212
10.3.5	Inductive Charging	212
10.3.6	Level-1 Charging (IEC 62196-3 Mode 2)	213
10.3.7	Level-2 Charging (IEC 61851-3 Mode 3)	213
10.3.8	DC-Fast Charging (IEC 61851-3 Mode 4)	213

10.3.9 Alternate Methods	213
10.3.10 Charging Speed Comparison	213
10.3.11 Australian Charging Standard Preference	214
10.3.12 International EV Plug Adoption	215
10.4 Analysis of Charging Station Usage	216
10.4.1 AC Charging and Maintaining Charge	216
10.4.2 AC versus DC Station Comparison (CS vs DC)	221
10.4.3 DC Station Comparison	224
10.4.4 DC Charging Connectors Used	230
10.5 Cost Modelling	231
10.6 Validation	236
10.7 Conclusion	236
11 Conclusions	239
11.1 Overall Findings	239
11.2 Future Research Recommendations	241
11.3 Final Remarks	242
References	245
Appendix A Robust Deep Learning System for Multi-Sensor Road Detection	277
A.1 Introduction	277
A.2 Problem Description	278
A.3 Sensor Configuration	279
A.3.1 Computer Vision	279
A.3.2 LiDAR	279
A.4 Proposed Pipeline and Algorithms	279
A.4.1 LiDAR Road Edge Detection	280
A.4.2 Semantic Image Segmentation	280
A.4.3 End-to-End Deep Learning	280
A.4.4 Simulation Framework and Experiments	281

List of figures

1.1	An EasyMile EZ10 driverless shuttle at a UWA charging station.	2
1.2	An autonomous car with its sensors visibly mounted on its roof.	3
1.3	SAE J3016	4
1.4	The REV Project’s autonomous Formula SAE Electric test vehicle.	6
1.5	UWA’s AC and DC charging stations.	8
2.1	Edge-less horizon detection	19
2.2	Vanishing point estimation	20
2.3	Lane detection with Hough transform performed using OpenCV	23
2.4	SegNet on a Western Australian road scene	26
2.5	KittiSeg on the Citiscapes dataset	27
2.6	Self-Driving Car Nanodegree output	28
4.1	MRS UGV with hardware modules	53
4.2	MR-SLAM architecture and software development diagram	54
4.3	MR-SLAM class diagram	58
4.4	Lengths and angles used for calculating the local SLAM prefilter.	59
4.5	Occupancy grid-map fusion	62
4.6	SLAM output (starting)	63
4.7	SLAM output (loop closure)	64
4.8	Completed global grid-map	64
4.9	Multimodal constraint output	65
4.10	Tracked ORB features	67
4.11	ORB-SLAM2 trajectory	67
4.12	SegNet output 1	69
4.13	SegNet output 2	69
5.1	Camera mounting location	74
5.2	Lane distance measurements with SegNet	76

5.3	Bollards' position with reference to the car	77
5.4	Topological distance between bollards and car camera	78
5.5	Frame captured from the vehicle's camera for distance calibration.	79
5.6	Calibration LiDAR plot	79
5.7	SegNet results on a Perth dual carriageway	80
5.8	Segmentation results from a parking area on campus grounds.	81
5.9	LiDAR plot at the position of Fig. 5.8	81
5.10	Detected objects and their errors	82
5.11	Segmentation results on pavement between faculty buildings.	83
5.12	Segmentation results at a road junction.	83
5.13	Segmentation results on a road with pronounced shadows.	84
6.1	The software architecture diagram of our proposed framework.	90
6.2	Simulated near-optimal path and manoeuvre selection	92
6.3	LiDAR plot at the position of Fig. 6.4	95
6.4	Segmentation results from a parking area on campus grounds.	96
6.5	Generated trajectory base frame	98
6.6	Map showing the path taken by the vehicle in with a solid red line.	99
6.7	Visual navigation output	100
7.1	Autonomous SAE Vehicle.	102
7.2	SAE vehicle software framework.	105
7.3	The SAE vehicle's rack-mounted sensors.	106
7.4	Captured scene with its instantaneous LiDAR point cloud	109
7.5	Calculated waypoints on RViz with inputs from (7.2) and (7.3).	111
7.6	Semantic segmentation results from test drive	114
7.7	ORB-SLAM2 experiment	116
7.8	Simulation software framework.	118
7.9	Generated path projections, ground truth and linear displacements	120
7.10	Experimental setup for cone driving.	121
7.11	Visual cone detection showing the detected cones in bounding boxes.	122
7.12	Visualisation of the path planner on cone driving	123
7.13	Real and simulated LiDAR scenarios for visual cone processing	125
7.14	Real and simulated LiDAR output for visual cone processing	125
8.1	Experimental track setup	128
8.2	High-level architecture of the Autoware project	130
8.3	Division of ROS node responsibilities.	131

8.4	Autonomous driving simulator hardware diagram.	132
8.5	Autonomous driving simulator setup.	133
8.6	FSAE vehicle ROS node structure.	134
8.7	Software framework architecture with CARLA simulator interface.	135
8.8	Comparison of frame rates	137
8.9	Autonomous FSAE sensor rack.	139
8.10	Scenarios used for comparing real and simulated cone processing outputs	144
8.11	LiDAR-based cone detection results from real and simulated scenarios	144
8.12	LiDAR point clouds from real and simulated scenarios	145
8.13	Number of LiDAR points returned	145
8.14	Computer vision based cone detection results	146
8.15	Cone detection algorithm processing times for real and simulated images.	147
8.16	Processor utilisation during LiDAR-based cone processing	149
8.17	Vehicle response times for real and simulated vehicles.	150
9.1	Network architecture of REView.	161
9.2	UWA's Tritium Veefil-RT DC fast charging station.	163
9.3	Simplified class diagram of REView's DC station management.	163
9.4	RFID cards supported by the DC station	165
9.5	Examples of visualisations for the DC station	166
9.6	DC charges table	167
9.7	UWA's dual outlet Elektromotive Elektrobay AC charging station.	168
9.8	Simplified class diagram showing REView's AC station management.	168
9.9	A Four-Faith F2414 M2M modem used in AC charging stations.	169
9.10	An RFID token used for our AC charging station network	171
9.11	Examples of visualisations for the AC station	176
9.12	Simplified class diagram showing REView's vehicle tracking system.	177
9.13	Astra Telematics AT240 vehicle tracking device	178
9.14	Vehicle tracking page screen grab	182
9.15	Examples of charts showing driving data.	183
9.16	Sample leaderboard displaying driving statistics for the trial vehicles	183
9.17	Individual driving statistics	184
9.18	Heat map of the vehicle fleet tracked over a month	185
9.19	Journey logs of a tracked vehicle	186
9.20	Simplified class diagram showing REView's energy generation system.	187
9.21	Graphs generated from solar PV data over a typical week.	188
9.22	Itemised bill generated for a station user for August 2018.	189

9.23 Itemised bill generated for station operators for February 2019.	191
9.24 Network overview bill generated for March 2019.	192
9.25 Smartphone application running REView.	193
9.26 Power drawn by hour of day for EV charging at various location types.	194
9.27 Energy drawn by hour of day	196
9.28 Amount of time spent at a charging station	196
9.29 Average power output of UWA's solar PV system	198
9.30 DC charge frequency regression model	200
9.31 DC energy consumption regression model	200
9.32 AC charge frequency regression model	201
9.33 AC energy consumption regression model	202
9.34 Per-hour energy usage comparison between AC and DC charging stations. .	203
 10.1 Global EV charging inlet adoption	207
10.2 Battery charge rate and State of Charge over time.	211
10.3 Australian charging inlet adoption.	214
10.4 Global DC charging inlet adoption.	215
10.5 Energy delivered for an AC station at each hour of day	217
10.6 Durations of AC charge per station at each hour of day	218
10.7 Energy delivered for an AC station for each day of week	219
10.8 Time taken to AC charge a vehicle for each day of week	219
10.9 Number of chargers per day between each AC station	220
10.10 Energy delivered at each AC station per day	221
10.11 Energy delivered by an AC station versus a DC station	222
10.12 Charging time on an AC station versus a DC station	223
10.13 The average charging duration for a DC and AC charge event.	223
10.14 The daily energy delivery for a DC and AC station.	224
10.15 Number of DC charge events per station per day of week	225
10.16 Number of charges per day for each station	226
10.17 Amount of energy delivered per day for each station	227
10.18 Energy delivered per station per day of week	228
10.19 Energy delivered per station per hour of day	229
10.20 Average charging durations on the DC stations.	229
10.21 Percentage of connector types used at the UWA DC station	231
10.22 Break-even points on required energy delivery	232
 A.1 UWA-REV Autonomous SAE-Electric car.	278

A.2 Hybrid CNN architecture for camera and LiDAR.	281
---	-----

List of tables

2.1	Summary of road datasets presented	25
2.2	Summary of semantic road segmentation algorithms presented	27
3.1	Summary of monocular VO methods reviewed	40
3.3	Summary of stereoscopic VO works reviewed	40
4.1	Functional requirements of software components	57
5.1	LiDAR Characteristics	74
5.3	Semantic segmentation accuracy	84
7.1	Displacement & Error Covariance Comparison	119
7.2	Error and Distance Measurements from Fig. 7.9	120
7.3	F ₁ Scores for Visual Cone Detection	122
7.4	Path Errors	123
7.5	Runtime Performances for Cone Driving	124
8.1	Real and simulated update frequencies of sensors and ROS nodes	136
8.2	Approximate autonomous driving test times.	138
8.4	Real and simulated turning radii for varying steering angles.	143
8.5	Vision based cone detection rates for real and simulated images	147
8.6	Divergence of detection metrics from real images for simulated images.	148
9.1	Efficiency and Theoretical Emissions of Electric Vehicles	158
9.2	The average instantaneous measurements of the solar PV plant	187
9.3	Charging station statistics June 2012 – March 2019 (81 months)	197
10.1	Outputs of various charging stations in southwest WA	209
10.2	Charging style configuration and time for small and large battery packs	214
10.3	Total statistics for the AC stations across the sample period	216

10.4 Comparison of average charging duration and energy consumption	230
10.5 Cost model	234

Nomenclature

Roman Symbols

b Slope

C Cost [Chapter 10]

C Number of charges

D Ground distance

E Energy consumption

F_1 F-measure

h Height

H_n Hermite basis function

i Interest

K Kalman gain

k Time instance [Chapter 7]

M Projected distance

M Profit margin [Chapter 10]

Δx State displacement

N Number set

n Integer set

\mathcal{N} Normal distribution

p	Classified pixel
P_k	Error covariance at instance k [Chapter 7]
P_x	Position at x
p_a^b	Submap pose
R	Break-even energy sales
r	Measurement range [Chapter 4]
r	Product-moment correlation [Chapter 5]
r_a^b	UGV pose
S	Charging station count
s	Paramaterisation
T	Tariff
t	Time period
u	Mean
d	Euclidean distance
v	Velocity
x	x -coordinate (Cartesian system)
y	y -coordinate (Cartesian system)

Greek Symbols

α	Declination
β	Inclination
ε	Error
κ	Curvature
φ	Vehicular orientation
σ	Standard deviation

θ Angular orientation

μ Estimate of x [Chapter 7]

Superscripts

W Euclidean coordinate frame

Subscripts

B Bay lease

C Charging instance

D Depreciation

d Delivery

E Energy

L Lifespan

i Classified class (semantic segmentation)

m Operation/maintenance

r Running

sup Energy supply

b B-spline

t Time

Acronyms / Abbreviations

4G Fourth generation

5G Fifth generation

AC Alternating current

ADAS Advanced driver-assistance systems

ADF Augmented Dickey-Fuller

ANMS Adaptive non-maximal suppression

ANN Artificial neural network

API Application programming interface

ARC Australian Research Council

ASCII American Standard Code for Information Interchange

ASP Application service provider

BRIEF Binary Robust Independent Elementary Features

CAGR Compound annual growth rate

CAN Controller Area Network

CAVI Cooperative and Automated Vehicle Initiative

CBD Central business district

CCD Charge-coupled device

CCS Combined Charging System

CNN Convolutional neural network

CPU Central processing unit

CS Computer science

CSS Cascading Style Sheets

SQL Structured Query Language

C-V2X Cellular vehicle-to-everything

DARPA Defense Advance Research Projects Agency

DC Direct current

DCSI Dense classifier score image

DDS Data distribution system

DIS Dense inverse search

DSO Direct sparse odometry

-
- DWT Discrete wave transform
- EKF Extended Kalman filter
- EV Electric vehicle
- EVSE Electric vehicle supply equipment
- FAST Features from accelerated segment test
- FOV Field of view
- FPS Frames per second
- FREAK Fast retina keypoint
- FSAE Formula SAE
- FTP File Transfer Protocol
- FTS Fleet telematics system
- GCS Ground control system
- GMM Gaussian mixture model
- GNSS Global Navigation Satellite System
- GPS Global Positioning System
- GPU Graphics processing unit
- GST Goods and services tax
- GUI Graphical user interface
- HD High definition
- HIL Hardware-in-the-loop
- HOG Histogram of oriented gradients
- HSDPA High Speed Downlink Packet Access
- HSV Hue, saturation, value
- HTML Hypertext Markup Language

- ICE Internal combustion engine
- ID Identification
- IEC International Electrotechnical Commission
- IMEI International Mobile Equipment Identity
- IMU Inertial measurement unit
- IoT Internet of things
- IoU Intersection over Union
- IoV Internet of vehicles
- IP Internet Protocol
- JSON JavaScript Object Notation
- KLT Kanade-Lucas-Tomasi
- LASV Locally adaptive self-voting
- LiDAR Light Detection and Ranging
- M2M Machine to machine
- MLE Maximum likelihood estimation
- MOG Mixture of Gaussians
- MR-SLAM Multi-robot simultaneous localisation and mapping
- MRS Multi-robot system
- NCSL National Conference of State Legislatures
- NEV New energy vehicle
- NMEA National Marine Electronics Association
- NTP Network Time Protocol
- OBD On-board diagnostics
- OCPP Open Charge Point Protocol

-
- OEM Original equipment manufacturer
- ORB Oriented FAST and rotated BRIEF
- PaaS Platform as a Service
- PA Pixel accuracy
- PID Proportional–integral–derivative (controller)
- PIL Python Imaging Library
- PM Particulate matter
- PNP Perspective-*n*-point
- Protobuf Protocol buffer
- PTAM Parallel tracking and mapping
- PV Photovoltaic
- QoS Quality of service
- RAC Royal Automobile Club of Western Australia
- RAM Random-access memory
- RANSAC Random sample concensus
- RGB-D Red, green, blue and depth
- RCD Residual-current device
- REHL Red Hat Enterprise Linux
- REV (The) Renewable Energy Vehicle (Project)
- RFID Radio-frequency identification
- RGB Red, green and blue
- RMS Root-mean-square
- ROI Region of interest
- ROS Robot Operating System

- RRT Rapidly-exploring random tree
- RTK Real-time kinematic (positioning)
- SAE Society of Automotive Engineers
- SIFT Scale-invariant feature transform
- SIM Subscriber identity module
- SLAM Simultaneous localisation and mapping
- S-MSCKF Stereo multistate constraint Kalman filter
- SOAP Simple Object Access Protocol
- SOLAR Symbiotic online learning of associations and regression
- SSH Secure Shell
- SURF Speeded-up robust features
- SVM Support vector machine
- SVO Semidirect visual odometry
- TCP Transmission Control Protocol
- UGV Unmanned ground vehicle
- UI User interface
- UMTS Universal Mobile Telecommunications System
- UNFCCC United Nations Framework Convention on Climate Change
- UUID Universally unique identifier
- UWA University of Western Australia
- V2C Vehicle-to-cloud
- V2G Vehicle-to-grid
- V2V Vehicle-to-vehicle
- V2I Vehicle-to-infrastructure

V2X Vehicle-to-everything

VIO Visual-inertial odometry

VO Visual odometry

vSLAM Visual simultaneous localisation and mapping

WA Western Australia

WSDL Web Service Description Language

ZNCC Zero-mean normalized cross-correlation

Chapter 1

Introduction

Intelligent vehicles and transportation are attracting tremendous traction in recent years. These interests are not only limited to public perceptions and academia, but participations from major corporate beyond the automotive and computing industries are also greatly contributing toward the developments in this area. The solutions stemmed are actively deployed across various sectors, including transportation, mining, defence, agriculture, telecommunications, energy and trade. As a result, many new vehicles are progressively perceptive and autonomous; they are also becoming more environmentally friendly, often relying on renewable energy sources to mitigate their carbon footprint. It is therefore not uncommon to see them incorporating both elements into the same product. Case in point, many autonomous vehicles are electrically driven, such as the example given in Fig. 1.1. It is often noticeable that autonomous, connected and electric vehicles are inextricably linked.

1.1 Autonomous Driving

The general attainability of precise sensors and high performance compute hardware have driven recent interests in autonomous driving. Autonomous cars (also known as self-driving cars or driverless cars) perform autonomous driving by processing sensor data using an advanced control system that actively calculates the vehicle's navigation trajectory with obstacle avoidance. The sensors are often a combination of LiDARs, radars, sonar, GPS, local odometry, cameras and inertial measurement units (IMUs), which collectively compute through a process known as sensor fusion. Results from sensor fusion therefore enable the vehicle to achieve self-localisation or dead reckoning, along with scene understanding and object tracking. An example of an autonomous car is photographed in Fig. 1.2, showing its roof-mounted cameras and positioning sensors.



Fig. 1.1 An EasyMile EZ10 driverless shuttle at a UWA charging station.

While autonomous cars have been developed as early as the 1980s [1], many would argue that it was not until the DARPA Urban Challenge in 2007 [2] before mainstream research into autonomous driving commenced. Since then, the developments in this area have been growing at a rapid pace. Market research reports published in 2018–2019 [3–5] have estimated the global capitalisation of autonomous vehicles to be valued at US\$55 billion in 2019, with a 35 per cent average compound annual growth rate (CAGR). Further, the Global Automotive & Transportation Research Team at Frost & Sullivan [6] is expecting this figure to raise up to US\$173 billion by 2030, and also stated that shared mobility services such as ride-hailing are to contribute towards a 65 per cent share.

Legislations pertaining to autonomous driving is also increasing in response to its growth. As of May 2019, according to the National Conference of State Legislatures (NCSL), 29 states in the United States have enacted autonomous vehicle legislations [7]. The NCSL has also set up a publicly available Autonomous Vehicle State Bill Tracking Database that is easily searchable to cover various topics from commercialisation to vehicle testing [8]. In Australia, the National Transport Commission (NTC) has been tasked by the Australian Government to draft legislations relating to autonomous vehicles at the federal level [9], including vehicle standards and safety concerns. This is performed while collaborating with various state governments, including Western Australia's Department of Transport [10] to ensure nationwide consistency.



Fig. 1.2 An autonomous car with its sensors visibly mounted on its roof.

Safety is often a salient aspect of any successful development or legislation of autonomous vehicles. This landscape intends to minimise the human factor in driving, noting that human errors caused 94 per cent of all vehicle accidents in the United States [11], which would imply that an ideal autonomous vehicle penetration will reduce accidents by up to 90 per cent [12]. Similarly, 51 per cent of road fatalities in Australia are caused by the driver in 2018 [13]. In order to encourage the penetration and public perception towards autonomous vehicles, MIT's Technology Review has noted the lack of an industry standard for the safety of autonomous vehicles, and have published a brief report relating to the safety regulations [14].

With the economics, legislation and safety setting the development baseline, autonomous driving applications have since evolved from advanced driver-assistance systems (ADAS), where it initially incorporated features including adaptive cruise control, automated parking, blind spot detection, lane departure warning, automatic lane centring and collision avoidance. Driving automation often consolidates these features with active navigation and control, minimising the need for driver intervention. To this end, SAE International has classified driving automation into five different levels under its J3016 “Levels of Driving Automation” standard, ranging from Level 0 (manual driving) to Level 5 (fully autonomous driving) [15]. The official SAE J2016 graphic is given as Fig. 1.3. An increase in driving automation level would typically require greater computation complexity, often using more sensors than its preceding level. Automotive manufacturers have begun the inclusion of Level 3 automation features in production vehicles since 2018, most notably with Tesla’s Autopilot feature.

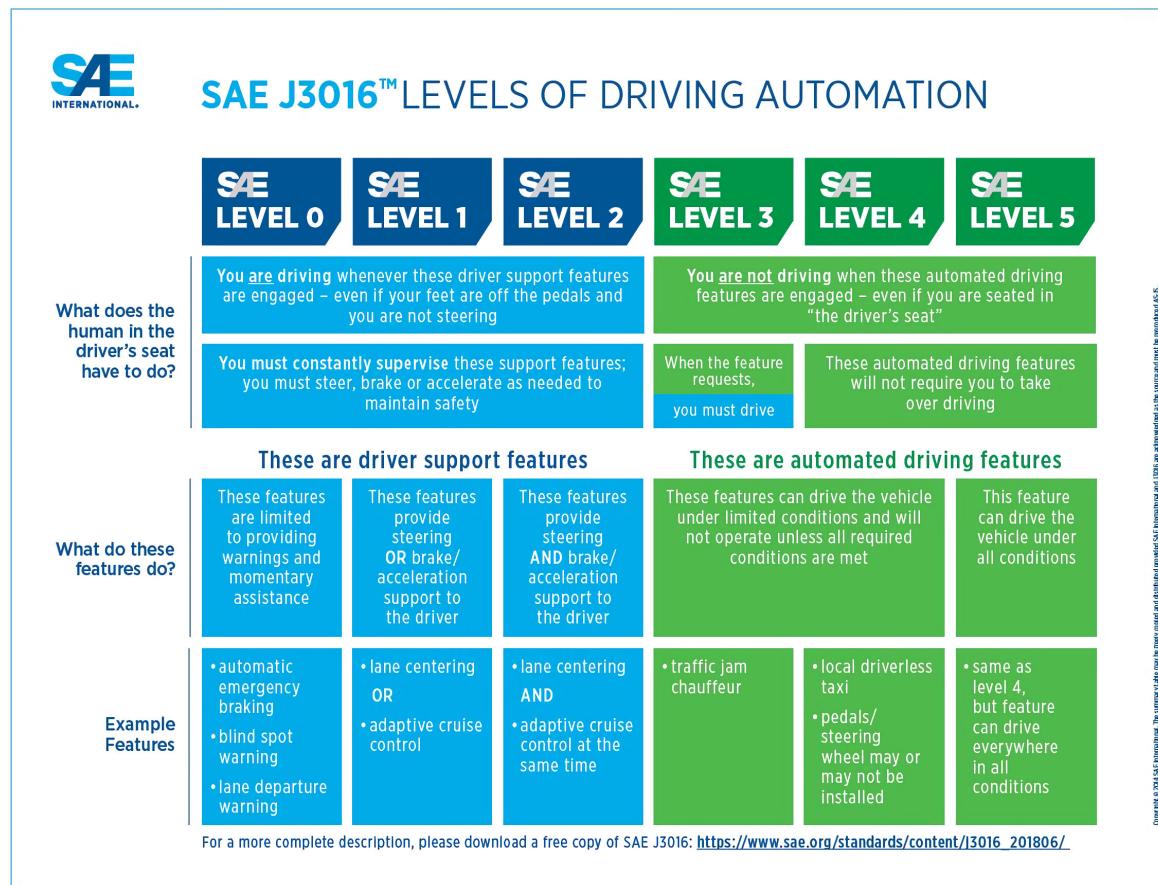


Fig. 1.3 The official graphic for the SAE J3016 standard as of January 2019 [16].

Trials for Level 4 and 5 automation are being conducted by technological corporates such as Google (Waymo) and Uber. It is currently less favoured by production vehicles as the addition of high accuracy sensors would render the ownership cost prohibitive. In the case of positioning, conventional GPS devices have a reporting accuracy of approximately 10 metres, which is inadequate for autonomous navigation; many applications instead use differential GPS or real-time kinematic (RTK) to obtain accurate positioning, incurring higher implementation costs. On the perception front, vehicles often use LiDARs or radars to survey their immediate environment, thereby enabling them to detect or classify objects, and perform obstacle avoidance when necessary. LiDARs are typically preferred over radars in for object tracking and mapping, as it provides distance measurements accuracies in the order of millimetres. They are also capable of producing high definition maps, enabling precise localisation for the vehicles. These vehicles often install multiple LiDARs on their chassis to achieve a 360° perception of its surroundings, often in addition to having dedicated LiDARs for mapping. This further increases the implementation costs, especially considering that individual LiDARs often cost tens of thousands of dollars.

Due to the lower cost of cameras compared to LiDARs, newer applications are starting to favour using the camera as the vehicle’s main perceptive sensor. These applications use computer vision methods to achieve localisation and object classification, often through a single input. However, computer vision algorithms are often more complex, requiring greater computation and memory footprints. This is particularly true when compared against LiDAR-based methods, as they only output a series of measurements, as opposed to a series of complete images that a camera outputs. Nevertheless, with the arrival of high-performance parallel computers, computer vision methods are more likely to better utilise these hardware platforms to achieve more desirable outcomes. These methods have been demonstrated to deliver results pertaining to accurate localisation, mapping and scene understanding; ideally replacing the need for LiDARs, radars, IMUs and local odometry. Using the cameras offer a more cognitive approach to autonomous driving, whereby it mimics a human’s visual perception of the world while driving. Tesla has always maintained a critical position towards LiDARs and favouring the cameras due to its cost and due impracticality, with its CEO Elon Musk claiming that “Anyone relying on LiDAR is doomed”, during his keynote address at the 2019 Tesla Autonomy Investor Day [17].

This thesis describes works that contribute towards using the camera as the primary sensor for autonomous driving, viz. visual autonomous driving. By evaluating these algorithms on the testbed shown in Fig. 1.4, it was found that algorithms relating to environmental perception and localisation can be substituted with computer vision methods. For instance, visual odometry is used in place of wheel odometry and inertial measurements; object classification, detection and tracking be done using the camera in place of LiDARs. Computer vision methods are therefore more versatile as multiple algorithms are able to leverage on a single data source. By doing so, these sensors can then supplement computer vision measurements as an alternative to improve upon classification or measurement accuracies.

1.2 Electromobility

The increased awareness of climate change at the turn of the century is contributing to the rise in sustainable and renewable energy sources worldwide. Heightened levels of greenhouse gas emissions have prompted international treaties, most notably the United Nations Framework Convention on Climate Change (UNFCCC). The UNFCCC Paris Agreement saw 195 ratifications to tackle issues relating to global warming, with a focus to reduce greenhouse gas emissions, and to increase the share of renewable energy and energy efficiency, limiting warming to under 1.5–2°C [18]. This agreement came into force on 4 November 2016. Carbon dioxide (CO₂) remains by far the largest contributor to greenhouse gas emissions (82



Fig. 1.4 The REV Project's autonomous Formula SAE Electric test vehicle.

per cent) [19]. In part, transport is responsible for 23 per cent of global emissions, and is projected to increase to 50 per cent by 2050; car emissions constitute half of this figure [20]. In Australia, transport remains the second largest source of greenhouse gas in the country, emitting 102 million tonnes (18 per cent) of CO₂ in 2018, and is projected to reach 111 million tonnes by 2030 at its current rate [21].

Noting that a vehicle's CO₂ emissions increases with its fuel consumption and type, the automotive industry is making efforts to reduce the carbon footprint of production vehicles with the introduction of green vehicles that run on alternative fuels, including electricity. Electromobility (or e-mobility) is a portmanteau of electric and mobility that is often used to describe electric driving in light of its renaissance that began in the late 2000s. Since the late 2010s, it is mainly used to describe electric vehicles (EVs), particularly electric cars as a relation to current motoring trends.

Electromobility was first conceptualised in the 1900s, back when climate change was unlikely to be a concern. The rationale to produce electric cars back then was to mitigate rising fossil fuel prices while being less noisy. However, it was quickly phased out of favour due to subsequent advancements with the internal combustion engine (ICE) [22]. Still, the availability and know-hows in electromobility continued to persist and improve over the years with overhead line-based transportation, and components such as motors and controllers have become faster and more efficient. Throughout the century, several attempts have been made to reintroduce electric driving to the market, more recently with the General Motors EV1 [23], but inadequate battery technologies and slow charging speeds have restricted their market penetration. These cars often use lead- and nickel-based batteries which are often heavy and have lower energy densities that are insufficient to sustain a suitable driving range.

The electromobility resurgence in the late 2000s was catalysed by climate change awareness and government incentives. This began with the introduction of hybrid electric-petroleum vehicles as a compromise between low tailpipe emissions and a limited electric range, as the batteries can be charged off its engine, ceding its reliance on charging stations. With the push towards zero tailpipe emissions by government lobbyists with additional incentives, automotive manufacturers have begun producing plug-in hybrid and battery EVs that mainly run on electric motors. Around the same time, the proliferation of lithium-ion batteries in personal electronic devices has benefited from improved affordability and energy density, to which these are recently adopted by EVs. Using lithium-ion cells introduces longer ranges and high speed charging to the vehicle, which is in line with current electromobility trends. Another energy storage that is gaining attention with EVs is the hydrogen fuel cell, which uses a redox reaction to generate electricity. These vehicles do not need to be charged, but rather rely on hydrogen as fuel, which requires them to be filled up at hydrogen filling stations. This requires hydrogen to be processed (often through electrolysis), transported and stored multiple times throughout its production chain, constituting to high energy usages even before it can be used as vehicle fuel, in addition to safety concerns during transport as hydrogen is highly flammable. This is in contrast to rechargeable batteries that can easily be charged off the grid.

The charging of EVs can occur at a standard electrical outlet, or at EV charging stations (see Fig. 1.5) which are capable of delivering faster charges, and can be installed in public or private car parks or garages. Charging station deployments are actively effectuated, often by local public authorities and corporate enterprises, especially in developed countries. PlugShare [24] is a website that maps charging station locations through crowdsourcing, and currently tracks them in more than 112,000 locations worldwide with at least 170,000 outlets. Many of these stations belong to a charging station network, which is a collective system of charging stations owned or managed by governments, automotive companies or charging station manufacturers; a notable example is the Tesla Supercharger [25] network. Many networks are capable of connecting to the Internet, which enables usage monitoring and tracking for station users and administrators. Connected charging infrastructures such as these are often also capable of facilitating smart charging or vehicle-to-grid (V2G) systems.

Regulations and incentives have been drafted across various countries either in preparations or in attempts to stimulate EV penetration due to increasing benefits beyond its carbon footprint. These benefits include the running cost of the vehicle, as [26] has calculated that an average EV (18 kWh/100 km) is up to five times cheaper to run per kilometre than a typical ICE vehicle (11.1 l/100 km), and are up to 90 per cent more energy efficient. Noting the higher initial cost of EV ownership, many countries have incentivised EV uptakes



(a) AC Charging Station



(b) DC Charging Station

Fig. 1.5 UWA's AC and DC charging stations.

across varying degrees [27]. Compared to successful initiatives such as in Norway [28], the consumer acceptance of EVs is still low in Australia, and researches to encourage local EV uptakes have been limited [29]. Notwithstanding, the Australian Senate has established a Select Committee on Electric Vehicles to investigate this issue, resulting in a table of a report [30]. The notable recommendations presented in this report include a development of national strategy for EVs and charging infrastructures, and setting up a national EV target. Researches in these areas will likely expedite the implementations of the recommendations.

This thesis presents works that address this area through a quantitative analysis of EV charging behaviours using data that was collected from charging stations around Perth. Comparisons are drawn across different charging station types, taking into account a variety of usage scenarios to better visualise the current EV landscape, where it can be used to supplement policy roadmaps to encourage uptake.

1.3 Connected Mobility

With the availability of low-cost GPS tracking, fleet operators have often relied on vehicle tracking systems to remotely manage and monitor fleets of vehicles. Modern tracking devices are capable of Internet connectivity to transmit telemetry data to a centralised remote server. In addition to location information, this data can include diagnostics from various sensors on the vehicle. In the case of EVs, this can include battery and charging information.

Further advancements in vehicular communications have incited this to evolve as part of the efforts in vehicle-to-everything (V2X) communications. V2X covers vehicular communication across several aspects including but not limited to V2I (vehicle-to-infrastructure), V2V (vehicle-to-vehicle), V2C (vehicle-to-cloud) and V2G (vehicle-to-grid) [31]. These technologies often communicate through a wired or wireless network over a machine to machine (M2M) channel. In the case of wireless connectivity, mobile networks such as 4G are often favoured due to its high transmission speeds, with the incoming 5G standard likely being favoured upon mass deployment, as optimisations are present to facilitate this application. Using mobile networks for V2X applications is often referred to as cellular V2X (C-V2X) [32].

With the advent of cloud and edge computing and the Internet of things (IoT), applications pertaining to V2C communications are becoming increasingly prominent, which include the works described in this thesis. V2C has specifically evolved from fleet management systems whereby the addition of a cloud infrastructure presents the application with intelligent control and monitoring [33]. In the case of an EV ecosystem, a V2C system is capable of consolidating data relating to the EV, charging infrastructures, user behaviours and other stakeholders to present a unified framework for the entire ecosystem. This establishes part of the foundation that leads to intelligent transportation in a smart city, where data from the driving ecosystem is mutually shared for traffic and grid optimisation with low latency connectivity [34, 35], setting the foundation for the Internet of vehicles (IoV).

These communication technologies are easily incorporated into intelligent transportation systems (ITS) to result in smart traffic management and planning; autonomous vehicles will be able to drive cooperatively using a collective perception similar to multi-agent or swarm robotics system. Automotive manufacturers have begun producing vehicles with limited connectivity, but market researches have predicted this market to expand by 45 per cent by 2020 with a 19 per cent CAGR. These implications have not gone unnoticed by governments. In Australia, the governments of Western Australia [36] and New South Wales [37] have studied produced reports relating to connected vehicles, and the Queensland Government's Cooperative and Automated Vehicle Initiative (CAVI) [38] has been established to devise policies pertaining to this matter.

Part of the work described in this thesis intends to establish some preliminary research into connected vehicles in Western Australia. It describes a cloud platform that aggregates data from edge computing that is delivered through a network of smart EV charging stations and an EV fleet. This performs according to a V2C and infrastructure-to-cloud communications system, thereby facilitating data management and reporting to deliver results relating to diagnostics, monitoring and usage forecasts. It is also configured to be extensible to account

for the exponential growth in vehicular and infrastructure data, serving as a pragmatic entry into forthcoming big data researches.

1.4 Contributions

The series of works presented in this thesis aims to formulate pragmatic solutions pertaining to the using the camera as the main sensor for autonomous driving, and interpreting data from EVs and charging infrastructures in a meaningful way. It is multidisciplinary whereby cohesion is ensured under the context of designing system frameworks autonomous electric vehicle applications.

To this end, the main contributions of this thesis are summarised as follows:

- A literature survey pertaining to visual road recognition with specific emphasises on the methods for autonomous driving applications. Learning methods are presented against conventional methods; recent works relating to academia and the industry are discussed. [Chapter 2]
- A literature survey pertaining to visual odometry with specific emphasises on autonomous driving applications. These are categorised according to the types of camera used in relation to the current state of research. [Chapter 3]
- An incorporation of visual odometry and semantic segmentation into a multi-robot system. This introduces visual navigation onto an existing system to improve odometric accuracies, and to enable scene understanding for dynamic object recognition. [Chapter 4]
- An implementation of semantic segmentation on a physical LiDAR-based autonomous driving testbed. This proposed method uses a low-cost monocular camera to segment road regions and lane markings for road centring. [Chapter 5]
- An autonomous driving software framework that is modularly unified to interface sensors with control modules independently. This framework uses protocol buffers to streamline module interoperability to provide an optimised compute performance. [Chapter 6]
- A hybrid extension to the aforementioned software framework using Robot Operating System (ROS). Algorithmic additions to path planning and visual navigation are included, along with sensor interfaces and safety functionalities. [Chapter 7]

- A hardware-in-the-loop (HIL) simulation system for autonomous driving without real-time constraints. The compute hardware is identical to that used on the autonomous driving testbed using the same ROS integration. [Chapter 8]
- A web-based software framework for electromobility telematics. It aggregates and curates data from connected vehicles, charging infrastructures and energy sources, interpreting it for meaningful real-time monitoring and visualisation. [Chapter 9]
- An analysis of EV charging behaviours on charging stations in Western Australia. Comparisons are drawn across the types and locations of charging stations for their adoption rate, and cost model presented subsequently. [Chapter 10]

1.5 Thesis Outline

This thesis comprises of 11 chapters, wherein two chapters present on background reviews, five on autonomous driving frameworks or methods and two on electromobility telematics. The 10 chapters that are subsequent to this introductory chapter is structured as follows:

Chapter 2 presents a survey into the current state of research on methods for computer vision-based road recognition. The backgrounds into the methods are first presented categorically according to conventional (non-learning) and machine learning methods, followed by the implementations of these methods. Conventional methods are presented structurally, following common implementations including horizon and vanishing point detection, region of interest isolation, image classification and model fitting; machine learning methods relate to support vector machines and deep learning approaches, covering popular datasets and image segmentation algorithms. These methods are further reviewed for their implementations in relation to autonomous driving. This is presented first as commercial implementations, covering works from corporates and startups, before presenting on recent academic works with practical implementations.

Chapter 3 presents a review on visual odometry methods for autonomous driving across three approaches — monocular, stereoscopic and visual-inertial. Related applications are discussed for each approach, focusing on works with practical implementations. This is followed by tables that summarise the methods and their presented applications with any applicable datasets. A discussion is drawn to analyse the practicality of the works presented, emphasising on their viability for autonomous driving applications.

Through this review it was known that many works truncate upon experimental validations on datasets, and never proceeded with a tangible implementation, leading to a scarcity in their implementations for autonomous vehicles. This chapter concludes by drawing the necessity of such implementations, as the dynamism of real-world environments must be accounted for.

Chapter 4 describes an implementation of visual odometry and semantic segmentation onto a multi-robot system. The proposal of this implementation acts as a preliminary testbed for the visual navigation algorithms to test their application feasibility before they are ported onto an actual road vehicle. In addition, the incorporation of these algorithms intends to improve upon the existing multi-robot system's localisation accuracy, as well as supplementing navigation with scene understanding. Navigation on the system is performed in a decentralised manner, such that navigational algorithms run independently on each robot without relying on an external or central computer. Evaluations on the visual navigation algorithms have ascertained the feasibility of their implementations in solving problems relating to odometry and object classification while being resilient against environmental dynamics.

Chapter 5 focuses on the application of a semantic segmentation method onto an autonomous driving testbed. The existing testbed is equipped with a LiDAR for object detection, and the addition of visual navigation intends to supersede that to achieve scene understanding and object perceptibility. A low-cost USB camera is mounted onto the vehicle's frame, where it and the other sensors are physically calibrated for camera-LiDAR distance measurements. Semantic segmentation is then applied to the camera recordings and its pixel accuracy is subsequently measured. Experimental results have shown that segmentation is adequate for road markings and lane detection on Perth roads.

Chapter 6 explores the first iteration for an improved software framework for the autonomous driving testbed. The original framework heavily relied on a central Control module which required all sensors and submodules to run. The proposed framework is more efficient whereby it is programmed using a C++ interface across all modules. Module interoperability is ensured using protocol buffers, which separate them into independent classes. Experimental results have validated the efficiency of the software framework, which is given in outputs relating to localisation, odometry, path planning, control and semantic segmentation.

Chapter 7 proposes a hybridised enhancement to the C++-based software framework as a high-level control system. This new framework is based on ROS, and modularly combines sensor data and navigation processing for autonomous driving, while simultaneously ensures vehicle safety and provides data visualisation. It is capable of navigating along with a set of predefined waypoints, or along a cone-delimited path. Visual navigation is once again presented for road and lane detection using semantic segmentation, visual odometry and cone tracking. A HIL simulator is also presented to introduce a parallel development platform using identical compute hardware. Experiments were conducted for sensor fusion, waypoint driving, cone driving and the simulator, where results have collectively demonstrated the system's robustness and adequacy for practical implementations.

Chapter 8 expands on the HIL simulation system described to elaborate on its features. Using a HIL system enables algorithmic prototyping to be rapidly deployed while reducing such risks when compared to a physical system. This is a CARLA-based simulation system at the front-end, whereas autonomous driving routines are performed using identical ROS-based compute hardware across real-world and simulation testbeds to illustrate realistic constraints in relation to its computation footprint. Comparisons were made between the simulation and the physical system, with articulations on the cone detection (LiDAR and vision-based) and path planning algorithms. Evaluations were drawn to benchmark these algorithms, in addition to vehicle dynamics and computation requirements, where it was verified that the tests conducted are transferable between physical and simulation systems.

Chapter 9 introduces the electromobility research in this thesis by detailing the software framework used to collect and process telemetry data from various EVs and their infrastructures. A centralised cloud server is developed for this telematics platform which EVs, charging infrastructures and power sources push data to. The application layer is entirely web-based and is capable of pulling data in real-time for user monitoring and visualisations. This is presented for charging stations, EVs fleet tracking and energy generation, wherein for each section, the back-ends and algorithms are elaborated to result in visualisations. Gamification is presented for vehicle tracking to encourage economical driving, and monetisation options are presented as bills to inform users of energy usage in charging stations. The results generated from this telematics platform was summarised as usages pertaining to charging infrastructures and energy generation, as well as heat maps for EV tracking. A forecast of the charging infrastructures' usage is also presented and analysed as a precursor to predicting the

local EV penetration. All platform modules were written in a modular approach to encourage future improvements and expansions.

Chapter 10 investigates EV charging behaviours by analysing data on the telemetry platform. This begins with a background on various types of EV charging worldwide, followed by charging speeds and cycles. Data is collected from charging stations managed by The REV Project, with comparisons drawn using data from the RAC Electric Highway in Western Australia. Data is analysed through various time series analysis, which investigated station usage frequencies and energy consumption over a predefined period; samples are given in hours-of-day and days-of-week to study usage patterns. A cost model is drawn to estimate the costs for running and maintaining different types of charging stations, and external scenarios such as parking bay rentals are also considered. These analyses are validated using a similar study, and it was concluded that slower charging stations are becoming obsolete and are shifting towards personal installations, whereas public installations will prefer fast-charging stations.

Chapter 11 concludes this thesis with a summary of the contributions made, along with suggestions to outline future research directions.

Chapter 2

A Methodological Review of Visual Road Recognition Procedures for Autonomous Driving Applications

The current research interest in autonomous driving is growing at a rapid pace, attracting great investments from both the academic and corporate sectors. In order for vehicles to be fully autonomous, it is imperative that the driver assistance system is adapt in road and lane keeping. In this paper, we present a methodological review of techniques with a focus on visual road detection and recognition. We adopt a pragmatic outlook in presenting this review, whereby the procedures of road recognition is emphasised with respect to its practical implementations. The contribution of this review hence covers the topic in two parts — the first part describes the methodological approach to conventional road detection, which covers the algorithms and approaches involved to classify and segregate roads from non-road regions; and the other part focuses on recent state-of-the-art machine learning techniques that are applied to visual road recognition, with an emphasis on methods that incorporate convolutional neural networks and semantic segmentation. A subsequent overview of recent implementations in the commercial sector is also presented, along with some recent research works pertaining to road detections.

2.1 Introduction

The field of autonomous driving is attracting much attention lately, ever since its feasibility was established in the 2007 DARPA Urban Challenge [2]. These days, technological and automotive corporates are expediting the announcements of autonomous vehicles to the

consumer market alongside electric vehicles, which are also becoming imminently available. From a research standpoint, this area is also well-documented in the literature. Conventional systems often rely on radar and subsequently, LiDAR to detect road kerbs and edges, but with the advancement of computer vision, cameras are quickly replacing these sensors as the preferred sensor to detect and recognise roads. Cameras also benefit from being versatile and low-cost, in addition to its ubiquity which enables the deployment of visual autonomous driving on a larger scale.

Using cameras for road detection and recognition however introduces challenges whereby it is heavily reliant on the robustness of the image processing algorithms to accurately recognise road regions, often in real-time. This is unlike LiDAR and/or radar-based approaches that usually relies on the processing of the sensor's measurement values to classify roads from non-road regions around the vehicle. A robust algorithm for road recognition should account for the dynamic variations of road types, conditions and illumination, as well as seasonal and weather changes pertaining to the road scene, while being able to accurately perform road classification.

A comprehensive background study and review on this topic was presented in 2013 [39], whereby the authors have discussed the problems faced by visual road detection and recognition, particularly in the two categories of common roads — structured (with lane markings) and unstructured (without lane markings). Like most visual computing problems, visual road detection is also susceptible to variations of lighting, along with weather and seasonal changes. Solutions to visual road detection are similar to most visual computing approaches, whereby a captured image frame will first undergo preprocessing where to reduce noise and other imperfections. The image will subsequently have its features detected and extracted using algorithms such as Scale-Invariant Feature Transform (SIFT) [40] or Speeded-Up Robust Features (SURF) [41], and then distinguishing these features as road areas. These extracted features will correspond to areas on the image where road and non-road regions are distinguished. This is essentially a sequential process of image preprocessing, feature extraction and model fitting [42]. Aspects that are unique to visual road detection are the prevalence of the horizon and the vanishing point of the road. The horizon is the boundary of the frame where the sky and land meets; and the vanishing point is the point where the road converges to. These aspects along with the road establish the region of interest (ROI) where visual processing can be carried out.

With the advent of deep learning, convolutional neural networks (CNN) are increasingly being incorporated into road detection algorithms to train and classify roads with improved accuracy [43–45]. Using CNN introduces high processing requirements and increases the complexity of the algorithm. Such an algorithm is usually trained offline at a dedicated

server or workstation to obtain a dataset related to the driving environment. Datasets for road detection includes KITTI [46] and Daimler [47]. Conversely, there are proposals of fast and simple algorithms that perform road recognition without the need of another computer, and does not require training [48, 49]. This further classifies visual road recognition into supervised and unsupervised algorithms, indicating the presence or absence thereof a training classifier within the algorithm. This being said, the application of CNNs onto an image processing problem means that computation using the graphics processing unit (GPU) is rapidly gaining in popularity. The high parallelism of GPU architectures is especially suitable for the parallel nature of visual and deep learning applications such as road detection. The Nvidia DRIVE [50] solution is a testament to this, whereby an industrial GPU maker is currently developing GPU solutions to autonomous driving that is centred around deep learning and computer vision. Additionally, other specialised hardware such as Mobileye's Automated Driver Assistance System (ADAS) is the core technology utilised by many of their 27 automotive manufacturer partners across 313 models for their autonomous driving feature [51].

This paper is organised as follows. Section 2.2 presents works covering the procedural implementation of road detection using standard classification methods, including the detection of vanishing points, region of interests, image classification and model fitting. Methods that incorporates machine learning, particularly on CNN and its methods, are presented in Section 2.3. Section 2.4 presents commercial implementations with regards to products and courses with road recognition, and Section 2.5 presents the current trends and works in recent years for road detection before this the concluding remark is presented in Section 2.6.

2.2 Conventional Methods

Road recognition for autonomous driving generally follows the methods described in Sections 2.2.1 to 2.2.5 in a chronological order. More specifically, many implementations start with a preprocessing stage to filter image noise and other inconsistencies, followed by a horizon detection algorithm to crop the horizon so that the image is processed only at the road portion that is below the horizon line. Researchers may then use vanishing point detection to orient or localise the vehicle on the road. Regions of interests may be used to isolate the process on road or lane marking edges for image classification. During image classification, methods include using a combination of either edge detection, colour histograms, textural comparison, machine learning or neural networks; which is then typically classified in binary (non-road/road regions) through methods such as Gaussian filtering or confidence voting.

Finally, the navigational boundaries are marked on the edges with lines and subsequently plotting the centre path for the vehicle to drive on.

2.2.1 Horizon Detection

Horizon detection algorithms are typically used for road detection to crop the image frame, thereby reducing overall computational requirements. Horizon detection is generally applied onto a preprocessed image to calculate the boundary of the skyline for an image frame. The area above the horizon will be isolated and ignored for the rest of the computation. A fast horizon detection algorithm is favourable to present minimal processing and memory footprint with reference to the overall visual road recognition solution. While the fastest and simplest approach may be to fix the horizon at a constant pixel location according to the camera's orientation, this assumes that the vehicle is always traversing on a perfectly even terrain with no variations of pitch nor incline, which is generally unachievable under normal driving circumstances. Horizon detection is either edge-less or edge-based [52]. Edge-less approaches use edge classification whereby the horizon is detected by removing non-horizon edges through the refinement of the edge map; in an edge-less approach, each pixel location is classified according to their probability of it being on the horizon. An edge-based approach was proposed by Lie et al. [53], where a multi-stage graph was generated from an edge map using a dynamic programming algorithm. An edge-less approach was proposed by Ahmad et al. [54] which instead uses a classification map to find the horizon line that incorporates machine learning and dynamic programming. According to the authors, using an edge-less approach will not require the assumption of the horizon line being close to the top of the image frame. An example of the process of using an edge-less algorithm is illustrated in Fig. 2.1. Ahmad et al. went on to propose a method that fuses edge-based and edge-less approaches [52], outperforming both edge-based and edge-less approaches. On the deep learning front, Verbikas and Whitehead [55] incorporated CNN into horizon detection, which outperforms other classifiers in their experiments in accuracy. CNN was applied to train classifiers to recognise sky and ground features with spatial feature extractors.

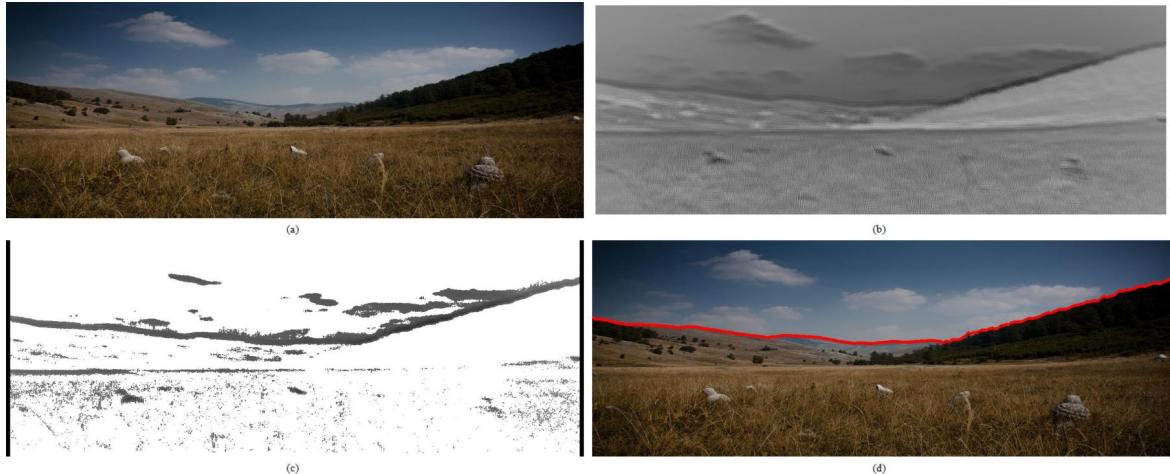


Fig. 2.1 An edge-less horizon detection algorithm generates a dense classifier score image (DCSI) (b) from a query image (a) using trained classifiers, where a threshold is then applied (c) before it plots the horizon line (d). Reprinted with permission from [54]

2.2.2 Vanishing Point Detection

This is typically used in tandem or as an alternative to horizon detection to localise the vehicle with respect to the image frame. The vanishing point is a point on an image where a pair or more parallel lines in 3D space converges to. According to Rother [56], the detection of vanishing point consists of an accumulation step and a search step. The accumulation step clusters line segments that share a common vanishing point, and the search step searches for dominant line clusters. Rother noted that the random sample consensus (RANSAC) method could be used to speed up vanishing point detection, to which Bazin and Pollefeys [57] proposed an approach that uses only three lines to achieve this. They proposed this approach for a three degrees of freedom (3DoF) robotic manoeuvrability system similar to a ground vehicle. This effectively enables the system to estimate its rotation based on its captured visual lines and vanishing point. Kong, Audibert and Ponce [58] described an approach to road detection that centres around vanishing point detection, and their method is as illustrated in Fig. 2.2. They proposed the Locally Adaptive Self-Voting (LASV) algorithm that estimates the vanishing point based on a confidence model in a local region for texture orientation estimation. While it is more accurate than conventional voting approaches, Zhu et al. [59] noted that this approach does not perform well in suburban environments with dense roadside vegetation and fixtures. They subsequently utilised a colour histogram method to compare the captured image against an *a priori* model to obtain the vanishing point. Still, line voting

remains a popular method to achieve vanishing point detection and it is also used by Zhu et al. and other recent works [60–64].

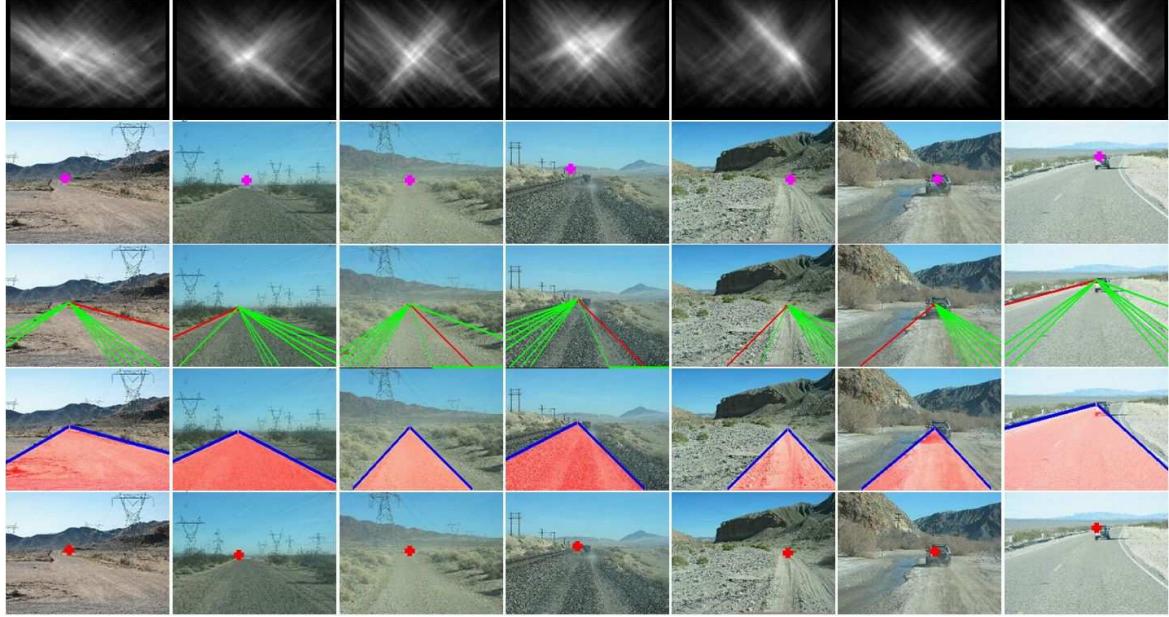


Fig. 2.2 Vanishing point estimation of seven desert road images showing the LASV algorithm. Reprinted with permission from [58].

2.2.3 Region of Interest Isolation

ROI isolation methods are used as a popular approach to recognise road segments from non-road segments. An ROI is usually identified and defined in frames before these image segments are classified. Instead of needing to process the entire frame, using an ROI isolates image processing to a frame's specific region to further reduce computing requirements. This may be used in tandem with horizon detection where certain regions below the horizon line are designated as the ROI. For road recognition, the ROI is generally a definitive region that encompasses both road and non-road regions or lane markings, as classification can then be drawn from processing that ROI. While conventional ROIs are usually fixed at a predetermined location on a frame, this assumes that the road boundary will always be on the same frame location [65, 66]. To circumvent this, adaptive ROI algorithms were proposed as a more robust solution that adjusts to illumination changes [67] or the location of vanishing points [68].

2.2.4 Image Classification

Image classification is used in this context to classify an image into road and non-road areas using a binary classifier [49]. Roads are recognised through a combination of the road lane markings and road boundary. Lane markers are usually painted in high contrast from the road surface to be conspicuous to the drivers, and visual processing also benefits from this whereby good edge detection results can be obtained more easily. Conversely, variations in road lane appearances such as colours, lines and condition from wear and tear may pose a challenge for lane detection algorithms [69]. Works that attempt to circumvent these variations include [70, 71]. Some roads, especially non-gravel roads, are unstructured and have no lane markings at all. In these circumstances, lane detection algorithms will not work, and road boundary detection algorithms will be applied. On urban roads, the road boundary is perceived as the region where the asphalt meets an unpaved ground. Works that detect road boundaries may also use kerbs [72] or highway barriers [73]. More robust road boundary detection algorithms aim to work across different road types, including dirt roads and snow roads [58]; or variations to illumination and weather such as night driving and rain [74, 75]. Techniques used for both road lane and boundary detection may vary, but they may also share some similarities, especially with the usage of edge detection algorithms. For instance, stereoscopic sensors can be used to perceive the tangible road boundaries in urban areas, which works as an alternative to radar or LiDAR [76], or in conjunction for added robustness [77]. Non-urban roads commonly share the same plane as non-road areas, so such detection algorithm should be purely appearance-based. An example of such an algorithm was proposed by Cristóforis et al. [49], where they applied a mixture of Gaussians (MOG) model onto an image's ROI on HSV colour space that is converted from RGB. This more commonly known as the Gaussian mixture model (GMM). The GMM is a form a Bayesian classification, which performs decision making using the probability theory based on the maximum likelihood estimation (MLE). As its name suggests, a GMM a combination of several Gaussian distributions and hence the MLE is derived from the weight sum of the Gaussians as the probability density function [78]. In the context of road detection, GMM analyses the colour distribution of the road and estimates the colour model based its similarities with similar model groups. Gaussian models are widely used as a supervised learning approach to image classification. Other works that incorporate a Gaussian model include [79, 80]. Alkhorshid et al. [81] used a histogram obtained from the calculation of frequency distribution of pixel values, and subsequently used candidate training to classify whether or not the ROI is fully, partially or not part of a road region; this is modelled after the AdaBoost classifier [82] that is employed to minimise weighted errors. Other methods of

classification include using textural features [83, 84], which assumes a homogeneous road texture that is compared to non-road textures.

2.2.5 Model Fitting

Once road and non-road regions have been classified in the image, navigational boundaries must be marked (fitted) to prevent the vehicle from veering off course. These boundaries can be marked more easily on marked roads using edge detection algorithms that benefit from the large gradient values. Edge filters such as the Sobel and Canny filters are commonly used [85–87]. Regions of the high gradient can then be plotted according to the filters' results. These plots will result in the road or lane boundary, and they can either be parametric, semi-parametric or non-parametric [42]. Parametric models comprise mostly of straight lines [58]; semi-parametric models comprise of splines [88] and polynomial curves [89, 90]; and non-parametric models comprise of continuous arbitrary lines [49]. Urban roads typically have well-defined lane markings and boundaries, hence these navigational boundaries can be marked with a parametric or semi parametric model, effectively reducing computation complexity. Rural and unpaved roads may require the use of semi or non-polynomial models. Outliers are commonly present with fitting models, therefore it is also common to implement RANSAC for outlier rejection at this stage. Aly [79] used RANSAC to fit lines and splines in his lane detection algorithm, and it is applied after performing a simplified Hough transform for lane line counting. With the road/lane boundaries marked, the vehicle can then be guided to drive at the centre of the road/lane by finding the distance between its left and right boundaries. In addition to road boundaries, model fitting can also be applied for marking the horizon and path for navigation, such as the approach used by Cristóforis et al. [49]. A popular technique of model fitting in road recognition is the Hough transform [91]. The Hough transform is a shape analysis technique typically used to extract shape features from an image. Road detection and recognition works commonly apply Hough line transform on an edge-detected image to detect lane markings and road edges according to the aforementioned line models. This achieves road segmentation, splitting road sections for vehicles to recognise areas such as lanes and non-road areas. The Hough transform uses a voting procedure to fit lines. This means that each point that may correspond to a line section votes for the likelihood that a line section may be from. More votes are cast when more points lie on the same line, and lines with higher votes will be fitted [92]. Fig. 2.3 illustrates an example of lane detection using Hough transform, the blue and red lines mark the left and right edge of the lane respectively, where a driving rule can be established to ensure that the vehicle does not cross these boundaries while driving. Works that employ the Hough transform or a similar voting approach to classify road and non-road regions are [45, 58, 59].

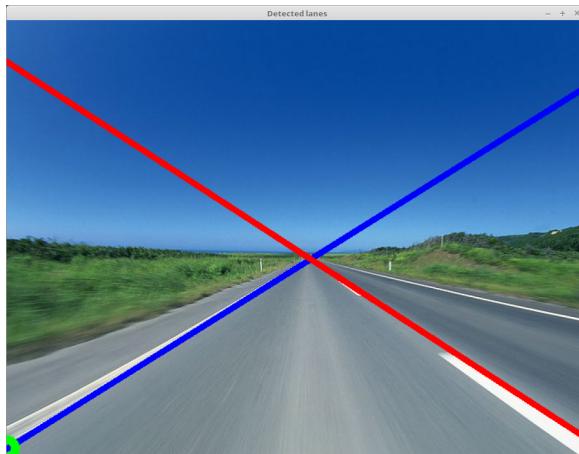


Fig. 2.3 Lane detection with Hough transform performed using OpenCV

2.3 Learning Methods

Many works are implementing learning methods these days for road recognition, where support vector machines (SVMs) [93, 94], neural networks [95] and AdaBoost [96] are among the more commonly implemented approaches. A neural network approach was proposed as early as 2003 by Conrad and Foedisch [97] using Matlab's Neural Network Toolbox. The authors compared this approach to a SVM approach and noted that while SVMs are more accurate, their computation times are long for road classification tasks. It is noted that SVM has historically been a mainstream learning method for road recognition, with newer implementations incorporating dynamic programming to account for the changes in road scenes [98]. Learning methods on road recognition is seeing a rise in popularity also due to the increased demand in autonomous driving, and the KITTI benchmark suite also includes are road and lane detection evaluation benchmark since 2013 [99]. This benchmark categorises road scenes according to a combination of roads types including urban marked, unmarked, multiple marked lanes/roads. There are currently 336 benchmark submissions for the categories to date, using various methods such as SVM [100], CNN [101, 102] and may incorporate other sensors such as LiDAR [103].

Due to the increased availability of parallel computers nowadays, recent works are more commonly implementing convolutional neural networks for road recognition in favour of SVM and custom networks. CNNs are feedforward neural networks with more than one convolution layer. This convolution layer applies a function repeatedly over the output of other functions, which greatly benefit from parallel computation. For road recognition, CNNs are commonly used for object detection and image segmentation, which are used on road scenes to detect and discern areas in the image that encompasses an object, where in addition

to roads, it also segments other elements such as pedestrians, vehicles, vegetation and road signs. This process is commonly known as semantic segmentation. Semantic segmentation classes objects in an image according to its pixels, thereby improving the system's comprehension towards road scenes in addition to road classification over conventional learning methods, allowing for a more holistic autonomous driving system that also incorporates features such as road sign understanding, pedestrian and vehicle detection, and collision avoidance [104–106]. Thoma [107] and Garcia-Garcia et al. [108] published surveys of semantic segmentation, which include a good background study of the underlying approaches of semantic segmentation. While the KITTI benchmark suite has not yet incorporated a semantic segmentation benchmark, it does provide a list of resources of KITTI images with semantic labels. MultiNet [105] is an example that combines semantic segmentation, classification and detection for road scenes that is consolidated from the same encoder to minimise time redundancy. Results were tested and benchmarked on KITTI's road dataset where it is found to be capable of real-time processing.

Semantic segmentation uses neural networks and is hence trained and tested on datasets. The review paper by Garcia-garcia et al. [108] provides a detailed analysis of the datasets used in semantic segmentation, categorising them into 2D, 2.5D and 3D datasets. The KITTI [109], CamVid [110] and Cityscapes [111] datasets are more commonly associated with training and testing semantic segmentation for urban road scenes. The KITTI Vision Benchmark Suite is an actively maintained project by the Karlsruhe Institute of Technology and the Toyota Technological Institute of Chicago. Images were obtained by driving a car around Karlsruhe, Germany, covering a variety of road scenes. This benchmark suite spans across several categories including optical flow, stereo vision, visual odometry, and road/lane detection. The road/lane detection evaluation benchmark consists of 289 and 290 training and test images respectively. Additionally, the CamVid dataset was proposed in 2008 containing 367 training and 233 testing images. Images were obtained from the City of Cambridge, England by driving a car around. This dataset was created for semantic segmentation and each training image pixel is labelled with a different shade of grey corresponding to one of the twelve classes for road scene objects, forming the ground truth. Finally, the Cityscapes dataset was obtained from 50 cities in Germany from a road vehicle across three seasons. This is a 5000-image dataset whereby 2975, 500 and 1525 images are categorised for training, validation and testing respectively. The authors of Cityscapes made comparisons to KITTI and CamVid and noted that semantic labelling can be easily achieved with their smaller datasets, and that the Cityscapes dataset provides a better challenge for new semantic segmentation approaches with its much larger dataset. Table 2.1 summarises the datasets presented here according to its publication data and number of sample images for training and testing.

Table 2.1 Summary of road datasets presented

Dataset	Year	Image Samples	
		Training	Testing
KITTI-Road [109]	2013	289	290
CamVid [110]	2009	367	233
Cityscapes [111]	2016	2975	1525

Examples of works that focuses on semantic segmentation for road scene recognition are SegNet [106], KittiSeg [105] and ENet [112]. SegNet was proposed by Badrinarayanan et al. as a CNN architecture that is often implemented on the Caffe [113] framework. Its architecture uses an encoder-decoder network that is followed by a pixelwise classification layer, where the encoder and decoder networks consist 13 convolutional layers each. The encoder network is the same as the VGG16 [114] network, which performs convolutions to obtain a set of feature maps. The decoder network then upsamples the feature map of each corresponding encoder, producing dense feature maps that are then batch normalised. A soft-max classifier at the output then individually classifies each pixel into one of twelve object classes to form the output image as illustrated in Fig. 2.4. SegNet’s proposal was compared against fully convolutional network decoding technique, also based on the VGG16 network, and DeconvNet [115], which uses fully connected layers. The authors tested SegNet on the CamVid dataset and SegNet’s road scene segmentation results showed that while DeconvNet and SegNet yielded favourable results, SegNet’s computational cost was significantly lesser due to its network being smaller. KittiSeg is the segmentation sequence of MultiNet whereby encoding is performed using the first 13 layers of the VGG16 network like SegNet. The fully connected layers of the VGG architecture is then transformed for decoding, thereby employing a fully connected network architecture. The authors used the KITTI Road Benchmark dataset for training and noted that the network converged quickly with high road segmentation efficiencies, which placed their KittiSeg on top of KITTI’s road leaderboard at the time of its publication. An example of KittiSeg’s input and output on the Cityscapes dataset is as illustrated in Fig. 2.5. Paszke et al. noted that the VGG16 architecture that these works are based on are very large and hence less suitable for embedded and mobile applications, leading to the proposal of ENet [112]. Since real-time semantic segmentation requires a frame rate of at least 10 frames per second (fps), this is difficult to achieve on embedded computers. ENet is a custom-designed neural network

architecture proposed for high computational speed and accuracy that is designed based on ResNets [116]. Performance tests showed that ENet is about 17 times faster than SegNet while running on an Nvidia Jetson TX1 [117] embedded computer, while being significantly more memory efficient. Training and testing benchmarks were performed across the Cityscapes, CamVid and SUN RGB-D [118] datasets with their results compared against SegNet. By measuring the intersection over union (IoU) matrices, ENet was able to outperform SegNet in the Cityscapes dataset, as well as the CamVid dataset in six of its eleven classes. Tremel et al. [119] subsequently proposed a new architecture that improves on the accuracy of ENet, while being implementable on embedded computers for real-time inference. This architecture follows SegNet whereby it uses an encoder-decoder pair. The authors modified a SqueezeNet [120] architecture for its encoder network favouring its low latencies, and a parallel dilated convolution layer [121] as its decoder to retain high computation performance while using fewer parameters. Testing and training were performed on the Cityscapes dataset over the Caffe framework. Results were compared against ENet, outperforming it in its IoU matrices in both class and category, while compromising on slightly lower framerates on the Jetson TX1, but still exceeding the 10 fps requirement for autonomous driving. A summary of the semantic road segmentation algorithms are presented in Table 2.2, listing each algorithm according to its encoder-decoder network, and the datasets used for their experiments.



Fig. 2.4 SegNet's input (left) and output (right) for a typical Western Australian road scene.



Fig. 2.5 KittiSeg’s input (left) and output (right) on the Cityscapes dataset using a Tensorflow [122] implementation. Reprinted with permission from [123]

Table 2.2 Summary of semantic road segmentation algorithms presented

Algorithm	Year	Encoder	Decoder	Dataset
SegNet [106]	2015	VGG16	Custom	CamVid
KittiSeg [105]	2016	VGG16	Fully convoluted	KITTI Road
ENet [112]	2016	bottleneck	bottleneck	CamVid, Cityscapes
Treml et al. [119]	2016	SqueezeNet 1.1	Parallel dilated convolutions	Cityscapes

2.4 Commercial Implementations

Commercial implementations of road recognition in the automotive industry are largely based on the availability of original equipment manufacturers (OEMs) that supply advanced driver-assistance system (ADAS) computers and sensors. Mobileye [51], Nvidia [50], Velodyne [124] and FLIR [125] are few of the OEMs involved in manufacturing autonomous driving systems. Mobileye is reputed for their ADAS system-on-chip (SoC) called EyeQ [126], where in addition to lane keeping, it is capable of supporting sensor fusion, visual computing, path planning etc. towards full (Level 5 [127]) autonomous driving while being power-efficient. Nvidia’s DRIVE PX [50] system uses their graphic processing unit (GPU) architectures to deliver on fast learning performance on mobile vehicles. They recently announced the Drive PX Xavier computer, which is an SoC that integrates a new GPU architecture, an eight-core central processing unit (CPU) and a computer vision accelerator with a 20 Watt requirement [128], making it ideal for real-time road recognition tasks. Nvidia has also published a work describing the mapping camera pixels to steering commands using

an end-to-end approach on a CNN, which is processed by the Drive PX [129]. Velodyne and FLIR are well-known OEMs that manufacture LiDAR and camera systems respectively for autonomous driving. Aside from OEMs, corporates that research into autonomous driving algorithms includes Google [130] and Uber [131], as well as automotive manufacturers such as BMW [132], Volvo [133] and Daimler [134].

Road recognition and detection techniques are also becoming more accessible to the masses. As part of an effort to produce an open-sourced autonomous car, Udacity has introduced its Self-Driving Car Nanodegree Programme, which includes road and lane detection as part of its Term 1 curriculum [135]. Specifically, Project 1: Finding Lane Lines, and Project 4: Advanced Lane-Finding. In Project 1, students utilised OpenCV's functions such as Canny edge detection and Hough transform for road and lane detection. Project 4 expands on this to classify lane boundaries, as well as to provide the vehicle's estimated position on the road and the road's curvature. Binary images of road scenes are perspectively transformed into a birds-eye view, where lane pixels are subsequently detected for a polynomial model fitting. The model fitted lanes can then determine the road's curvature. Fig. 2.6 shows the final output of a Project 4 report, which includes marked lane boundaries with estimations of the lane curvatures and the vehicle's position.



Fig. 2.6 The Self-Driving Car Nanodegree's Term 1, Project 4 output. Reprinted with permission from [136].

comma.ai [137] is a startup company by George Hotz that specialises in providing assisted and autonomous driving systems to the consumer market. Their goal is to achieve full autonomous driving with existing road vehicles with after-market devices. Most of the software that comma.ai creates are open-sourced, which includes its autonomous driving system, openpilot [138]. openpilot performs adaptive cruise control and lane keeping that can be retrofitted to existing cars. comma.ai also includes semantic segmentation for road

scenes [139], where they have experimented with SegNet and ENet, and proposed a solution based on ENet and ReSeg with VGG convolutional layers [140] dubbed Suggestions Network (SugNet) to automatically label ground truths. This is in addition to the recurrent neural network approach that they took with autoencoders to learn a driving simulator as part of their initial research, which generates realistic road image predictions [141].

2.5 Recent Works

Recent research developments in road detection are inclining towards supervised learning and neural networks. For instance, Brust et al. [43] described an approach that uses an image patch that is fed into a CNN for label estimation, dubbed the Convolutional Patch Network. The image patch is used as the spatial prior for this method, which corresponds to a position of an object from a small group of pixels in the image frame. Among the other implementation methods proposed by the authors is a normalised initialisation approach to neural network parameters, thereby circumventing the vanishing gradient problem. For benchmarking, the authors used the KITTI dataset in a birds-eye view. This means that a transformation was done to convert dashboard view (ego view) images into birds-eye view, to which [99] claimed that road detection is more efficient this way. Training weights are therefore chosen according to the pixel sizes after this conversion, as the authors noted that any classification errors that happens near the horizon pixels in ego view will escalate to many more pixels in birds eye view. Experiments performed by the authors for road detection yields a 10% improvement over that of Alvarez et al., which is largely contributed to the addition of spatial priors into the network. Visual road recognition implementations for autonomous driving are sometimes performed on mobile robots due to local legislations and safety concerns on autonomous vehicles [142]. Öfjäll et al. [142] developed a road-following system that incorporates supervised and self-reinforcement learning called symbiotic online learning of associations and regression (SOLAR). The system is initially trained by a human driver with a recording camera for the system to learn the road's appearance, where upon sufficient training, the system will be capable of taking over controls from the driver. The system predicts visual feature vectors of subsequent frames using a Hebbian associate learning procedure [143], allowing the system to perform self-feedbacks for reinforcement learning. The authors implemented SOLAR on a remote controllable robot car in environments that simulate real roads, and subsequently compared its autonomous driving capability and learning time to qHebb [143], along with a CNN approach with the Caffe framework. Results showed that the CNN approach is incapable of running in real time with long learning times. Comparisons with qHebb notes that SOLAR is able to simultaneously improve learning

speeds due to its reinforcement learning. Another work that implements road detection on a mobile robot for testing is the visual road following approach by Krajník et al. [144]. It shares some similarities with Cristóforis et al.'s [49] work whereby it produces a path guide for the autonomous navigation of a mobile robot. This work, however, emphasises on their photometric methods that adjust to the reflectance of captured objects for shadow removal, hence providing an illumination invariant solution for visual road recognition. For testing, the authors implemented two threads in parallel; one for manoeuvring the robot, and the other calculates the robot's orientation with respect to the path boundary. This effectively ensures that the robot navigates at the centre of the path. An image captured by the robot's on-board camera is processed into an intrinsic image — a process that decomposes an image into multiple layers of intrinsic properties. Using intrinsic images enables the algorithm to be illumination invariant. The authors subsequently used the intrinsic images to compute the robot's path through histogram equalisation, which segments the path from the background, thus binary classifying them into path and non-path regions. The robot's orientation for navigation is then calculated from a probability distribution estimation of intrinsic pixels from a histogram based on Shannon entropy [145]. Experiments were conducted offline using datasets and online on a mobile robot, and the results proved that using intrinsic images allows the robot to move autonomously across different illumination conditions.

2.6 Conclusion

This paper presented a review of literature that covers the visual road recognition process according to its associated methods, followed by methods that incorporate machine learning, and a brief review of current commercial implementations. In a typical chronology, methodological approach starts with horizon detection under the assumption that all road regions are below the horizon, which effectively isolates and segregates the sky portion above the horizon from any image processing. Subsequently, detecting vanishing points allows us to find the point of convergence between the road and the horizon, enabling horizon and road segmentation to be performed corresponding to that vanishing point. The road region below the vanishing point can be further segmented for computation efficiency by the introduction of the region of interests, which usually encapsulates the road or its edges where computation can be concatenated. Computational processes for visual road detection generally involves binary image classification, which classifies roads from non-road regions using techniques ranging from Gaussian models to histograms. Recently, many works incorporate CNN for visual classification with improved accuracies. With image classification complete, model fitting is applied to visually distinguish roads from non-road areas. There is also a recent

shift in road recognition approaches using deep learning whereby semantic segmentation is increasingly utilised for road detection, along with other objects in road scenes. From a hardware perspective, visual road recognition is quickly replacing conventional methods such as using LiDAR and radar due to the rapid improvements in cost and availability of image sensors. With the general availability of datasets and libraries such as KITTI and OpenCV, along with open-source deep learning libraries such as Caffe and Tensorflow, visual road recognition is now easily implementable and evaluated even in embedded systems. Additionally, recent approaches in visual road recognition is steadily addressing the research challenges presented in this area, which encompass those that are generally found in visual computing, such as illumination invariance and camera distortions. Therefore, a robust visual road recognition system should provide high accuracies while maintaining real-time computation capabilities that is able to compensate for the dynamic changes in road scenes at any time.

Acknowledgment

The authors would like to thank Mr Andrea Palazzi, Mr Thomas Anthony, Mr Touqeer Ahmad and Prof Hui Kong for the granting of permission to use their figures in this paper.

Chapter 3

A Review of Visual Odometry Methods and Its Applications for Autonomous Driving

The research into autonomous driving applications has observed an increase in computer vision-based approaches in recent years. In attempts to develop exclusive vision-based systems, visual odometry is often considered as a key element to achieve motion estimation and self-localisation, in place of wheel odometry or inertial measurements. This paper presents a recent review to methods that are pertinent to visual odometry with an emphasis on autonomous driving. This review covers visual odometry in their monocular, stereoscopic and visual-inertial form, individually presenting them with analyses related to their applications. Discussions are drawn to outline the problems faced in the current state of research, and to summarise the works reviewed. This paper concludes with future work suggestions to aid prospective developments in visual odometry.

3.1 Introduction

Autonomous driving has come a long way since it was first promoted in the DARPA Urban Challenge back in 2007 [2]. With major car manufacturers lobbying their technologies in autonomous driving, the ownership of autonomous vehicles is set to rise in the future. Current autonomous vehicles rely on a variety of sensors to achieve self-localisation and obstacle avoidance. These can include a combination of laser scanners, radar, GPS, and camera. However, the installation of sensor arrays on a vehicle greatly increases its cost and complexity. At the same time, the increasing affordability and ubiquity of cameras

and high-performance graphics processing units (GPUs) are catalysing the resurgence of image processing and computer vision applications. In other words, these applications that were once computationally expensive, are gradually replacing tasks that were performed using other sensors and methods. These tasks include the motion estimation of the vehicle, where precise odometry is crucial for the accurate localisation of the autonomous vehicle. The odometry problem exists such that conventional GPS sensors are unable to provide the necessary road lane precision (≈ 3 m), and that it is unable to function indoors such as inside tunnels and buildings. Additionally, standard wheel odometry suffers from accumulating drift errors that increase over time. While the use of sensors such as high precision differential GPS and inertial sensors could alleviate this problem, they are significantly more expensive to purchase than a standard camera setup.

Visual odometry (VO) is a research area that is becoming increasingly popular in recent years. Ground vehicles and robots rely on odometry to measure and record their traversed path as they navigate, making this an essential component for autonomous navigation. Visual odometry is odometry that is performed by analysing visual data such as one from a mounted camera. This concept was first proposed by Moravec in [146], and the term “visual odometry” was coined by Nistér et al. in [147]. Conventional wheel odometry estimates a robot’s position by measuring the wheel rotation using sensors from the servos. A common issue experienced by wheel odometry is wheel slip, whereby pose estimations becomes incrementally inaccurate from the occasional loss of traction from the wheels. Visual odometry negates this problem.

VO techniques can be classified according to their utilised imagery — either stereoscopic or monocular visual odometry, and their processing techniques — either feature-based or direct (image/appearance-based). These methods can either use a combination of feature matching, feature tracking or optical flow [148, 149]. Since a majority of visual odometry implementation recreates a 3D navigation environment from a set of captured images, most approaches are of a stereoscopic approach that utilises a pair of mounted cameras on the robot. By accounting for the cameras’ capture frame rate and the distance between them, the robot’s displacement and velocity from an object can be calculated with ease through the triangulation of image features [150]. Therefore, the monocular visual odometry problem is more complex and it is not until recently that we are starting to see an increasing trend in this area. Monocular visual odometry achieves motion estimation and environment recreation through a combination of a series of at least three 2D images in series, along with its bearing data. An adaptation of the parallel tracking and mapping (PTAM) algorithm [151] is used in many monocular implementations. PTAM is originally devised for augmented reality (AR) implementations, but its speed and robustness while relying only on existing map features made it a popular choice for researchers of visual odometry.

On the processing techniques front, feature-based approaches achieve motion estimation by extracting image features such as lines and edges, and tracking them in subsequent frames; by calculating the Euclidean distances of each feature points between frames, the displacement and velocity vectors can be calculated. Direct approaches use pixels in an image frame and track the changes in pixel intensity [152], where pixel selection can either be all pixels (dense) or sparsely selected (sparse). In feature-based approaches, feature matching detects and tags existing features on a given set of frames. Feature extraction and matching techniques such as Scale Invariant Feature Transform (SIFT) [40], Features From Accelerated Segment Test (FAST) [153], Speeded Up Robust Features (SURF) [41], Binary Robust Independent Elementary Features (BRIEF) [154] and Oriented FAST and Rotated BRIEF (ORB) [155] are some of the more commonly implemented ones in literature. Feature tracking techniques allow features to be tracked across subsequent frames. This is usually used in tandem with features obtained from a feature extraction technique. Feature tracking is essential for visual odometry, as it allows the robot to achieve a consistent measurement to localise itself [149]. Varying conditions in the environment such as lighting conditions and dynamic obstacles can impede accuracy with outliers. To circumvent this, many works employ the Random Sampling Consensus (RANSAC) [156] outlier rejection scheme or a variation of it; more recently, Buczko and Willert have also proposed an outlier detection scheme for monocular [157] and stereoscopic [158] approaches. Finally, optical flow allows the robot to estimate its distance from an environmental object by tracking its features from the robot's camera feed. With optical flow, the robot can perform obstacle detection and avoidance during navigation. An optical flow algorithm outputs an image pattern that relates to the movement of objects within the robot's field of view (FOV). Examples of popular optical flow algorithms include the Lucas and Kanade's [159], Horn and Schunck's [160], Farneback's [161], and SimpleFlow [162] algorithms. Optical flow algorithms can either be dense (tracks a full frame) or sparse (tracks extracted features). Dense optical flow requires greater computation performance, whereas sparse optical flow methods employ feature extraction prior to its computation to make it less intensive.

It is also common for researchers to use a combination of the three processing techniques to achieve robust optical flow. For example, Wang and Schmid [163] used a combination of SURF descriptors, RANSAC for outlier rejection, and dense optical flow to achieve the prediction of human actions. Liu et al. [164] proposed an optical flow approach based on the Maximum Likelihood Estimation (MLE), which is implemented on a mobile robot and compared against their RANSAC development for optical flow; they concluded that their MLE approach is more accurate than the RANSAC approach. More recently, Kroeger et al. [165] proposed a faster approach for dense optical flow computation using the dense

inverse search (DIS) [166] method, noting that many optical flow proposals have neglected time complexity in favour of accuracy. The authors' evaluation of the DIS fast optical flow showed that while it introduced slight estimation errors, it is much faster even when compared to newer optical flow methods.

This paper reviews monocular and stereoscopic VO methods according to their procedures to achieve motion estimation, as well as their methods of evaluation. A shorter section on visual-inertial odometry is also presented to explore works that combine inertial measurements for VO. The review of these VO methods is intended to gauge their suitability for use in real-time, on-line autonomous driving, which is motivated by the research gap in VO applications for autonomous vehicles. The aim of this article is hence to understand the current trends in VO and to determine if the current state of VO is adequate enough to be utilised in autonomous vehicles.

3.2 Monocular Visual Odometry

Using a monocular camera setup for VO benefit implementations that are lower in cost and complexity. A monocular setup will also alleviate the decrease in depth measurement accuracies as the distance between the camera and the scene increases beyond the stereo baseline. This setup, however, introduces several challenges in addition to the lack of depth measurements on a stereoscopic setup. This was pointed out by Yang et al. [167], where they have investigated several challenges that pertain to this area namely photometric calibration, motion bias, and (assuming that a roller shutter camera is used) the rolling shutter effect. Photometric calibration is required as the pixel intensity for a same 3D point will experience varying values due to the changes in camera adjustment such as optical exposures and gains; motion bias notes that VO performances are different for forward and backward playback on the same sequence; and the rolling shutter effect is predominantly present in rolling shutter cameras whereby an image will distort while a camera is in motion as the frame is captured line by line. The authors then analysed these challenges using a feature-based method, a semi-direct method, and a direct method, which are ORB-SLAM [168], SVO [169] and DSO [170] respectively. The several conclusions drawn from this analysis allowed us to deduce that direct methods are more robust with photometric calibration while being insensitive to pixel discretisation artefacts; while it is affected by the rolling shutter effect, in terms of autonomous driving, using a global shutter camera will nullify this.

For feature-based VO, Chien et al. [171] compared the SIFT, SURF, ORB and AKAZE [172] feature extraction methods for monocular VO. Experiments were conducted on the KITTI dataset using OpenCV 3.1, and concluded that while SIFT is the most accurate at

extracting features, ORB is less computationally intensive, the A-KAZE method sits between SIFT and ORB in computational requirements and accuracy. We hence decided that as our autonomous driving implementation uses an embedded computer, the ORB method is better suited for our applications.

Prominent monocular VO algorithms that are recently proposed include Direct Sparse Odometry (DSO) [170] and Semidirect Visual Odometry (SVO) [169]. As their names suggest, DSO uses a direct approach whereas SVO uses a semi-direct approach to monocular VO. DSO also uses a sparse formulation thereby decreasing computation complexity, as opposed to dense [173, 174] and semi-dense [175, 176] formulations of past proposals. This meant that DSO is capable of achieving real-time computation, as it samples only points of sufficient intensity gradient, and neglecting the geometric prior. DSO functions by continuously optimising photometric parameters from the camera to achieve photometric calibration. This optimisation was performed using a Gauss-Newton method through a sliding window. DSO uses sparse technique whereby it samples data points that are of a limited and equally distributed number across space and active frames, thereby reducing sampling redundancy for data point management. Experimental results showed that DSO as a direct approach is robust against photometric noise, and is able to achieve high accuracies with proper calibration.

SVO was proposed to solve the slow computations and lack of optimality and consistencies of direct methods by combining traits of direct and feature-based methods. This algorithm performs a minimisation of photometric errors on features of the same 3D point, where subpixel features are subsequently obtained through the relaxation of geometric constraints. The minimisation of photometric error is performed at the sparse image alignment stage using a method of least squares, where it assumes that depth information is only known at corners and features that lie on intensity edges. A sparse method uses little depth information and hence the authors enhanced its robustness by aggregating the photometric cost for pixels surrounding the feature, with approximations similar to the feature depth. SVO employs drift minimisation by relaxing geometric constraints and aligning corresponding feature patches to an older reference patch, which is subsequently optimised for reprojection errors using a bundle adjustment. To improve computation efficiency, SVO uses a second thread for mapping, which initialises a new depth filter at FAST corners at every keyframe, thereby estimating the pixel depth using a recursive Bayesian depth filter. Experiments comparing SVO against ORB-SLAM and LSD-SLAM showed that SVO is more efficient at tracking features due to its sparse approach while being robust to high-speed camera captures without the need for outlier rejection methods such as RANSAC. For application requiring high accuracies, the authors used iSAM2 [177], which applies incremental smoothing for the

trajectory motion, thereby achieving the same accuracy as a batch estimation of the entire trajectory in real-time. SVO can also be fused with inertial measurement to further increase odometric accuracies.

3.2.1 Related Applications

Monocular VO algorithms are often tested as a benchmark on datasets such as MonoVO [178] and KITTI’s monocular VO dataset [109]. Recent works that implement monocular VO include the work by Sappa et al. [179], which uses fused images to achieve monocular VO through a discrete wave transform (DWT) scheme where the characteristics of the captured image determine the DWT parameters. Using an image fusion technique condenses information from multiple image frames into one before it is used for motion estimation. This method is compared against VISO2 [180], and experiments were performed on video sequences captured on vehicles driving at different times of day and location. By comparing this fusion approach with previous approaches, the authors noted that the algorithm performs well in challenging environments such as low light drives. Results during daytime are similar across all compared approaches.

The online supervised approach presented by Lee et al. [181] uses ground classification to achieve monocular VO. The authors employed an appearance-based approach with RANSAC over three successive frames to obtain the image flow. Online self-learning is achieved by combining geometric estimates with the ground classifier, which uses a histogram of colour labels. The authors tested their approach on the KITTI odometry dataset where it is compared against the VISO2 algorithm and concluded that their approach was superior in terms of stability and translation performance. While the exclusive use of ground information is adequate to achieve VO for autonomous driving, the authors noted that it could be worthwhile to extend the work to estimate other object models.

Additionally, works that combine semantic segmentation and monocular VO include [182, 183]. An et al. [183] introduced a semantic-segmentation aided VO in order to identify and compensate for dynamic visual obstructions in a camera frame. A modified version of SegNet [106] is used to find visual cues that represent regions of actual motion. The VO method is a semi-direct method whereby in the feature-based section, the authors employed a k -nearest neighbour method to match keypoints from prior frames with its transformation solved using a least-square minimisation method; in the direct (alignment-based) method, the authors used a semi-dense method whereby the framework only utilises regions with certain segmentation labels to ensure that planar objects that are not in motion are selected, which are road and pavement regions, and road markings. Tests were performed on the KITTI odometry dataset and the authors’ Beijing Wuhan dataset. The VO approach was compared

against the VISO [184], DSO and ORB-SLAM2 algorithms, and concluded that a semantic segmentation approach is able to compensate moving objects on the road, where the VISO, DSO and ORB-SLAM2 could not.

The approach by [183] ties into our requirements for our autonomous driving platform, as we also employ semantic segmentation to achieve visual autonomous driving. Also, the authors noted that ORB-SLAM2 achieved the best accuracy on the KITTI dataset on low traffic segments. This could also indicate that an ORB-SLAM-based method with semantic segmentation could be implemented for autonomous driving if it's properly optimised. If performance cost is an issue, we hypothesise that using a ground-only approach such as [181] could achieve adequate VO for autonomous driving, while neglecting dynamic road objects. A robust day-night implementation could benefit from the implementation of a fusion technique as in [179], but its overhead performance cost needs to be taken into consideration. A summary of the monocular VO algorithms reviewed is given in Table 3.1, which lists its approach type, descriptors/features, outlier rejection scheme, dataset evaluated, and intended environment.

3.3 Stereoscopic Visual Odometry

Visual odometry is stereoscopic when a depth measurement is obtainable from a range imaging/RGB-D camera, often through a stereoscopic camera which uses stereo triangulation to measure depth. Other range imaging cameras that are used for stereoscopic VO include structured-light cameras [198, 199] and time-of-flight cameras [200]. This depth measurement enables the distance between the vehicle and its surrounding objects to be perceived, thereby simplifying VO calculations. It is also possible to rely solely on this depth measurement to perform VO, as opposed to using an image-based (feature or appearance-based) approach; this is known as a depth-based approach [201]. Applications for stereoscopic VO are popular, having a list that includes the Mars rover [202] and autonomous aerial drones [203]. Its popularity today can be attributed to the leaderboard on the KITTI VO benchmark [109], whereby a great proportion of the most accurate methods are stereoscopic. As opposed to a monocular approach, stereo VO requires proper calibration and synchronisation of the camera pair, as errors will directly affect VO performance [180].

Stereoscopic VO can also be classified from a combination of feature- or appearance-based approaches. A review was presented by Fang and Zhang [201] in 2014, which compares several stereoscopic VO methods according to their approaches. The authors tested these methods empirically on the TUM RGB-D dataset [204], along with the authors' dataset collected from an indoor environment. This dataset tests the algorithms in a variety

Table 3.1 Summary of monocular VO methods reviewed

Method	Approach	Descriptor	Outlier Rejection	Dataset	Environment
DSO [170]	Direct	Photometric	Threshold filter	monoVO, EuroC, ICL-NUIM	Aerial drone
SVO [169]	Semi-direct	Photometric + FAST	Depth filter	TUM RGB-D, EuRoC, ICL-NUIM, Circle [169]	Aerial drone
Sappa et al. [179]	Feature	SURF (ViSOD) + DWT	RANSAC	KAIST [185], CVC [186]	Road vehicle
Lee et al. [181]	Direct	Geometric estimate	RANSAC	KITTI	Road vehicle
An et al. [183]	Semi-direct	Photometric + SegNet	RANSAC	KITTI, Beijing Wuhan [183]	Road vehicle

Table 3.3 Summary of stereoscopic VO works reviewed

Method	Approach	Descriptor	Outlier Rejection	Dataset	Environment
CV4X [187]	Feature	FAST + BRIEF	RANSAC	KITTI	Road vehicles
Stereo DSO [188]	Direct	Photometric	Threshold filter	KITTI, Cityscapes	Road vehicles
Wu et al. [189]	Feature	KLT	RANSAC	KITTI	Road vehicles
Proen�a and Gao [190]	Feature	SURF + LSD	Circle matching	TUM RGB-D, ICL-NUIM	Pedestrian (Indoor)
Holzmann, Fraundorfer and Bischof [191]	Direct	SAD	Cauchy Loss function	Rawseeds [192], KITTI, [191]	Aerial, ground robots
Jaimbez et al. [193]	Direct	Photometric + Background Seg	<i>None</i>	TUM RGB-D	N/A
Liu et al. [194]	Feature	SURF (ViSOD)	Circle matching	KITTI, New Tsukuba	Road vehicles
Kunii, Kovacs and Hosti [195]	Feature	FREAK, CenSurE	RANSAC	Real world (Izu Oshima)	Mobile robot
Sun et al. [196]	Feature	U-SURF	RANSAC	Real world	Mobile robot
Kim and Kim [197]	Direct	Photometric	t-distribution	TUM RGB-D, real world	Mobile robot

of environments with a combination of fast motion, illumination invariances and limited features. Stereo VO algorithms were measured for their accuracies and computational performances. Results showed that depth-based algorithms such as Rangeflow [205] is robust in environments that lack features or illumination; an image-based algorithm is suitable for feature-rich environments with adequate illumination.

It should be noted that the implementation simplicity of stereo VO, when compared to the added computation complexity of monocular VO, could imply that the research advancements made by monocular VO are more substantial than that of its stereo counterpart; this argument was pointed out by Persson et al. [187], to which they have presented a stereo VO algorithm dubbed the CV4X that leverages on monocular VO techniques. The authors selected a feature-based method using FAST descriptors for corner extraction that is filtered with an Adaptive Non-Maxima Suppression (ANMS) filter, and tracking is performed on BRIEF descriptors. The camera's pose estimation was performed using RANSAC for the perspective- n -point (PNP) problem. Stereo triangulation errors are minimised through an iterative minimising function. CV4X was tested on the KITTI VO dataset and subsequently achieving first place on the benchmark leaderboard at its time of publication. The performance of this algorithm was optimised using OpenMP [206] and CUDA [207] for subtask parallelisation during its experiments.

Monocular algorithms that are adapted into a stereo method also exist, with recent examples including the Stereo DSO [188], where the authors noted that both stereo approaches are complementary, and that multi-view stereo is able to negate the limitation in depth measurements that occur in direct stereoscopy. Multi-view stereoscopy captures stereo images using two or more images [208], whereas direct stereoscopy achieves this using only an image pair. Direct stereo is used here for the initial depth estimation for multi-view stereo. The tracking of features is achieved using direct image alignment [169] that is optimised using a Gauss-Newton method. Experiments were performed on the KITTI VO and Cityscapes [111] datasets to evaluate tracking and 3D reconstruction. Results showed that the Stereo DSO yields low translational and rotational errors when compared to LSD-VO [176] and ORB-SLAM2 [168], with denser and more accurate 3D reconstructions.

The work by Wu et al. [189] is one that combines pruned Kanade-Lucas-Tomasi (KLT) tracking [209] and Gauss-Netwon optimisation with RANSAC to achieve fast feature-based stereo VO. The proposed pruning-based corner detector reduces redundancies while achieving robust corner detections for feature tracking. The KLT tracker was subsequently optimised to track these features through the addition of a pyramidal process. Evaluations on the KITTI dataset showed that while the proposed method was not able to best the state-of-the-art on

the leaderboard, its fast estimation process is able to ensure that VO is performed with a smaller computation footprint.

Methods that rely on time-of-flight or structured-light cameras such as the Kinect sensor captures depth maps that are prone to noises that affect the accuracy of depth measurements. Feature tracking on depth maps can be simplistic due to its reduced detail and colour as compared to an RGB image. For example, Proen  a and Gao [190] recently proposed a minimalistic environmental representation that combines points, line and planes to achieve VO. The authors noted that the limited field-of-view and the presence of noise on the depth map, which prompted them to propose a method that performs noise reduction using the missing depth measurement recovery technique with depth uncertainty modelling. The extraction of these features is done using SURF for points, LSD [210] for lines and the plane model is fit through a segmented point cloud. Experiments were performed on the TUM RGB-D dataset and the ICL-NUIM [211] dataset, as well as a dataset collected by the authors. Results showed that this method outperforms the point and line VO methods, as well as DVO and FOVIS [199], this method is unable to outperform methods such as [212, 213].

Holzmann, Fraundorfer and Bischof [191] proposed a line-based direct stereo VO method using vertical lines. Limiting detections to vertical line enhances detection speed thereby making it suitable for real-time applications. Line verticality is determined using an IMU or a gravity-aligned camera. These lines are matched at every keyframe with direct pose estimation to achieve fast VO. The authors noted that the algorithm performs well in man-made environments, especially indoors with walls and fixtures, even in poorly textured environments; this approach is also adequate for urban driving scenes with well-defined buildings, vehicles and road edges, as experiments on the KITTI VO dataset revealed that this method achieves results that are comparable to VISO2. However, the heavy presence of textures in outdoor drive scenes prevented the accuracy of this method to surpass that of VISO2.

Another recent method to achieve fast VO is through geometric clustering. Using geometric clustering solves the same problem with dynamic scenes as listed in [183], whereby objects in the frame that are in motion are capable of distorting VO accuracies. The approach presented by Jaimez et al. [193] applies k-means clustering on image points, in addition to segmenting image regions where static objects such as road regions for VO. The authors calculated VO through the minimisation of photometric and geometric residuals between the stereo image pairs, which is then estimated using a Cauchy M-estimator. VO performances were evaluated on the TUM RGB-D dataset and are compared against DIFODO [205], DVO and SR-Flow [214]. Results showed that while this approach performs remarkably on dynamic scenes, it is unable to outperform the accuracies of the other algorithms on static

scenes. Nevertheless, this method is capable of fast runtimes and real-time performances, which was not achieved by the other compared algorithms.

In order to improve outlier rejection in dynamic environments, the work presented by Liu et al. [194] presents a stereo VO method that aims to improve accuracies using an improved outlier rejection method. The PASAC method is an improvement of RANSAC that achieves higher accuracies through a series of procedure. The authors noted the degradation of RANSAC's accuracy on frames with many outliers, as well as its uniformly generated hypothesis through the sampling of input data, thereby motivating the proposal of PASAC though an enhancement to its hypothesis generation for increased outlier rejection speed and accuracy. As a feature-based method, this method first extracts corner-like features from a stereo image pair, which are then matched using a sum of absolute differences method. Outliers from the image pair and its subsequent frames are then rejected through circle matching to identify mismatched features before the features are tracked according to its detection timestamp. Motion estimation is subsequently performed by an iterative solving of the non-linear least square optimisation problem, with PASAC as the outlier rejection model. This approach was tested on the KITTI VO and New Tsukuba [215] datasets, outperforming RANSAC and PROSAC [216] in execution speed and accuracy.

3.3.1 Related Applications

An implementation of stereo VO was first described by Nistér et al. in 2006 [147], which estimates the ego-motion of a camera mounted on an autonomous ground vehicle. Since then, recent stereo VO methods are mostly applied in the robotics field, where they are often implemented onto mobile robots and aerial drones. While datasets from KITTI and TUM are often used to evaluate new stereo VO algorithms, real-world VO applications on-road vehicles are, to our knowledge, quite scarce. For example, while a recent thesis by Aladem [217] described VO for autonomous driving, its evaluations of VO is limited to a ground robot and datasets; an attempt was made to collect local data from a car-mounted camera, but it resulted in unfavourable VO results. For these reasons, we will look into VO methods that are implemented onto mobile ground robots, as its application is most similar to that of a ground vehicle.

The application by Kunii, Kovacs and Hoshi [195] uses a feature-based stereo VO method on a mobile robot through landmark tracking. The authors extracted FREAK [218] descriptors using CenSurE [219] after comparing various extraction and descriptor methods for its environment, and stereo-matching is performed using a sum of absolute differences; RANSAC is used as the outlier rejection scheme. By using these VO parameters, the authors compared VO performance against GPS data and deduced errors of less than 5% in both 2D

and 3D space. The VO method is enhanced using template matching to improve computation footprint and accuracy, which returns the robot's position relative to the obstacles ahead. By comparing several methods against a laser scanner, the Zero-Mean Normalised Cross-Correlation (ZNCC) method was used as it returns the least amount of deviation from the laser scanner. Field experiments confirmed that the addition of template matching is able to result in accurate localisation.

Likewise, the application by Sun et al. [196] implements stereo VO on a differentially-driven mobile robot. The VO method is feature-based whereby feature extraction is done using an upright SURF method, and stereo-matching is done using the perspective-three-point method; RANSAC is used as the outlier rejection scheme, which is also applied to the perspective-three-point method to filter erroneous landmarks. The authors proceeded to describe a fuzzy model for driving the robot that utilises stereo VO for trajectory stabilisation. Experimental results showed that the robot was able to achieve accurate trajectories based on the utilisation of VO alongside the Takagi-Sugeno fuzzy model [220].

Noting that applying VO on background sections of a frame can result in better accuracies in dynamic environments, the application by Kim and Kim [197] implements a depth-based dense VO onto a differential drive robot. This background model-based dense-visual-odometry (BaMVO) algorithm isolates moving objects in the foreground by using a nonparametric model in [221], measuring the depth differences for objects in consecutive frames. VO on the background model is performed similarly to DVO [222], using a minimisation of the weighted sum of squares method. Outlier rejection of the background model is performed using the method in [223], which filters outliers over a t-distribution. The BaMVO was evaluated on the TUM RGB-D dataset before it was implemented for trajectories captured on the mobile robot in a dynamic indoor environment with pedestrians. The algorithm was compared against DVO and other state-of-the-art, which resulted in the BaMVO being the most accurate, especially in dynamic environments where the other algorithms had erroneously calculated VO based on moving objects.

By evaluating the aforementioned methods with regards to autonomous driving, the contributions made by [195] involve using a template matching method to enhance VO accuracies can be adapted for autonomous driving whereby it can be used to estimate the vehicle's distance relative to its surrounding obstacles. By complementing ZNCC with stereo VO techniques, a more robust VO solution can be applied for autonomous driving. The proposal in [196] which utilises stereo VO to ensure accurate motion trajectories can be similarly applied for autonomous driving to complement the path planning module of the autonomous car. This means that the adjustments in trajectories can be dynamically adjusted using results from VO to result in added robustness for path planning. Finally, the

takeaway from Kim and Kim's [197] application suggests that adopting a VO algorithm for dynamic environment enables the vehicle to filter moving road objects such as other vehicles to restrict VO calculations on static, background models, thereby increasing VO robustness. A summary of the stereoscopic VO algorithms reviewed is given in Table 3.3, which lists its approach type, descriptors/features, outlier rejection scheme, dataset evaluated, and intended environment.

3.4 Visual-Inertial Odometry

Visual-inertial odometry (VIO) is a technique whereby a VO method is fused with the output from an inertial measurement unit (IMU) in order to improve odometry accuracy. Most VO algorithms can be adapted for VIO, resulting in VIO approaches that are either stereoscopic or monocular; using direct, semi-direct or feature-based methods. The addition of an IMU to VO effectively introduces a fusion pipeline that uses state estimation filters such as the extended Kalman filter (EKF) or particle filters; this is a filtering-based VIO approach, which encompasses most proposed VIO works [224]. In addition to using a filter, the different sampling rates between the IMU and the camera (i.e. the fusion interval) needs to be synchronised, as IMUs generally sample at a rate that is several times faster than the camera's frame rate. This synchronisation is typically achieved by the resampling of the IMU data at the camera's frame rate [225].

OKVIS [226], MSCKF [227] and VINS-MONO [228] are few of the more popular examples of filtering-based VIO methods in literature; additionally, new VIO methods that are based off existing VO methods such as SVO [229, 230] are often proposed as well. Delmerico and Scaramuzza [231] recently published a benchmark comparing monocular applications of these VIO methods across several hardware platforms to measure their performances for autonomous aerial drones. These methods were tested on the EuRoC MAV [232] dataset, which consists of visual-inertial sequences that were recorded off an aerial drone. Results from these benchmarks concluded that algorithms which result in higher accuracy and robustness generally require higher computation requirements, thereby implying that VIO methods have to be carefully selected and optimised for their specific applications. VIO methods are often applied to autonomous aerial drones as the addition of an IMU sensor enables estimations up to the six degrees of freedom (6DoF) that is required for their functions. Additionally, a recent method for stereo VIO was presented by Sun et al. [233], which postdates this benchmark with the proposal of the stereo multistate constraint Kalman filter (S-MSCKF) method. FAST features are tracked using KLT tracking with RANSAC to remove outliers. Comparisons against OKVIS, ROVIO, VINS-MONO on the

EuRoC dataset and on an autonomous aerial drone resulted in the S-MSCKF was able to achieve a balance between computation footprint and accuracy, which suggests that it can be applied on cost-sensitive platforms. While applications on aerial drones are different from autonomous ground vehicles, the benchmarks and works done by [231] and [233] can certainly be used when choosing the proper VIO method for implementation.

A stereo application on mobile ground robots is described by Liu et al. [234], where they have implemented a stereo VIO method onto a model remote controlled car. The VIO method that was proposed utilises multiple Kalman filters for position, orientation and altitude for increased robustness. As the IMU used in the application is considered to be low-cost, the authors also proposed a cascading fusion architecture to estimate orientation measurements, as well as using linear subfilters with low computational footprints with the intention of an embedded computer application. The stereo VO method is a feature-based one whereby SURF descriptors are tracked from the feature pair from each camera, which is detected using the CenSure detector; outlier removal is done using RANSAC. The proposed VIO method was first tested on the KITTI dataset and noted that a pure VO approach will fail at certain turning corners, which is thus rectified with a VIO approach. The second test was performed in a real-world campus pedestrian environment using the mobile robot, where the authors noted that their VIO approach has the least closed-loop error when compared against other state-of-the-art methods, yielding high accuracies while reducing IMU drift errors. While the authors stated that an implementation on an embedded computer with other sensors such as the GPS is part of their future work, the relation to this and autonomous driving is significant, whereby the same method can be adapted for urban road environments, especially when it is benchmarked on the KITTI dataset.

More recently, the use of event cameras for VIO has been proposed by Vidal et al. [235]. These cameras are capable of high frame rates and are insusceptible to motion blur, as they transmit pixel intensity changes and not the intensity itself. This method achieves VIO with both an event camera and a standard camera, by tracking FAST corners using the KLT tracker, as well as using the Ceres Solver [236] for optimisation, which is a non-linear optimizer that selects between the event camera and the standard camera for VO, based on its current environmental conditions around the keyframe. Evaluations were performed on the Event Camera Dataset [237] comparing results from using frames (standard camera), events (event camera) and the IMU, whereby the combination of events, frames and IMU yielded significantly better results. Real-world tests on an aerial drone were tested to be resilient against sudden illumination changes, with accurate positioning even in low-light conditions. The high frame rate and robustness achieved using this sensor combination yields results that are real-time and accurate. These results will be highly beneficial to autonomous driving

where precise positioning and real-time frame calculations are required. However, the lack of urban road testing by the authors implies that this method should be replicated to assess its capabilities in an autonomous driving environment.

3.5 Discussions

This section discusses our overall observations and deductions with regards to the works that were reviewed that are in-line with the current research trends in VO, as well as the requirements for VO to be implemented in autonomous driving applications.

An application for autonomous driving will require that the implemented methods are capable of running in real-time; in the case of visual algorithms such as VO, 10 Hz is the minimum desired frame rate in order for the vehicle to sustain driving in urban environments [129], while some are explicitly optimised for real-time applications [191, 193], other methods could be further optimised for real-time performances. It should also be noted that the speed performances of VO algorithms can be further improved just through the use of hardware with higher specifications such as high-performance GPUs. Additionally, in order for a VO method to be accurately robust for autonomous driving, the method should account for the various environmental dynamism that occurs on road scenes. This includes moving obstacles, scene changes and illumination invariances. This was noted in [183, 197] where dynamic changes in the road scene will affect the accuracies of VO. Through our studies, this dynamic road scene problem can be addressed either through an outlier rejection model such as RANSAC or PASAC [194], or through a semantic selection of classified objects [183]. We observed that while the object classification approach might yield higher accuracies due to its deep learning back-end, using an outlier rejection method is computationally simpler and it is more capable of the real-time performance required for autonomous driving.

We observed an insufficiency in real-time VO methods that are explicitly implemented in an autonomous vehicle. While there exist VO implementations on vehicles such as on the parking camera [238], to the best of our knowledge VO implementations for autonomous vehicles are unavailable. Most autonomous vehicle developments utilise the camera for lane-keeping and obstacle detection/avoidance. However, effective autonomous driving will require precise vehicle localisation and dead-reckoning in the order of centimetres for it to navigate even in unmapped environments. This precision is unattainable through conventional GPS receivers with accuracies of approximately 10 metres; expensive differential GPS receivers are typically installed to achieve this but it introduces redundancy when VO algorithms themselves can be utilised for accurate localisation. Another option to achieve relative localisation is through the installation of an IMU, but unless the IMU is highly

precise, the drift errors introduced by an IMU will exponentially affect accuracies; hence the proposals of visual initial odometry. The lack of real-world implementations can also be credited to local legislation and the lack of development vehicles for autonomous drive tests. Here we see that almost all of the recent VO works presented are tested on datasets, and while public datasets such as KITTI or TUM is a great platform and yardstick for method comparisons, it remains to see how these methods will eventually perform in the real world. Testing on these datasets also limit the testing environments to the location where the dataset is captured and is not an effective indicator of the performances of these algorithms in other cities and countries, which will introduce different road scenes altogether. Another reason for the lack of real-world implementation is likely due to the higher computation requirement of VO, which implies that an on-line implementation will require a computer with dedicated and adequate parallel processing hardware. Commercial autonomous driving computers such as Nvidia's DRIVE PX2 [239] are expensive and are generally unaffordable for developments on a budget, whereas mobile computers such as laptops do not possess the parallel computing capabilities of desktop GPUs. However, the recent availability of low-cost, high performance embedded computers such as the Nvidia Jetson [117] and the optimisations of fast VO methods [189, 193, 205, 222, 233] could catalyse these implementations.

VO methods are instead often tested on aerial drones and mobile robots, as they usually provide better feasibility and cost-effectiveness as compared to an actual vehicle implementation. VO on an aerial drone is more complex and often fused with inertial measurements as it has more DoF than a ground vehicle; however, VO on a mobile ground robot also differs from an autonomous vehicle whereby it is difficult to replicate the dynamism of road scenes outside of a simulated environment or in the real world.

As we observed that many VO methods evaluated its performances against visual SLAM (vSLAM) algorithms, this should suggest that VO is similar to vSLAM such that vSLAM uses VO to achieve relative localisation. VO is however different from vSLAM whereby it does not perform loop closures that are necessary for area mapping. Since an autonomous vehicle does not rely on mapping for odometry nor localisation, vSLAM is therefore beyond the scope of our applications.

3.6 Conclusion

The availability of recent works is greatly contributing towards the solution for the visual odometry problem. We have reviewed the various types of visual odometry methods in relation to their applications for autonomous driving. Both monocular and stereoscopic VO are viable approaches for autonomous driving, whereby the hardware will only differ

according to its camera setup. The easy attainability of publicly available datasets for autonomous driving such as the well-known KITTI dataset is also a major contributing factor that encourages works in VO. By reviewing recent works pertaining to VO that are not more than three years old, we can confirm that the current VO trend is steering towards a low-cost, high accuracy model that encourages applications on low-powered hardware such as embedded computers. Although the availability of datasets is promoting the proposal of new VO algorithms, we have also observed a current shortage of real-world VO applications, especially in autonomous road vehicles. Many proposed methods stop short of a practical implementation, and only evaluated their algorithms on datasets. We have observed that results from a dataset evaluation often deviates from a complete indication of how the algorithm will perform in a local environment, thereby highlighting our necessity for a practical VO application. We have deduced from our observations that a couple of factors could attribute to this issue. Firstly, the attainability of test beds for autonomous vehicles is often associated with high costs and complex local legislation, as it usually involves the purchase and retrofitting of an actual road vehicle, especially when considering the higher probability of accidents when new algorithms are tested. On the other hand, while embedded computers are now capable of efficient parallel processing, they are still often unable to provide the necessary computing performance required for visual navigation on the test bed, especially when we compare them against workstation-class GPUs that are typically used to evaluate newly proposed algorithms. Nonetheless, forthcoming high-performance mobile computers and an increasing public recognition towards autonomous vehicles will undoubtedly encourage practical applications of visual odometry in the near future.

Chapter 4

Cooperative Multi-Robot Navigation — SLAM, Visual Odometry and Semantic Segmentation

This chapter describes three systems for multi-robot navigation: multi-robot SLAM for large environments, visual odometry for high odometric accuracy, and semantic segmentation for dynamic scene understanding. Our multi-robot SLAM is capable of navigation, exploring and mapping a large-scale urban environments. Environmental exploration and mapping are achieved through wheel odometry and LiDAR, which interface through the Robot Operating System (ROS) framework on each robot. This system is distributed and decentralised whereby each robot performs localisation and mapping independently, while maintaining persistent maps are being transmitted to a ground control centre that verifies the map data. Visual navigation in the form of visual odometry is used with minimal changes to the existing software and hardware framework. We use visual odometry as an alternative to wheel odometry, which is affected by wheel slip, a persistent error that accumulates over time. Semantic segmentation complements object detection data from the LiDAR by introducing a pixel-based object recognition method that allows each robot and their visual odometry to vary its reaction based on the detected object class. These visual navigation subroutines can complement the existing mapping and localisation routines as an alternative solution.

4.1 Introduction

The University of Western Australia (UWA)'s multi-robot system (MRS) [240] comprises seven Pioneer 3AT-based outdoor robots. It was designed to solve the multi-robot simultane-

ous localisation and mapping (SLAM) problem through strongly coordinated behaviours with task allocations that are performed explicitly whereby each task is divided into subtasks that are dynamically allocated and re-allocated in response to changing conditions or failure [241]. In other words, this system is capable of structured communications while being aware of one another.

For an MRS to properly navigate an environment and perform cooperative tasks, these localisation estimates need to be robust. Our system uses a contained localisation system to allow for rapid deployments in unstructured/mixed indoor/outdoor environments, which is originally presented in [242]. Path planning and obstacle avoidance currently follow our implementation in [243, 244]. We use a multi-robot SLAM (MR-SLAM) solution to localise robots through the construction of a shared local map. The MR-SLAM problem is complex, whereby localisation is achieved through the registration of each robot in a consistent, global coordinate system, in which large amounts of sensor data fusion must occur on-line. Additionally, these robots often rely on wireless communication that is often lossy and subject to interferences with variations in latencies and bandwidth. Loop closures that are predominant in SLAM problems become more challenging, as they create large sequences of constraint cycles, which can cause a combinatorial increase in computational complexity. This problem stems from the variations between the robots' vantage points where uncertainties in data association might arise due to the pairings between object detections and sensor measurements.

MRS projects presented in the recent literature [245–247] demonstrated a combination of low-level capabilities such as cooperative SLAM, exploration, object identification, object tracking, and object manipulation. A comprehensive review is available in [248, 249]. These systems are off-line or on-line. Off-line systems collect the first sensor data that is processed at a later stage, while on-line systems perform MR-SLAM and other tasks in real time during deployment. It is worth noting that off-line systems are typically deployed based on their ease of implementation and prototyping, while often relaxing limitations imposed on computation and communication requirements. Additionally, many works are often implemented in constrained environments, such as in indoor laboratories [250, 251], thereby preventing the robots from exposure to environmental irregularities, including noise, temperature, illumination, and seasonal variations. These systems can therefore be cheaper and more convenient to implement, as they usually do not require long-range mobility and sensors. Conversely, our MRS was designed for deployment in unconstrained, outdoor urban environments, requiring higher performance sensors and more rugged robots.

The incorporation of visual navigation onto the MRS stems from our motivation to solve problems relating to wheel slip in odometry and scene understanding. Works that implement

visual odometry in robots [195–197] have proven that its ability to reduce the accumulating error caused by wheel slip can lead to more robust SLAM solutions. Visual SLAM algorithms such as ORB-SLAM [168] and LSD-SLAM [176] are often implemented on robots, with favourable outcomes. Likewise, scene understanding can be applied alongside obstacle detection from LiDAR measurements, such as classifying static and dynamic obstacles [252]. For our application, we have decided to apply semantic segmentation for scene understanding, as it offers a pixel-wise classification of a captured scene while being versatile and compatible with low-cost camera setups.

4.2 Robot Hardware Design

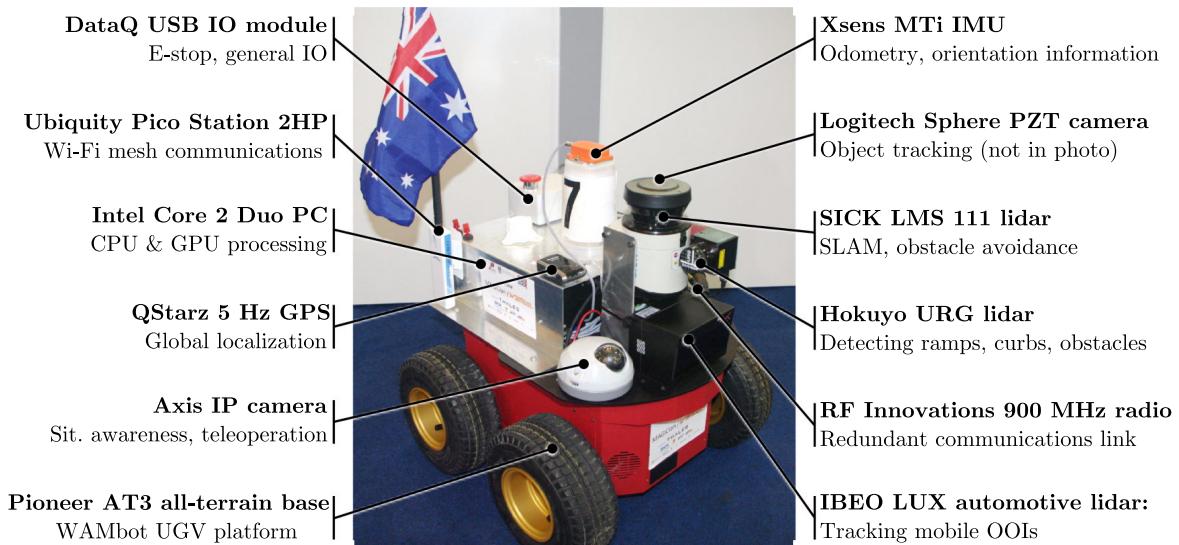


Fig. 4.1 Photo illustrating an single MRS UGV with its hardware modules as labelled.

Each unmanned ground vehicle (UGV) in our MRS is fitted on top of a Pioneer AT3 [253] base, which provides a chassis, differential drive wheels with motor controllers and encoders, and batteries (see Fig. 4.1). High-level controls are performed through an Intel Core 2 Duo automotive PC that is connected to several sensors (see Figure 10.1). The sensors comprise an ibeo LUX 4 LiDAR [254], a SICK LMS-111 LiDAR [255], a Hokuyo URG-04LX LiDAR [256], an Xsens MTi inertial measurement unit (IMU) [257], a QStarz GPS receiver [258], wheel odometry on the Pioneer base, and a Logitech Sphere PZT camera [259]. Communications are performed between UGVs and base station through a Ubiquity Pico Station 2HP [260] over a Wi-Fi mesh, with an RF Innovations 900 MHz radio [261] as a

redundant communications link. The Pico Station, LUX 4, and LMS-111 interface via 100 Mbps Ethernet, while the other sensors interface through USB 2.0.

The sensors perform LiDAR-based SLAM, whereby the LiDAR array maps the environment horizontally and has a 20 m, 270° range at 25 Hz; the URG-04LX is mounted vertically and has a 4 m, 240° range at 10 Hz; and the LUX 4 is mounted horizontally and has a 50 m, 110° range that spans across four parallel, horizontal layers. The LMS-111 is used as the main SLAM sensor, where it is placed 0.5 m above ground to scan a single-layer horizontal plane. This results in 1080 measurements that translate to 2D “slices” of the environment around a 20 m radius. The other LiDARs mounted on the UGV are used for object/obstacle detection and tracking.

4.3 Cooperative Localisation and Navigation

We incorporate our hybrid-decentralised and distributed MR-SLAM system onto the UGVs, which allows the decentralised UGVs to build distributed global grid-maps and navigate large urban areas [262, 263]. Using this system enables the system to be deployed rapidly while allowing SLAM on the UGVs with minimal reliance on a ground control system (GCS).

4.3.1 Mapping

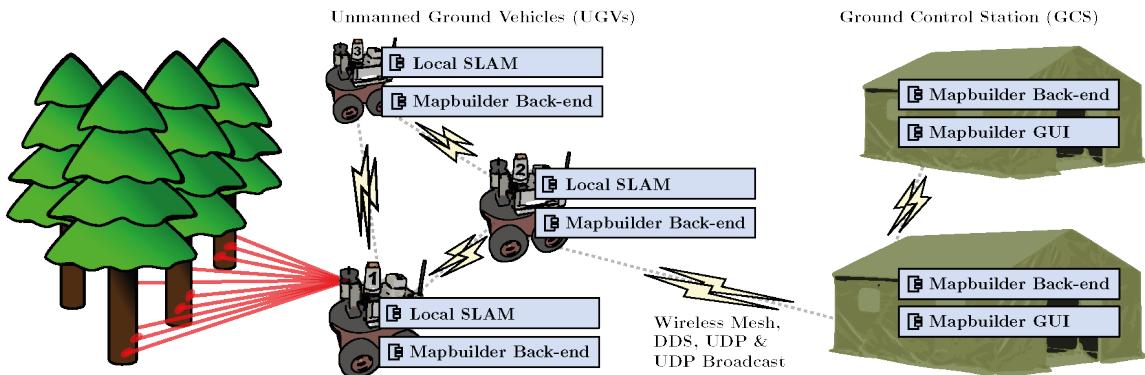


Fig. 4.2 The MR-SLAM architecture and software development diagram showing the software components run on the GCS and UGVs.

A typical deployment scenario of the MR-SLAM system is illustrated in Fig. 4.2, showing that the back-end is executed across UGVs and GCSs. Each back-end instance stores a local copy of all submaps and constraints, which are then optimised and fused, building global maps; new loop closure constraints between submaps are also searched. Submaps are

rectangular grid-maps with dynamically increasing dimensions determined by the LiDAR’s maximum range R , the environment’s shape, and a threshold heuristic described later in this section.

Local SLAM is performed independently on each UGV whereby a single-robot SLAM algorithm builds its own submap by processing its sensor data, which is then broadcasted across the mesh network. Graphical user interfaces (GUIs) are installed on GCS computers to enable operators to view and manipulate global and submaps and interact with pose graphs through a point-and-click interface.

A global grid-map is fused through several overlapping submaps that are obtained from the SLAM algorithms running on each UGV, thereby achieving a distributed map-building sequence. The fusion algorithm searches for overlaps in the grid-maps and subsequently determines if the cells in the global grid-map are occupied, free, or unknown. Each submap initialises with a local coordinate frame on the global frame with its submap pose p_a^W , which is estimated through pose graph optimisation in a global Euclidean coordinate frame W . Given the UGV pose r_t^a always broadcast relative to p_a^W , and that each submap is created with the UGV at its origin, therefore $r_0^a = [0, 0, 0]^\top$, and the UGV’s time-varying pose in the global frame is thus:

$$r_t^W = p_a^W \oplus r_t^a \quad (4.1)$$

Each submap is assigned a 128-bit hexadecimal universally unique identifier (UUID) and exists either in an “open” or “closed” state, where they are always created through an “open” state to indicate that an occupied UGV is in the process of building it. Once map building is complete, the submap changes to a “closed” state to render the area immutable and non-traversable by any UGV, only allowing the back-end to update its pose, thereby fusing it onto the global grid-map. Having these states increases the robustness of the MR-SLAM system while ensuring its logic simplicity. A UGV can only occupy a single submap at any given time, and these “open” submaps are often connected to an adjoining “closed” submap. At the creation of a new submap, its map and pose uncertainty is reset. Using this approach enables the system to minimise bandwidth, storage, and redundancy across the MR-SLAM back-end.

Using a ray tracing technique based on [264], the LiDAR scans obtained while navigating a submap is aligned and fused into a single 2D occupancy grid-map, which maintains an accurate representation of the environment. The LiDAR scans are aligned with scan matching prior to ray tracing to circumvent the accumulation of minor quantisation noise, which is done through a batch rounding of these LiDAR measurements to the nearest grid cell using the grid-map representation.

Aside from quantisation noises, UGV pose uncertainties are also prevalent while it is building a submap. Although a UGV always initialises a new submap with no pose uncertainties, this uncertainty will always accumulate whenever the UGV is manoeuvring, with odometric noise as its main contributor. Therefore, it is more pronounced in larger areas. To solve this problem, the algorithm initialises a new submap whenever this uncertainty surpasses a set threshold, which is determined by comparing the current angular pose uncertainty against the average distance to obstacles in the environment, estimating the amount of “blurring” in distant grid-map cells. Current LiDAR scans will not be fused if a new submap is triggered using this approach; using this heuristic thus minimises distortions entering the submap grid-map, and large distortions that result in misaligned LiDAR scans can be prevented. The algorithm then transfers this accumulated uncertainty’s covariance into the new constraint’s covariance that is used to connect the old and new submaps through a maximum likelihood estimation.

By assuming an average distance between submaps D , the maximum overlap between a sequence of submaps separated by D is given by a ratio:

$$\text{Maximum overlap} = \frac{2R}{2R+D} \quad (4.2)$$

The MRS yields a maximum submap overlap of 93%, which implies that the same UGV could create submaps that overlap the same area up to 15 times. This overlapping redundancy is required to allow the distributed back-ends to compare and align map data.

4.3.2 MR-SLAM Architecture

With reference to Fig. 4.2, we have identified the functional roles of the software components as Table 4.1, which illustrates a minimalistic logical design diagram that considers a single UGV and GCS.

Table 4.1 Functional requirements of software components

Component	Input	Behaviour	Output
Local SLAM	Sensor data (LiDAR, odometry, IMU, GPS)	Performs local SLAM, creates a sequence of submaps	Broadcasts submap data, constraints, real-time UGV pose estimates
MR-SLAM back-end	Submap data from all UGVs, submap constraints, ground-truth constraints	Optimises pose graphs, fuses submap data, searches for constraints	Global or windowed maps, submap pose estimates, submap constraints
MR-SLAM GUI	All MR-SLAM messages, GUI events; for example, keystrokes and mouse clicks	Displays global maps, interprets operator commands	Messages that alter graph structure; for example, ground-truth constraints

The class diagram in Fig. 4.3 illustrates the various message types used by the system for MR-SLAM, which are all derived from the Submap message class. All messages are time-stamped with the source participant’s priority and the submap UUID.

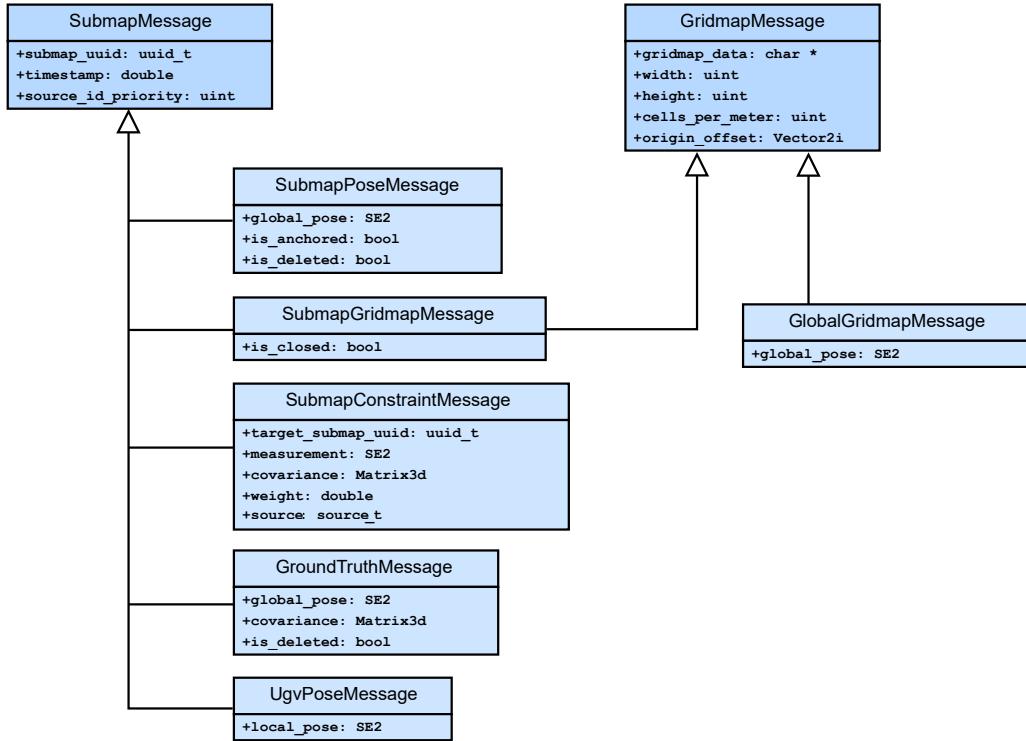


Fig. 4.3 A class diagram showing the MR-SLAM system’s message types and their variables.

For the local SLAM front-end, we use a heuristic-driven EKF-SLAM [265] single-robot algorithm that takes all sensor data and outputs submaps, constraints, and real-time pose estimates, which are broadcasted over the mesh network. We have designed this front-end based on the following requirements:

- To estimate UGV pose and broadcast in real-time at more than 10 Hz locally, or 1 Hz globally.
- To build 10 cm submap grid-maps that are broadcasted at more than 1 Hz locally, or 0.2 Hz globally.
- To robustly handle moving objects including human gaits up to 6 km/h.
- To handle challenging sensing conditions such as sparse and/or featureless areas.
- To detect odometric errors to minimise submap corruption.
- To compress submap grid-maps before broadcasting.
- To use less than 25% of total computation and memory footprint.

Likewise, the back-end requirements of our MRS are:

- To optimise pose graphs robustly and efficiently at less than 5 seconds per iteration.
- To output large (5000×5000) grid-maps to the local partition at more than 1 Hz.
- To match submaps to generate robust constraints at less than 5 seconds per match.
- To broadcast **SubmapPose** messages globally to maintain decentralised pose graph.
- To output **SubmapPose** updates to the local partition at more than 1 Hz.

4.3.3 SLAM Implementation

We apply EKF-SLAM with scan matching which builds submaps by aggregating LiDAR scans at every 20 cm of movement or 20° of rotation, where pose estimation is achieved through an EKF. Scan matching was incorporated to reduce computation requirements by using the described threshold heuristics to decide when a current submap should be closed, which is augmented by a threshold on the percentage of LiDAR returns that are aligned successfully. This method enables the detection of matching failures, especially in sparse environments. EKF is used to estimate the UGV's pose by initiating each cycle to predict its current pose using the latest wheel odometry and IMU data, which aligns the LiDAR scan against the current submap through scan matching [266]. RANSAC [156] is also incorporated to reject outliers in the form of moving objects. The EKF and pose estimate is subsequently updated using this scan matching alignment method. Odometric noises that are present in the EKF update are adjusted according to the local ground slope gathered from the pitch and roll measurements from the IMU. An increase in slope leads the module to assume an increase in odometric noise due to wheel slip, thereby switching the filter preference for scan matching over odometry.

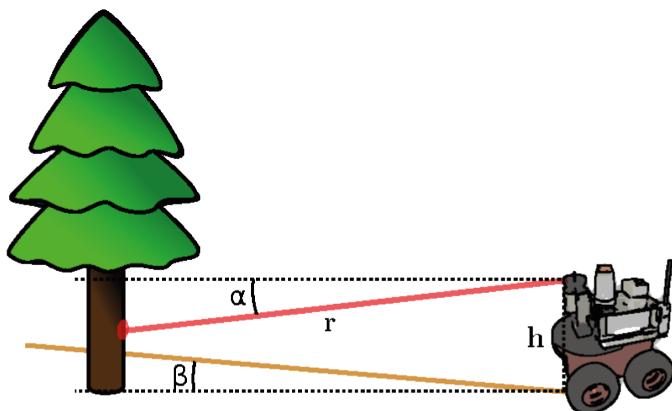


Fig. 4.4 Lengths and angles used for calculating the local SLAM prefilter.

To filter these errors using the IMU, we assume that (1) the terrain inclination is less than β_{\max} , and (2) the terrain follows the ‘‘Manhattan World’’ [267] assumption. With reference to Fig. 4.4, each LiDAR measurement range r is first corrected to account for the declination α from the same measurement as $r \cos \alpha$, whereby an inequality is obtained referencing the height of the LiDAR mounted above the ground h .

$$0 < h - r \sin \alpha - r \cos \alpha \cdot \tan \beta_{\max} \quad (4.3)$$

To prevent any measurements from grazing the ground, any instance of r with declination α that dissatisfies (4.3) is filtered. LiDAR measurements that are corrected and filtered will then be passed to the SLAM algorithm.

4.3.4 UGV/GCS Communications

Communications between UGVs and GCSs are facilitated through a Wi-Fi mesh network over the IEEE 802.11n standard in a multi-hop configuration over a data distribution system (DDS), which provides a publisher-subscriber framework that provides robust real-time communications. Publishers are separated into partitions which are either global (all participants) or local (within a participant). Global partitions are mostly used to prevent network overloads, as the local partition is used for high-rate inter-process communications, where messages are passed over a shared memory between the front-end, back-end, and other high-level MRS software components.

Messages are broadcasted by the front-end as submaps are closing in the form of compressed grid-map data and the constraint that links the closed submap to the new one. Incomplete grid-maps for open submaps are also broadcasted to visualise real-time global maps; these maps are flagged to indicate that they are not yet immutable. The front-end broadcasts three distinct, time-stamped message types with a fixed DDS buffer size n with varying quality of service (QoS) priorities:

1. **SubmapConstraint** ($n = 1000$) defines the entire pose graph structure. A large buffer size is allocated for the series of small yet vital messages.
2. **SubmapGridmap** encodes the actual shape of the environment, constituting most of the MR-SLAM data, which are either open or closed.
 - (a) **Open** ($n = 0$) grid-maps are disposable as they are periodically broadcasted by the front-end, these are sent to the local partition at the LiDAR’s scan rate.
 - (b) **Closed** ($n = 100$) grid-maps have higher priority as they are only transmitted once.

3. **UGVPose** ($n = 0$) are broadcasted frequently in real-time, which stale quickly and is subsequently disposable. These are also sent to the local partition at the LiDAR’s scan rate.

A rendering algorithm ray traces the accumulated LiDAR scans into an empty grid map based on the methods described in [264]. The grid-map data is segregated into 32×32 cell tiles that are broadcasted over UDP on a 50 : 1 compression ratio.

Likewise, the back-end’s publishing policies are:

1. **SubmapPose**:
 - (a) **Global Partition** ($n = 0$) does not require delivery guarantees since priority-based filters synchronises this between participants.
 - (b) **Local Partition** ($n = 0$) does not require QoS as DDS uses shared memory to delivery real-time submap pose estimates.
2. **SubmapConstraint** ($n = 1000$) is assigned with the highest priority as they define the entire pose graph structure, hence a large buffer size is allocated.
3. **GlobalGridmap** ($n = 0$) are delivered through shared memory by DDS in real-time; QoS is therefore not required.

4.3.5 Loop Closures

A loop closure is an event that occurs when a UGV revisits a location it has previously threaded, thereby correcting its accumulated errors. The identification of loop closures occurs between overlapping submap pairs and spatially similar grid maps, which are broadcasted as new constraints that form cycles in the distributed pose graphs, bearing residual errors that require optimisation. The back-end searches for local loop closures between “open” submaps in real-time, especially when multiple UGVs are operating overlapping areas to ensure proper localisation and prevent the accumulation of errors. Using this method enables our system to efficiently accommodate high rates of loop closures and map changes in real-time.

The large number of loop closures generated by the system, as well as the distributed algorithm design, prompted us to utilise the graphics processing unit (GPU) to search for loop closures and merge submaps. Additionally, descriptive spatial relationships between submaps are extracted using a grid-map correlation algorithm on the GPU that calculates likelihood volumes and extracts multimodal Gaussian constraints. Using robust multimodal constraints enables the algorithm to preemptively add loop closures to the pose graph and perform outlier rejection by consensus.

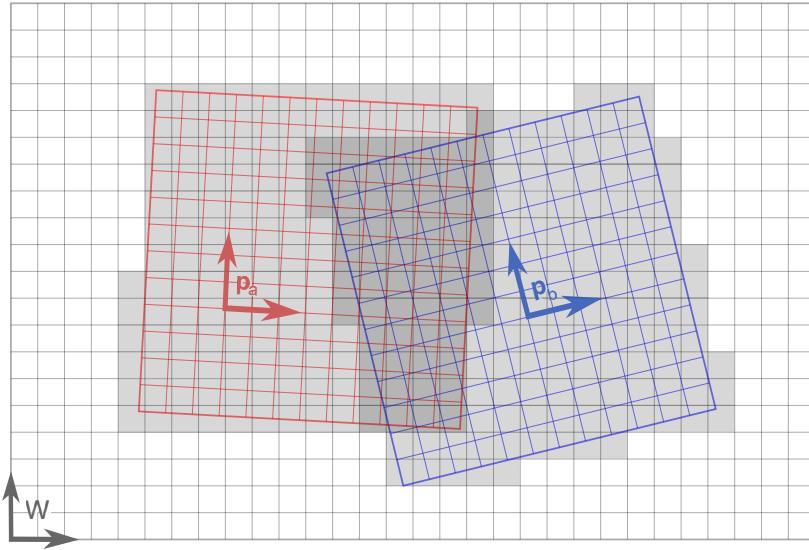


Fig. 4.5 Occupancy grid-map fusion illustrating two submap grid-maps with their origins p_a^W and p_b^W to be fused onto the global output grid-map W shown as grey grids.

An overlapping submap pair also initiates an occupancy grid-map fusion algorithm. To achieve the fusion described in Fig. 4.5, we use a GPU-based approach whereby the algorithm checks for each overlapping cell in the output grid-map and performs a transformation of that cell in its 2D coordinates into the submap's coordinate frame, subsequently fusing it into the output cell based on the corresponding submap cell value.

To optimise complex multimodal Gaussian constraints, we utilised a continuous mode blending optimisation technique that is based on nonlinear least-square approaches and exhibits convergence properties that are representative of the underlying multimodal constraint distributions.

4.3.6 SLAM Evaluation

For the evaluations described here, ten UGVs were deployed to explore an 80×40 m environment. The total elapsed time was 36 minutes [242]. The SLAM routine was completed with minimal user intervention whereby the UGVs autonomously explore the environment while displaying their progress in real-time onto the MR-SLAM system's GUI. This process follows our approach in [268] and is illustrated in series across Figs. 4.6 through 4.8 with timestamps on the upper right corner of the images in minutes and seconds; the total odometry across all UGVs is presented in meters at the lower right corner; UGVs are shown as dots with colour-matched lines showing their trajectories; pose graphs are green with dots, lines, and red triangles representing submap poses, submap constraints, and ground truth constraints, respectively.

The UGVs start at the southeast corner of the warehouse (see Fig. 4.6), where two teams split to explore the west and north sections, respectively. Both teams explore independently until a loop closure is evident, as illustrated in Fig. 4.7, where the first team is about to exit via the southwestern corner. The exploration forms a total closed path length of 230 m that is measured after the separation of the UGVs in the first room. The final result is presented in Fig. 4.8. This corresponds to about 70 constraints in the pose graph. Based on 88 samples collected across the surveyed area, the root-mean-square (RMS) error was calculated to be ≈ 0.27 m.

The 2D correlation between each submap in Fig. 4.9 is represented by each slice, with a ± 3 m translation variation along with x and y -axes at a fixed angular rotation. For example, the middle row represents rotations of -6° , -3° , 0° , 3° , and 6° , respectively. The 2D ellipses represent three-sigma covariance modes through a Gaussian mixture model that fits the maximum likelihood volume. Occlusions have reduced the overlapping area of occupied cells (black) between the submap pair, and the matcher's output is mostly dominated by an array of columns in the environment.

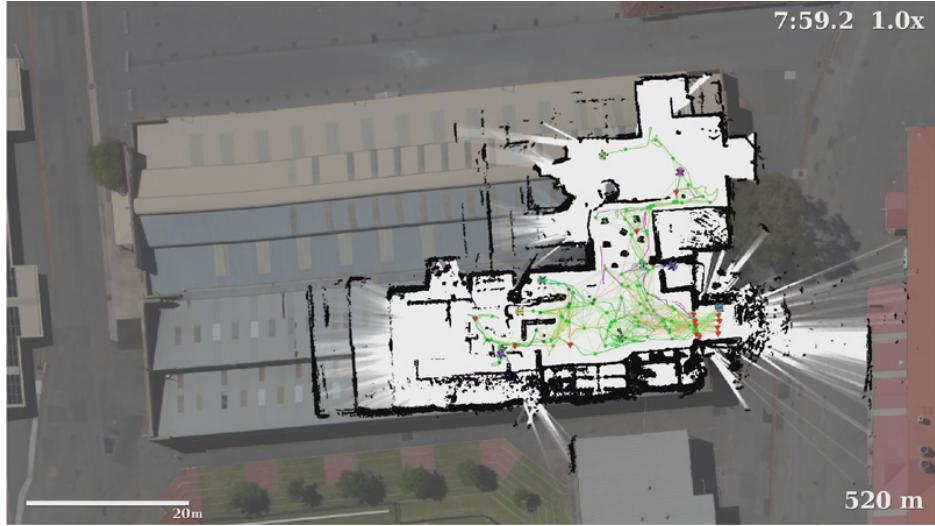


Fig. 4.6 SLAM output showing the UGVs starting at the southeastern corner of the environment.



Fig. 4.7 SLAM output showing loop closure between both teams, with the first team about to exit via the southwestern corner.



Fig. 4.8 Completed global grid-map with ground truth overlaid in magenta.

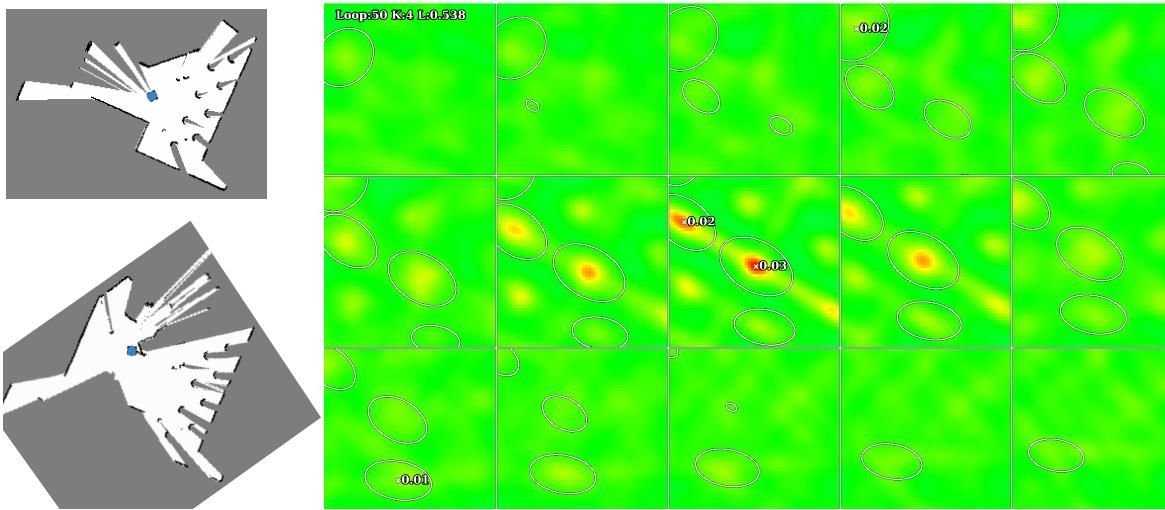


Fig. 4.9 Example of multimodal constraint output with perceptual aliasing. The correlation results for a test area with perceptual aliasing that was caused by repetitive geometry. It shows an overlapping submap pair (left) along with 15 slices through their constraint likelihood volume.

4.4 Visual Odometry

While some effort was made to circumvent wheel slip accumulation in Section 4.3, other works have demonstrated that visual odometry is often an effective solution to this problem [269]. In the case of the MRS, this can be incorporated with no hardware or sensor modifications. Taking advantage of the system’s software scalability, this was programmed on top of the existing software with minimal impact to the overall system.

4.4.1 Visual Odometry Method

As with most practical visual computing applications, the implementation of visual odometry in the real world comes with its own sets of challenges [270]. Environmental dynamics, including variations in seasons, light intensity, sensor occlusions, and motion blurs are capable of distorting visual odometry results that will in turn affect its accuracy. Our application will require that the algorithm is robust enough to withstand the environmental variations present in an urban outdoor environment.

Visual odometry works by tracking either features or appearances in the image frame [148, 149] with appearance-based approaches usually resulting in more accurate tracking but at the cost of computation complexity. The decentralised nature of the MRS dictates visual odometry to be performed on each robot’s computer as a separate subroutine to MR-SLAM,

thereby requiring a feature-based method to be implemented. While visual odometry can be applied across different image features and many feature-based methods do exist [149, 171], oriented FAST and rotated BRIEF (ORB) features [155] were found to be most compatible for the system due to the following:

- ORB features achieve a compromise between accuracy and system footprint. In other words, it is adequately accurate for the MRS application while having computation requirements that are low enough to be run on the individual robots.
- ORB features were proposed and tested in urban environments, whereby the structured appearance of urban environments enabled ORB features to be extracted effectively, especially on the KITTI benchmark suite [109].

Based on these rationales, an algorithm based on ORB-SLAM [168] was implemented as the visual odometry solution for this system. Once a camera has been corrected for radial distortions, the algorithm tracks ORB features across each frame to determine the displacement of each tracked pixel at every new frame, thereby localising the robot. It functions as a separate thread and routine on the onboard computer to minimise any interference to the other routines running on the robots.

As visual odometry is effectively a visual SLAM algorithm without loop closures, the MRS' implementation of ORB-SLAM is hence used purely for odometry; loop closures and SLAM are still managed by the LiDAR-based front-end, independent from visual odometry.

4.4.2 Visual Odometry Evaluation

Evaluations for visual odometry were performed on individual UGVs as the implementation was fully decentralised. To optimise for performance and to reduce redundancy, a modified version of ORB-SLAM2 was proposed whereby all subroutines related to visual SLAM, such as loop closure detection and mapping, are removed. By delegating all SLAM routines to the LiDAR-based front-end, this modification yielded a 120% increase in performance gain in terms of output frame-rate. Additionally, maps created through the LiDAR-based front-end delivers more accurate point cloud measurements as compared to our monocular camera setup, and a LiDAR-based map requires lower computation and storage requirements than a vision-based solution.

ORB-SLAM2 achieves visual odometry by tracking ORB features, as shown in Fig 4.10. This evaluation was performed in an outdoor environment with unconstrained lighting conditions using a calibrated monocular camera. Tests were carried out while driving along a 220 m path while generating its trajectory as shown in Fig. 4.11, where it is indicated in blue;

the black dots represent previously tracked ORB features, whereas the red dots represent the features that are currently tracked.



Fig. 4.10 ORB features tracked by ORB-SLAM2 as shown in bounding boxes.

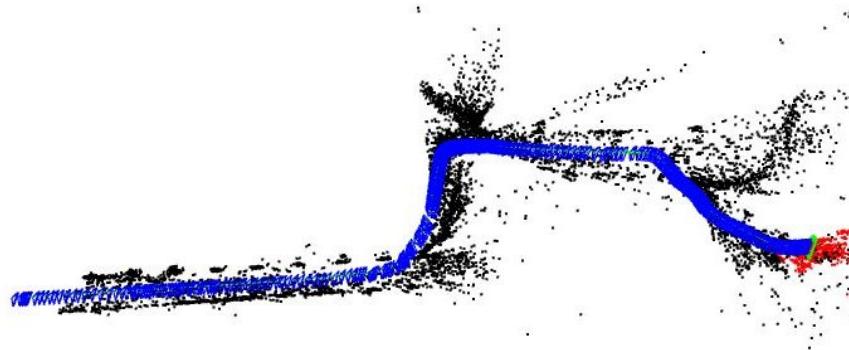


Fig. 4.11 Trajectory measured using ORB-SLAM2 for dead reckoning (blue), previously tracked ORB features (black) and presently tracked ORB features (red).

4.5 Semantic Segmentation

The incorporation of semantic segmentation into the MRS enables navigation to be supplemented with scene understanding and object classification. Semantic segmentation is a deep

learning process that classifies each pixel in an image frame according to the object class it belongs to. This is especially useful in complex environments with multiple objects, with little uniformity in pose, features, and illumination.

4.5.1 Semantic Segmentation Method

For the semantic segmentation application of the MRS, SegNet [106] was selected based on its high compatibility and ease of implementation. The architecture of SegNet uses a convolution encoder and decoder setup that classifies objects from one of the following classes: sky, building, column-pole, road-marking, road, pavement, tree, sign-symbol, fence, vehicle, pedestrian and bicyclist; with a class average classification accuracy of 65.9% [106]. This MRS uses SegNet whereby pedestrians, vehicles, buildings, vegetation, and pathways are classified as illustrated in Fig. 4.12, and are subsequently classified into static and dynamic objects.

Static objects are stationary (with stationary positions), while dynamic objects are moving (with time varying positions). It is important for an MR-SLAM system to differentiate static and dynamic objects to devise proper navigational reactions to the environment. For example, static objects such as buildings and vegetation are permanent placements in the environment; these objects will be mapped by the MR-SLAM algorithm as part of the environment. Conversely, dynamic objects such as pedestrians and vehicles are in motion or are temporary placements in the environment; these will not be mapped by the MR-SLAM algorithm. Overall, this process of differentiating object types will ultimately result in higher mapping accuracy, especially when the ground truth does not contain dynamic objects.

The recognition of dynamic objects also enables the system to estimate the motion of a specific moving object. In other words, by segregating moving vehicles or pedestrians within an image frame, the LiDAR can then be utilised to estimate the motion and trajectory of the said object. This enables the robot to actively perform obstacle avoidance according to its motion, which can be implemented by comparing the robot's current speed against the LiDAR measurements on the classified dynamic object based on the image frame.

4.5.2 Semantic Segmentation Evaluation

Like visual odometry, semantic segmentation was also implemented in a decentralised approach onto individual UGVs. A Caffe [113] implementation of SegNet is installed onto the individual UGVs, which enables pixelwise object classification that corresponds to LiDAR measurements at that time instance, which can be any of 12 classifiable classes. We subsequently separate these classes into static and dynamic classes. For example, bicycles,

pedestrians, and vehicles are dynamic, whereas buildings, fences, pavement, poles, road, road markings, road signs, vegetation, and sky are static. By matching the position of the classified pixel at the x -axis against that of the LiDAR on a fixed y -plane, dynamic objects can therefore be segregated and tracked using the LiDAR for motion detection. Objects in motion will be compared against the trajectory of the UGV to ensure that there is no impending collision. These dynamic objects will also be ignored as part of the SLAM routine so that it does not become mapped as part of the environment.

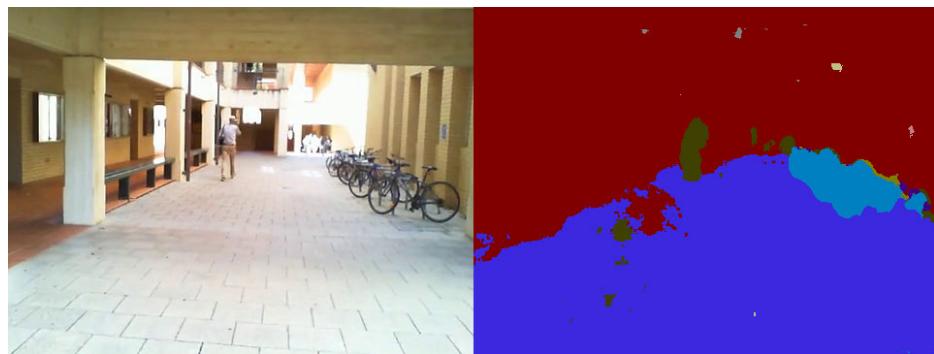


Fig. 4.12 SegNet output showing segmented pedestrian (olive), bicycles (light blue), pathway (blue) and building (red).

Fig. 4.12 was captured while navigating along the path as described in Section 4.4.2. The parked bicycles on the right side and the pedestrians in the distance were properly segmented as dynamic objects, and the building and pavement as static objects. Several false detections are present due to variations in lighting and image quality, which accounts for 2.83% of the total pixels classified on the pavement region.



Fig. 4.13 SegNet output from a parking area showing segmented road regions (purple), road markings (orange), and poles (yellow).

Fig. 4.13 was also captured on the same path. While driving on roads, the road region and markings are properly classified along with the electric poles, vehicles, and pedestrians.

Uniform lighting resulted in accurate classification accuracy with 0.69% of all pixels falsely classified.

4.6 Conclusion

In this chapter, we have presented a decentralised multi-robot system for SLAM in urban outdoor environments together with visual odometry and semantic segmentation techniques. The vision-based methods can be used as an alternative to LiDAR-based localisation and object classification and may lead an overall cheaper and more powerful environmental perception system. We have presented an on-line, distributed, and decentralised MR-SLAM system that has proven to be resilient against environmental dynamics such as variations in lighting, terrain, pose, and moving objects. Evaluation results have confirmed the feasibility of using visual odometry as a viable solution to the odometry problem in an MRS, while semantic segmentation is a robust solution to object classification and scene understanding. Practical applications of this system can include search-and-rescue or reconnaissance missions in uncharted or hazardous environments, where detailed maps can be built quickly and accurately using a swarm of robots that are easily deployable, while being robust enough to cater to changes in the environment and hardware setup.

Chapter 5

Implementation of Semantic Segmentation for Road and Lane Detection on an Autonomous Ground Vehicle with LiDAR

While current implementations of LiDAR-based autonomous driving systems are capable of road following and obstacle avoidance, they are still unable to detect road lane markings, which is required for lane keeping during autonomous driving sequences. In this paper, we present an implementation of semantic image segmentation to enhance a LiDAR-based autonomous ground vehicle for road and lane marking detection, in addition to object perception and classification. To achieve this, we installed and calibrated a low-cost monocular camera onto a LiDAR-fitted Formula-SAE Electric car as our test bench. Tests were performed first on video recordings of local roads to verify the feasibility of semantic segmentation, and then on the Formula-SAE car with LiDAR readings. Results from semantic segmentation confirmed that the road areas in each video frame were properly segmented, and that road edges and lane markers can be classified. By combining this information with LiDAR measurements for road edges and obstacles, distance measurements for each segmented object can be obtained, thereby allowing the vehicle to be programmed to drive autonomously within the road lanes and away from road edges.

5.1 Introduction

The Renewable Energy Vehicle (REV) Project at the University of Western Australia conducts research into electric vehicles, vehicle automation and autonomous driving systems. Recent projects include the development of an Autonomous Formula-SAE Electric car [271]. This vehicle is an open-wheeled, electric drive race car, with electronic drive-by-wire and electromechanical brake/steering actuation. The vehicle serves as a compact, flexible test-bed for sensor testing and the development of autonomous driving algorithms.

Prior research has been conducted on road and road edge detection through optical systems [272], radar [273] as well as using Light Density and Ranging (LiDAR) sensors such as in the winning entry in the 2007 DARPA Urban Challenge [274]. The methodology described in [275] utilises a feature-extraction algorithm while other algorithms such as [276] rely on the presence of curbs and seek to identify and track curbs as features in the LiDAR data. More recently, there has been an increase in the use of cameras to achieve this [81], giving rise to visual road detection. Methodologies to achieve this include feature extraction and classification [81], horizon and vanishing point detections [44], and artificial neural networks (ANNs) [45].

The problem of path-finding can be described as: “Given a start state, a goal state, a representation of the robot and a representation of the world, find a collision-free path that connects the start with the goal satisfying the system constraints” [277]. In mobile robotics, a proven method to obtain the requisite “representation of the world” is via the use of LiDAR data to generate a virtual map in real-time both as the sole sensor [278] and in conjunction with data from additional sensors [279]. Similar LiDAR based map building approaches have been shown to be suitable for outdoor terrain [280]. These generated maps vary from simple two-dimensional maps suitable for basic path planning consisting of traversable regions, obstacles and unexplored regions [281] to more complex three-dimensional maps from which sophisticated cost maps are generated [282]. A more detailed map can be built by supplementing the camera in addition to LiDAR. These additional details can include a combination of vehicle detection and classification [283], road sign recognition [284], and scene recognition [285].

Visual cameras and LiDAR are often incorporated in autonomous driving systems. Works that combines LiDAR and camera sensors for autonomous driving include the approach from Zhang, Clarke and Knoll [283], where they have proposed the fusion of LiDAR and the camera as a compromise for each sensor’s drawbacks, with LiDAR providing range information, and the camera identifies objects and scenes. The authors achieved low false alarm rates and a high detection rate for vehicles in urban environments. A similar fusion of multiple LiDAR, radar, and camera sensors to achieve object detection and tracking was

proposed by Cho et al. [286]. By tracking pedestrians and vehicles, the system could detect and track vehicles from 150 m away, and pedestrians and cyclists within a 20 m radius. To the best of our knowledge, works that incorporate semantic segmentation onto a LiDAR-based autonomous ground vehicle has not been established at the time of writing.

Our work is an enhancement to the work done by Drage, Churack and Bräunl [287], where we have proposed a LiDAR-based road edge detection approach on the same vehicle. Our algorithm could detect road curbs and edges by measuring the differences in surface smoothness, which in turn allows the positioning of road edges and curbs.

5.2 Implementation

This section describes the addition of visual perception to the LiDAR-based autonomous SAE car as described in [287], which includes sections that describe our testing environment, and its applicable procedures to achieve visual autonomous driving. By mounting a monocular camera onto the chassis of the vehicle, above the LiDAR (see Fig. 5.1), road recognition and obstacle detection are achieved using semantic segmentation. This camera supplements the LiDAR, where the LiDAR is responsible for providing distance measurements for objects and road edges detected by the camera. Semantic segmentation was achieved using SegNet [106], a convolutional neural network (CNN) architecture for semantic segmentation that is often used for road scenes. Its architecture uses an encoder-decoder network that is followed by a pixelwise classification layer, where the encoder and decoder networks consist 13 convolutional layers each. The Caffe [113] implementation of SegNet is used for this project. To interface the sensors for autonomous driving, SegNet is installed onto an Nvidia Jetson TX1 [117], and the LiDAR interfaces directly to a Raspberry Pi 3 [288], which drives a control system. A GPS module and an inertial measurement unit (IMU) module also connects to the Raspberry Pi 3 for positioning and localisation.

The LiDAR system used in this project consists of an ibeo Lux automotive LiDAR with specifications as shown in Table 5.1. This sensor utilises reflected infra-red light to measure distance (via time-of-flight) and can build a 3D point cloud by scanning horizontally in four vertical layers. The ibeo sensor has sophisticated internal data processing functionality including object detection and classification. Data is delivered using TCP/IP over an Ethernet connection and includes scan data in polar coordinates and object data in x - y coordinates referenced to the sensor.



Fig. 5.1 The camera is mounted above the LiDAR system from [287], beside the IMU.

Table 5.1 LiDAR Characteristics

Specification	Value
Technology	Time of flight (output of distance and echo pulse width)
Range	200 m
Field of View (Horiz / Vert)	85° / 3.2°
Layers	4
Echo Detection	3 measurements per pulse
Update Rate	Up to 50Hz
Accuracy	10cm

The following subsections describe the process of achieving visual autonomous driving for our project using semantic segmentation with respect to its application environment and its driving sequences.

5.2.1 Application Environment

SegNet was tested within the grounds of the University of Western Australia (UWA), which is the same location that the LiDAR system was tested in [287]. The roads within UWA offers a similar drive environment to standard suburban roads. These single carriageway roads are of low traffic density, with views of pedestrians, faculty buildings, and vegetation for SegNet to recognise and segment. As a feasibility test, we also tested SegNet off a car-mounted dashcam recording while driving on local roads.

This application environment was selected to test the suitability of using SegNet for autonomous driving locally and to gauge the visual autonomous navigation performance of the vehicle. To achieve a successful autonomous drive using SegNet on the vehicle, road edges and lane markings must be properly recognised, before the application can be subsequently expanded onto a road-licensed vehicle.

It should be noted that our initial implementation uses the trained dataset from the University of Cambridge, CamVid [106]. This dataset was recorded in the City of Cambridge, England. Like most British cities, Cambridge's roads are often narrow, with dense buildings by the side. There is also a large pedestrian population due to it being an academic city. Comparatively, roads in Perth are generally wider, with a sparser build-up density than Cambridge. Its low population density means that there are fewer road pedestrians as compared to Cambridge. The ground terrain around Perth is mostly flat, with long sunshine hours. This means that one can generally expect excellent road visibility on Western Australian roads on most days. However, during poor visibility and night time drives, suburban roads around Perth are generally poorly lit, which may affect road segmentation accuracy. By testing this dataset, we will subsequently contemplate on the need to record and use a dataset from Perth for more accurate segmentation results.

5.2.2 Autonomous Driving Procedures

To use SegNet's output for autonomous driving, we assume that the car begins with a position in the middle of the road lane and that the road incline is flat. By mounting the camera at a fixed position on the car, the central driving position of the car can be obtained, along with distance measurements from the road edges to the left and right sides of the car. With the camera, this is done by identifying and segregating a fixed trapezoidal image region that encapsulates the road segment, which is then transformed into a birds-eye view perspective to obtain vehicle's position with respect to the road's centre. By scaling the road width according to the Australian standards of 3.3–3.5 m, along with the detected road edges and/or lane markings on SegNet's output, the distance from the vehicle's centre to the left and right

road edges/lanes can be obtained as Fig 5.2, with its confidence value determined by the successful detection of road edges or lane markings. From these three sections distance thresholds for the left and right edges or lane markers distances for the car to autonomously centre itself on the road while driving. We call this road centring. In the event where lane markers are not found, road edges will be used instead.

To perform road centring, the car must steer itself in the opposite direction when it crosses the distance threshold to either the left or right road edge/lane markers. The distance from the car to the road edges or lane markers are constantly analysed. If the car is too close to the edge or lane marker, the road centring algorithm will then send commands to the drive system to steer away from the edges with fine adjustments, until the car is cleared from the distance threshold.

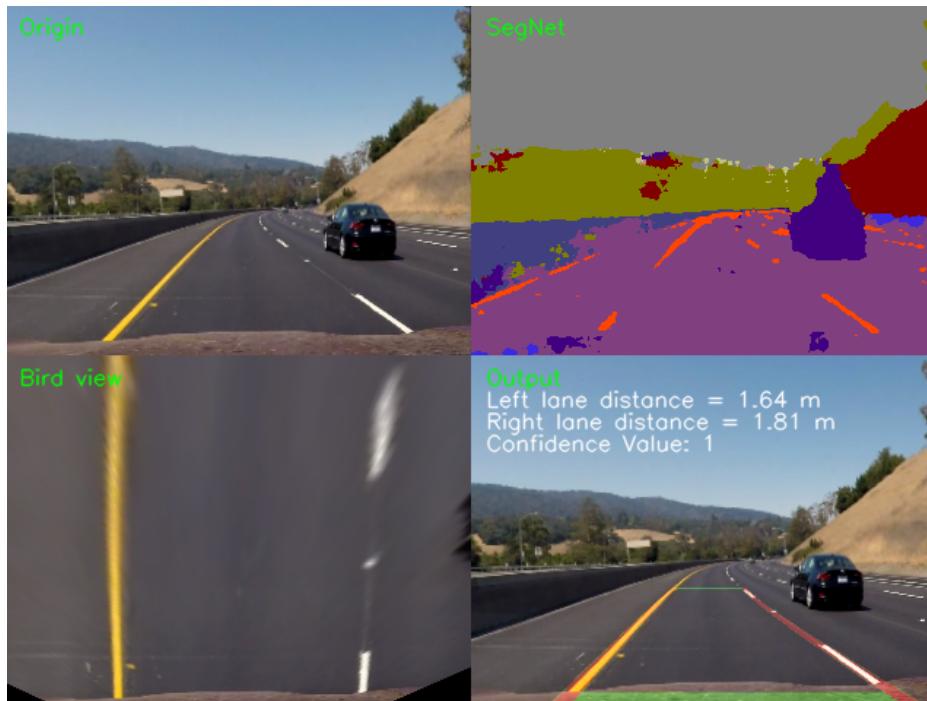


Fig. 5.2 Lane distance measurements with SegNet’s output on Udacity’s Self-Driving Car Nanodegree recording [135] as the input in “Origin”.

5.3 Testing and Evaluations

5.3.1 Methodology

Testing begins with the calibration of the camera, where distance measurements in the real world will be represented in pixel ratios on SegNet’s output. Here, we calibrated a Microsoft

LifeCam HD-3000 camera. This was done by measuring the distances between road bollards in front of the parked vehicle on the road as illustrated in Fig. 5.3.



Fig. 5.3 Photo illustrating the bollards' position with reference to the car in the centre.

The bollards are placed a three-row formation to allow distance measurements from two-point distances on the frame. The first row of bollards on the car establishes the starting distance, with the centre bollard measuring the distance from the camera to the front of the car, while ensuring that the camera is pointed to the centre of the car. All distances are measured from the base at the centre of each bollard. Subsequently, a topological representation of the measurements can be illustrated in Fig. 5.4, which is then represented again in the camera frame in Fig. 5.5. These measurements are verified with the LiDAR plot at that position as illustrated in Fig. 5.6, where the bollards (represented as dot plots) are clearly present around the 2 m, 8 m and 10 m mark on the y-axis. From Fig. 5.5, each image pixel was calculated to represent 17 mm and 23 mm when measured from 8.5 m and 11.6 m respectively from the camera, and that a level road will converge at around 41° on the camera frame.

The LiDAR readings complement SegNet's output for road edge detection, whereby we use our Kalman Filtered Linear Regression Model as described in [287]. Our algorithm minimises the square residuals between the fit line y and the data (x_i, y_i) , where the most suitable data line will be obtained for a given data set, and its success measured by the product-moment correlation coefficient r . The slope b and r values are given as (5.1).

$$y = (\bar{y} - b\bar{x}) + bx \quad (5.1a)$$

$$r = \frac{s_{xy}}{s_x s_y} \quad (5.1b)$$

where:

$$b = \frac{s_{xy}^2}{s_x} \quad (5.2a)$$

$$s_{xy} = \frac{\sum_{i=1}^n x_i y_i}{n} - \bar{x}\bar{y} \quad (5.2b)$$

$$s_x^2 = \frac{\sum_{i=1}^n x_i^2}{n} - \bar{x}^2 \quad (5.2c)$$

$$s_y^2 = \frac{\sum_{i=1}^n y_i^2}{n} - \bar{y}^2 \quad (5.2d)$$

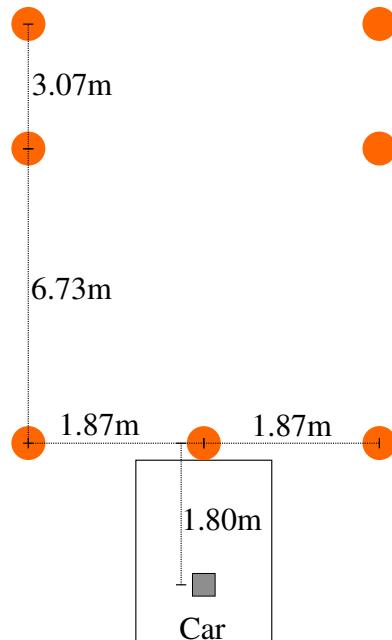


Fig. 5.4 The topological distance between bollards (dots) and the camera on the car (shaded square).

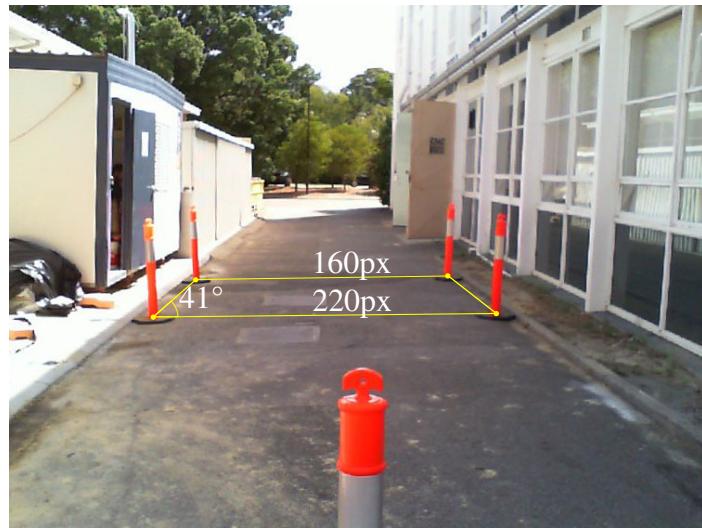


Fig. 5.5 Frame captured from the vehicle's camera for distance calibration.

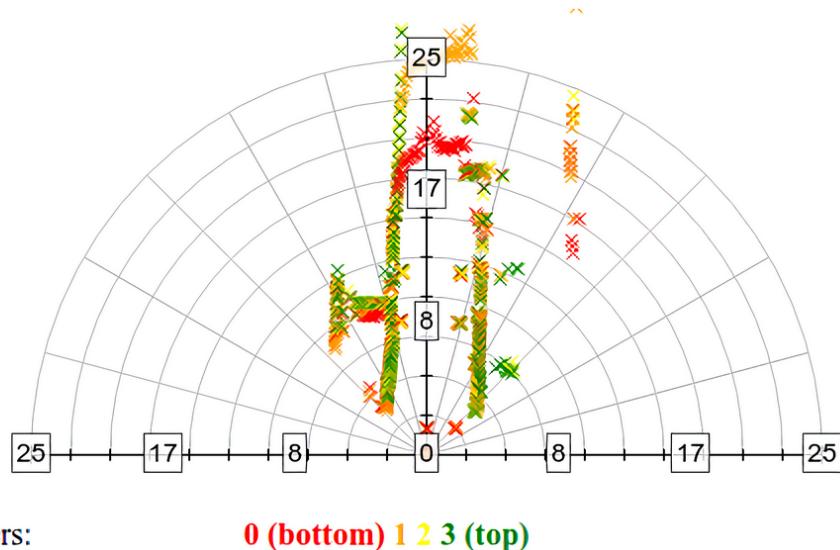


Fig. 5.6 LiDAR plot showing the detected road edges and the bollards' positions from the car where Fig. 5.5 was captured. The graph axes measure distances in metres.

With a calibrated camera and LiDAR, the SAE car was driven around the application environment while the camera is recording. The recorded camera footage was used as an input for SegNet. For testing purposes, these footages were processed off-line with SegNet running off an Nvidia GTX Titan X GPU with a 480 by 360-pixel resolution, which resulted in a segmented image output at a consistent 10 frames per second (FPS) sampled at each video frame. This framerate is consistent with the visual autonomous driving results published by Nvidia in [129], making it adequate for autonomous driving. Further work is required for

real-time on-line processing on the Jetson TX1. Likewise, LiDAR plots are also recorded during the duration of the drive, where timestamps are used to synchronise outputs.

5.3.2 Results and Discussions

Results from semantic segmentation are presented first on open roads with a dash cam recording, and then on the UWA campus ground with the SAE car. Image results in this section are presented with the left image showing segmentation input, and the right image showing its output through SegNet. In the case of false detections, they will be measured in their pixel accuracy (PA). This is done by finding the percentages of road regions, which is done by counting the number of falsely detected pixels p_{ii} against the total number of pixels k in the road region p_{ij} for that image frame:

$$\text{PA} = \frac{\sum_{i=0}^k p_{ii}}{\sum_{i=0}^k \sum_{j=0}^k p_{ij}} \quad (5.3)$$

Segmentation results from the open road testing from the dashcam footage yielded consistently favourable results, as shown in Fig. 5.7. Recordings were captured on a clear day, driving on a low traffic dual carriageway. The road region is accurately segmented with negligible false detections. In addition to the road and its markings, SegNet can detect and segment the speed limit sign and the right turn sign ahead. All vehicles and vegetation were also properly detected and segmented. The minor false detections in the sky region are due to the CamVid dataset being trained on a cloudy day, but this should not affect road detection accuracy in any way. This result confirms the feasibility that a CamVid-trained SegNet can be implemented for autonomous driving in the Perth metro area.



Fig. 5.7 SegNet’s input (left) and output (right) for a dual carriageway in the Perth metro area.

Tests were subsequently performed on the SAE car for a vision-LiDAR-based implementation. Here, runs on campus grounds were recorded using the camera while manually driven on the SAE car following a predetermined route on campus. The car traversed across roads and pavements and the segmentation results are as follows.



Fig. 5.8 Segmentation results from a parking area on campus grounds.

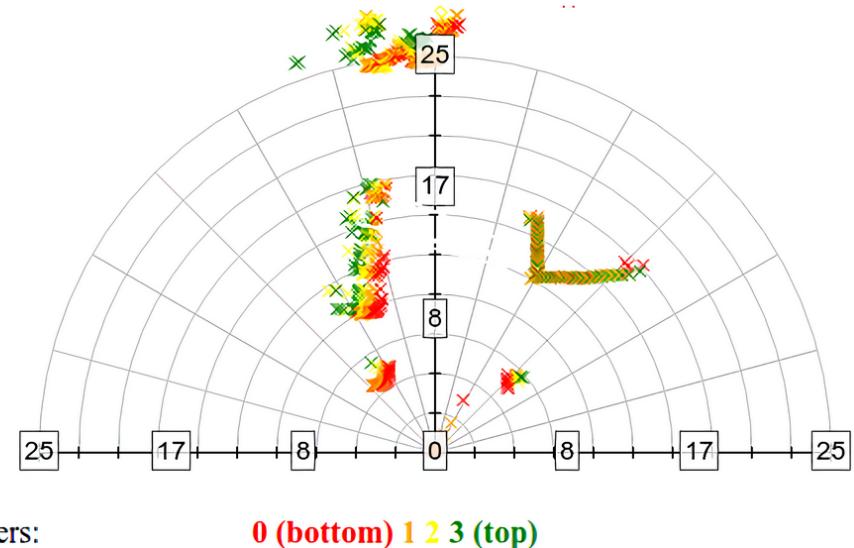


Fig. 5.9 LiDAR plot showing the detected parked vehicles at the position where Fig. 5.8 was captured. The graph axes measure distances in metres.

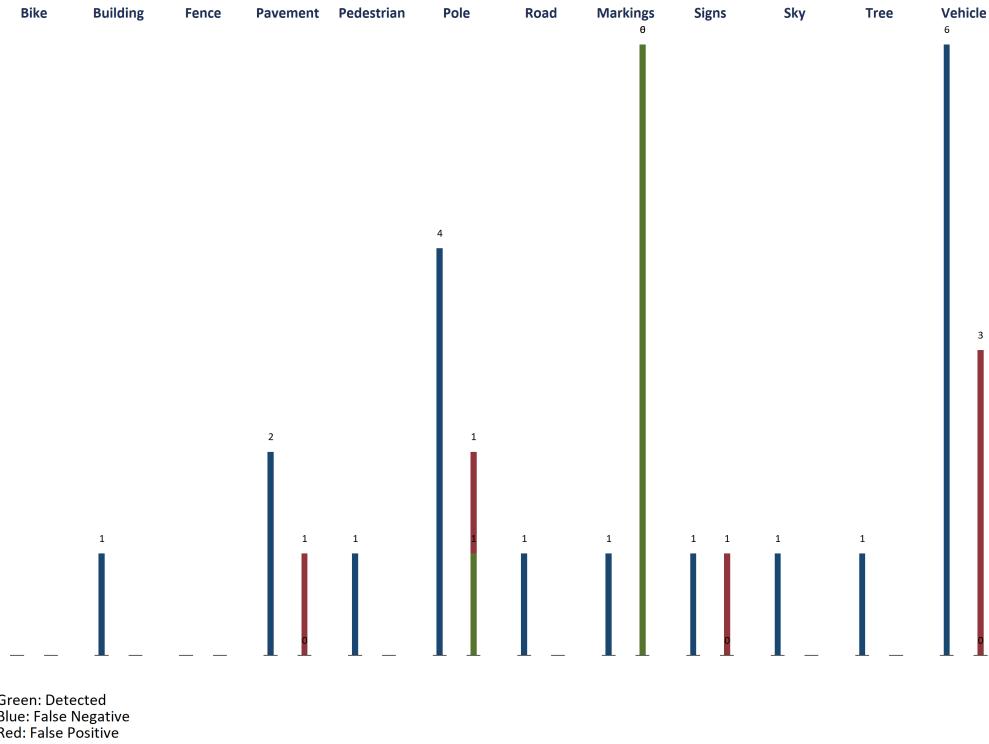


Fig. 5.10 The number of detected objects along their errors in detections according to the objects.

The test run began with a drive through the car park. SegNet's output in Fig. 5.8 shows that the image was segmented with good accuracy. With the exception of some minor (0.69%) false detections on the car's shadows on the left side, the road, parking lane, and pavements were properly segmented, along with the pedestrian and vehicles. With the LiDAR actively measuring the distance from the parked vehicles to the SAE car (see Fig. 5.9). Here, we adopt the Linear Regression model that we described in Section 5.3.1, which plots the road edge position from the parked vehicles so that a fixed distance can be kept between the autonomous cars and the parked vehicles. We have also counted the number of detected objects along with their positive and negative false detections, which are plotted according to their detection/error pairs in Fig. 5.10, whereby the labelled number on each bar indicated the number of correctly identified objects, if present.

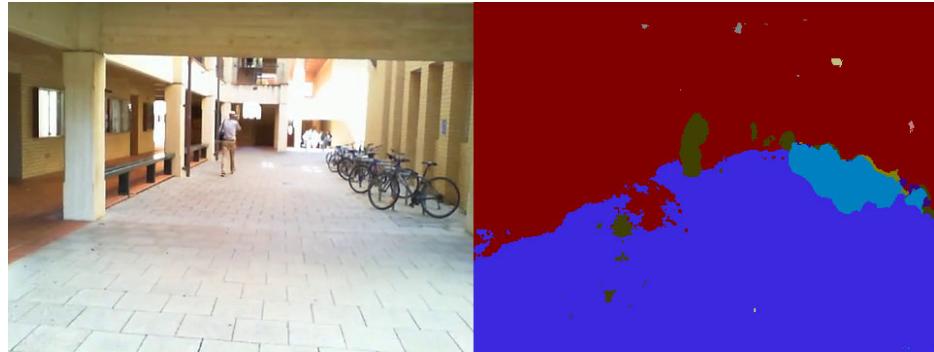


Fig. 5.11 Segmentation results on pavement between faculty buildings.

From the parking area, the car drives onto the pavement between faculty buildings. From Fig. 5.11, SegNet could discern pavements from roads as the grounds are now coloured blue. In addition, it was also able to detect the bicycles parked towards the right, and the pedestrians in the distance. False detections are present on the left side of the pavement, where it is incorrectly detected as buildings and pedestrians due to uneven lighting, accounting for 2.83% of the total pavement region.



Fig. 5.12 Segmentation results at a road junction.

Fig. 5.12 was captured when the car was stopping at a suburban road junction beside the campus. The segmentation output from this figure shows that while the objects in the distance were properly segmented, some parts at the bottom of the image were incorrectly classified as pavements and buildings, which makes up 16.19% of the road region. This was partly due to the clear weather resulting in a high brightness recording, while SegNet was expecting a darker surface to classify roads.



Fig. 5.13 Segmentation results on a road with pronounced shadows.

Fig. 5.13 was captured while the car was driving on a shadowed road on campus. The shadows on the road introduced a high contrast region, and that the bottom portion of the road was overexposed, resulting in a false classification of 13.93% and undetected lane markers. However, the overtaking vehicle, vegetation, road sign, and building were correctly segmented. When presented with a false detection on the road as with Figure 7, results from the LiDAR road edge detection system will compensate the regions of false detection, as the road and road edges were properly measured by the LiDAR system.

Table 5.3 The number of road region pixels and its false detections pixels in that region for each figure along with their error percentages.

Fig.	Road Pixels	False Classifications	Error
5.7	41648	16	0.04%
5.8	86561	600	0.69%
5.11	76731	2168	2.83%
5.12	108684	17595	16.19%
5.13	83831	11679	13.93%

The false detection rates for each of the example figures is summarised as Table 5.3, which also tabulates the number of road pixels in each detected road segments for each figure, along with the number of falsely classified pixels within that road segment. From these numbers, the detection error is calculated as a percentage that corresponds to the area of each figure's road segments. The false detection rate is at its highest in Fig. 5.12, where the road region encompasses most of the frame, the unevenness in road surface and lighting is the

main contributor to this error rate. Conversely, Fig. 5.7 records the lowest false detection rate its road segment, as the road area was well-defined, and the frame was properly exposed.

5.4 Conclusion

We have presented a semantic segmentation-based visual navigation approach for autonomous ground vehicles. This approach improves on existing LiDAR-based vehicles to introduce object recognition and classification while driving. SegNet adequately performs semantic segmentation to recognise roads and lane markers, which in turn allows the vehicle to maintain a safe distance from the road and lane edges in addition to LiDAR measurements. We have also shown that the segmentation results from SegNet on the CamVid dataset is satisfactory on Perth metro roads. With a calibrated camera, visual autonomous driving can be achieved using real-time semantic segmentation. Future works will focus on the complete on-line implementation of SegNet on the SAE car for real-time visual autonomous driving.

Chapter 6

A Modular Software Framework for Autonomous Vehicles

Software frameworks for autonomous vehicles are required to interface and process data from several different sensors on board the vehicle, in addition to performing navigational processes such as path planning and lane keeping. These can include a combination of cameras, LiDARs, GPS, IMU, and odometric sensors to achieve positioning and localisation for the vehicle and can be challenging to integrate. In this paper, we present a unified software framework that combines sensor and navigational processing for autonomous driving. Our framework is modular and scalable whereby the use of protocol buffers enables segregating each sensor and navigation subroutine individual classes, which can then be independently modified or tested. It is redesigned to replace the existing software on our Formula SAE vehicle, which we use for testing autonomous driving. Our testing results verify the suitability of our framework to be used for fully autonomous drives.

6.1 Introduction

UWA’s Renewable Energy Vehicle Project (REV) operates two autonomous vehicles, a BMW X5 and a student-built Formula SAE-Electric car. Formula SAE [289] is a long-running annual competition organised by the Society of Automotive Engineers with competition events worldwide. The design competition includes petrol cars, as well as the SAE-Electric class which includes ours with two electric motors driving each of the two rear wheels via independent controllers. We have incorporated full drive-by-wire control of the throttle, steering and the hydraulic braking system. The use of a Formula-SAE car provides several advantages for such a project; the car is mechanically simple, Formula-SAE cars with similar

designs are common at universities worldwide and the size of the car makes testing accessible without forgoing race car vehicle dynamics. Furthermore, the use of an electric vehicle makes the project significantly more practical for student work as the risks and environmental issues associated with petrol-engine cars are eliminated and the systems installed in this project can take advantage of the large amount of electrical energy already available on the vehicle.

For the driverless FSAE project, our goal was to be able to autonomously drive a vehicle around a race track. This is being achieved by placing waypoints along the ideal driving line, as well as "fence points" to lock out non-driving areas. Maps can either be recorded by human or remote-controlled driving or specified through a Google Maps driven web-interface. During autonomous driving, a laser scanner and camera are used for detection of road edges as well as any obstacles on the track. Safety systems are essential for a driverless system, as the car weighs more than 250 kg and is capable of driving at a speed of 80 km/h. Both the low-level drive-by-wire, as well as high-level navigation system have independent safety systems built in. These include remote intervention, remote heartbeat and remote emergency stopping, which are implemented through a fail-safe wireless link to a base station as well as through hard-wired buttons on the car itself.

Our motivation for designing and presenting this framework is to improve upon the existing autonomous drive software on our SAE vehicle that is documented in [290]. This software utilises a web-based user interface (UI) that displays via a touch screen mounted in the vehicle's cockpit. Our proposed framework utilises this existing UI with a revamped backend as described in Section 6.2. It was noted that the existing software had a heavy reliance on a central Control module, which required all the sensors and their submodules to run to function. These submodules were programmed throughout the years with different programming languages and redundancies, which made integration difficult. Our proposed framework presents a streamlined approach whereby each module will be programmed with a C++ interface that communicates with either a path planner or a drive control system. This system also benefits from additional features including visualisation data playback (online or offline) and a web-based user interface. As a customised framework for our project, this also alleviates the reliance on node-based solutions such as ROS, which usually requires a perpetually running Master node to function and allows higher reliability when the individual components are integrated.

Additionally, this approach presents a long-term advantage whereby our framework is made fully open and contributable by students and enthusiasts looking to implement our framework onto their custom-fabricated vehicles. When compared against other autonomous driving frameworks such as Apollo [291] and Autoware [292] that mostly target commer-

cial vehicles and requiring expensive hardware, our approach leverages on hardware and fabrication methods that are more accessible in cost.

6.2 Autonomous Driving Framework

This section introduces our proposed software framework for the SAE vehicle, with the software architecture diagram as illustrated in Fig. 6.1. The software architecture of the vehicle utilises a modular design to allow for continuous parallel development on each of the sensors, the path planning algorithm, and the control policy. Input sensors are comprised of the LiDAR, camera, GPS, IMU and four wheel speed sensors, which are required to function simultaneously in order for the car to drive autonomously. The LiDAR, camera, GPS and IMU each have their own receiver class on the Jetson TX1, which takes the input from the sensors and processes the data. Additionally, rotary-encoder odometry is performed using an Arduino microcontroller connected to each of the wheels' encoder. This information is received in the high-level software by another receiver class which processes the data to produce wheel speed measurements. Relevant data for path planning and localisation is encoded in protobuf [293] format and then passed to either the path-planning program or the control program, which performs localisation as well as some communication and logging utilities. The control program also communicates with the web GUI, providing a visualisation of the data and allows the user to start and stop autonomous driving as well as the safety trip monitor and the controller, thereby introducing a high-level interface for driving the car. The control program, once it has communicated with the path-planner to combine localisation data with a set of future way points, will transmit driving instructions to the controller. This in turn transmits them to a low-level microcontroller. Not pictured in the diagram is the fusion of data from sources such as the GPS and IMU, and IMU and LiDAR in order to improve the accuracy of localisation. Detailed explanations of the sensors and classes are covered in Sections 6.2.1 through 6.2.8.

6.2.1 Path Planner

We use a real-time capable path-planning algorithm based on [294]. Given a set of pre-recorded or pre-defined waypoints along a track, the planner will generate a optimised path through all way points, which serves as a base frame for trajectory generation. During operation, the algorithm evaluates a variety of possible trajectories in the configurations space of the vehicle using rapidly-exploring random trees (RRTs) [277]. Those intermediate trajectories are generated along a curvilinear coordinate system, along with the base frame.

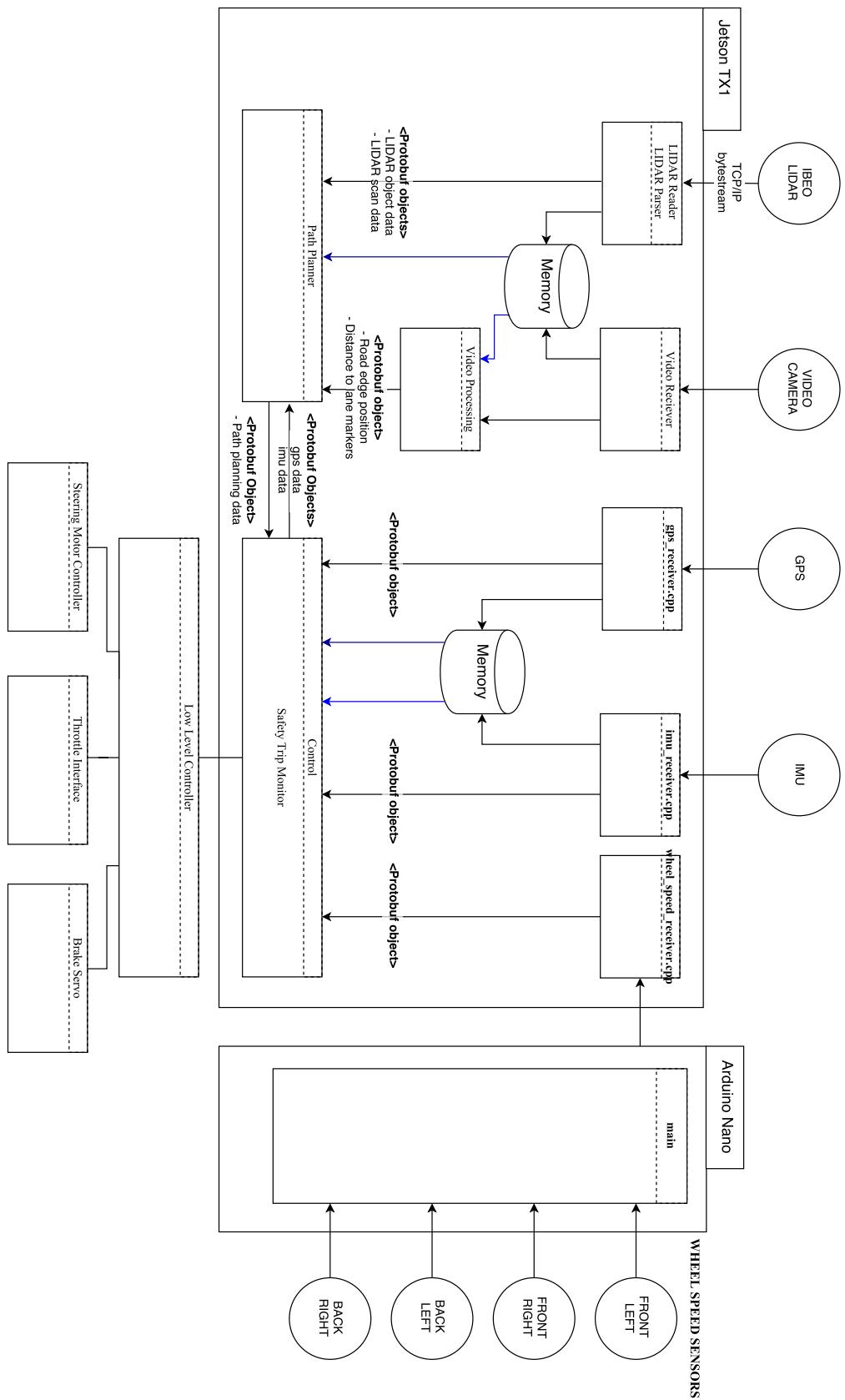


Fig. 6.1 The software architecture diagram of our proposed framework.

Each possible trajectory is checked for collisions with obstacles. A collision free path is then picked, utilizing a cost-function, that enables us to influence the driving style of the vehicle. The algorithms are implemented in C++14 and rely heavily on the C++ source libraries Boost and Eigen3.

We use an arc-length parametrised cubic B-spline $P_b(s)$ [295] to generate a base frame for the curvilinear coordinate system, which can be described as a non-linear transformation of the parameter domain on the four waypoints a to d , parametrised by s .

$$P_b(s) = \begin{cases} x_b(s) &= a_{x,i}(s - s_i)^3 + b_{x,i}(s - s_i)^2 + c_{x,i}(s - s_i) + d_{x,i} \\ y_b(s) &= a_{y,i}(s - s_i)^3 + b_{y,i}(s - s_i)^2 + c_{y,i}(s - s_i) + d_{y,i} \end{cases} \quad (6.1)$$

The curvature κ_b can be calculated from the first and second derivatives of $P_b(s)$

$$\kappa_b = \frac{x'_b y''_b - x''_b y'_b}{(x'^2_b - y'^2_b)^{\frac{3}{2}}} \quad (6.2)$$

where:

$$\frac{dx_b}{ds} = x'_b = \cos \theta_b \quad (6.3a)$$

$$\frac{dy_b}{ds} = y'_b = \sin \theta_b \quad (6.3b)$$

The curvature of a cubic spine is continuous in all sections. For this reason, a parametric cubic B-spline is adopted for the base frame. Since the position of the vehicle is provided in Cartesian-space, we need to find a transform those coordinates to a curvilinear representation, of which the B-spline provides the base. This is equal to finding the closest point to the vehicle on the base frame, by minimising the Euclidean distance between the position of the vehicle and the cubic B-spline.

$$\min_s d(s) = \sqrt{[x_v - x_b(s)]^2 + [(y_v - y_b(s))]^2} \quad (6.4)$$

We chose Brent's method [296] to find the minimum. Another suitable and stable algorithm is provided in [297]. From the lateral distance to the base frame and the base frame we construct the curvilinear coordinate system in which we generate a number n of paths as cubic polynomials. Each polynomial is defined by a lateral offset $q(s)$ and a curvature κ , to cover a broad section of the configuration space of the vehicle. n is a design parameter and can be chosen to adjust the computational load of the algorithms. We now design a vehicle

model of a set of differential equations in Cartesian space.

$$\dot{x} = v \cos \theta \quad (6.5a)$$

$$\dot{y} = v \sin \theta \quad (6.5b)$$

$$\dot{\theta} = v \kappa \quad (6.5c)$$

This vehicle model is transformed onto the curvilinear coordinate system, and the position of the vehicle in Cartesian space can then be determined by forward integration. Paths that violate the nonholonomic constraints of vehicle motions or collide with an obstacle are eliminated. The remaining paths are evaluated by a construction. The cost function itself can be chosen to optimise driving for an arbitrary property, like sportiness or safety. By simulating the path planner using equally weighted costs, the near-optimal path can be obtained as Fig. 6.2a, with the path drawn in green and the base frame in blue. The manoeuvre selection is subsequently presented as Fig. 6.2b Because this planner insoles a simple vehicle model as a set of ordinary differential equations, it can be conveniently integrated into any control algorithm.

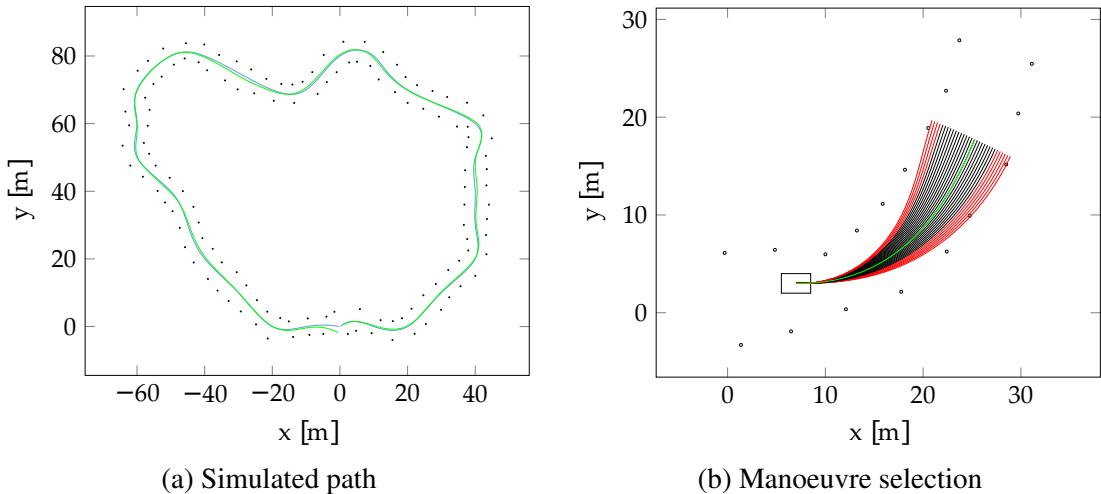


Fig. 6.2 Simulated near-optimal path (a) and manoeuvre selection (b). Black dots represent road delimiters.

6.2.2 Software Communications

The sensors on the vehicle communicate with the path planner using protocol buffers. Protocol Buffers (protobuf) [293] are a formalised data structure developed by Google for use in cross-platform systems. Protobuf allows for the combination of several standard variables into a single packed structure that can be easily serialised and accessed using automatically

generated methods. The protobuf library has bindings for many common programming languages, including C++, Python and Java, meaning that a protobuf structure packaged in a C++ application, can be read in by a Java application with no conversion needed.

Protocol buffers are used internally within our software as regularly structured state variables, with easy to use member functions. For example, the GPS software stores its current state within a protobuf object containing data such as latitude, longitude, ground speed and angle. This GPS state can then be used by internal code regularly, or it can be used serialised and stored. The serialised protobuf data is very compact and space efficient, meaning that a huge amount of logged data can be saved sequentially.

By abstracting the actual data from a specific sensor behind a protobuf object, it allows for the use of Polymorphism within our software, and so dependencies on a specific piece of hardware are loosened. As long as a specific hardware device can be interpreted into the appropriate protobuf form, it can be integrated easily into the system, with only short wrapping code needed to be written. There is also no dependency for this protobuf data to come from a physical sensor, the protobuf data can be read in from a previously serialised log, allowing for the replay of data, or it can be read from an external piece of software, allowing for the use of simulation programs.

6.2.3 Localisation

The vehicle achieves localisation through the inertial measurement unit (IMU) and global positioning system (GPS). The IMU used in the project is the Xsens MTi [257] 9 depth-of-field sensor. The sensor contains several advanced internal algorithms in order to provide accurate and noise-free measurements of the current gyroscopic position, the acceleration, and the magnetic field. These values are returned by the IMU readings as the velocity, acceleration, and the three rotations (pitch, roll and yaw) along the x , y and z axis. The sensor is used within the project to determine heading, and assist in the calculation of local positions.

The GPS receiver used is the Columbus V-800 GPS receiver. It is used with the GPSd [298] software to parse the National Marine Electronics Association (NMEA) standard data outputted by the GPS unit, and retransmit internally in an easier to use format. The data used from the GPS unit are the GPS coordinates, the ground speed, and the heading. These GPS coordinates are first converted into a local reference frame, as a distance from a datum point. The acceleration and position data from the IMU and the GPS units respectively are fused together using an extended Kalman filter (EKF), producing a value for positioning that has a higher accuracy than GPS alone. This fusion system outputs the filtered position, velocity, and acceleration data which is then fed into the Path Planner and Control modules

along with the bearing to ensure that the vehicle can reliably localise itself and obtain accurate readings for the position, velocity and acceleration.

6.2.4 Odometry

The SAE car has been fitted with Hall Effect sensors that send its data through a comparator and an OR gate, this makes a pulse train where we use an Arduino UNO [299] to count the time between pulses to give angular velocity, which can be translated to meters per second. This gives the car Odometry, in which software will use an EKF to fuse the measurements together to improve their individual measurements and the vehicles' localization capabilities. As the Arduino UNO is too slow to control the steering as well as breaking for the SAE Car, we added a second Arduino to do the odometry which then sends the data through serial communications to the main Arduino Nano. This low-level communicates steering and wheel velocity to the Nvidia Jetson TX1 for processing the data through the EKF, using a simple car kinematic model. The goal is to achieve the localization with reduced reliance on the GPS.

6.2.5 LiDAR

The Autonomous Formula SAE vehicle uses an ibeo LUX [254] Light Distance and Ranging system (LiDAR) to sense distance information about the world around it. The LiDAR records the time interval between emission and recapture of thousands of infra-red light pulses to record a stream of 3-Dimensional points. The LiDAR is specifically designed for automotive purposes and is capable of internal data analysis; detecting and classifying objects in its field of vision. The LiDAR connects via an Ethernet switch to the Nvidia Jetson TX1 [117]. A LiDAR reader class receives the serial bytestream which is continuously being transmitted. The data parsing is handled by the parser class which converts the bytestream into Protobuf objects. This format facilitates the storage and sharing of the information to the LiDAR visualisation.

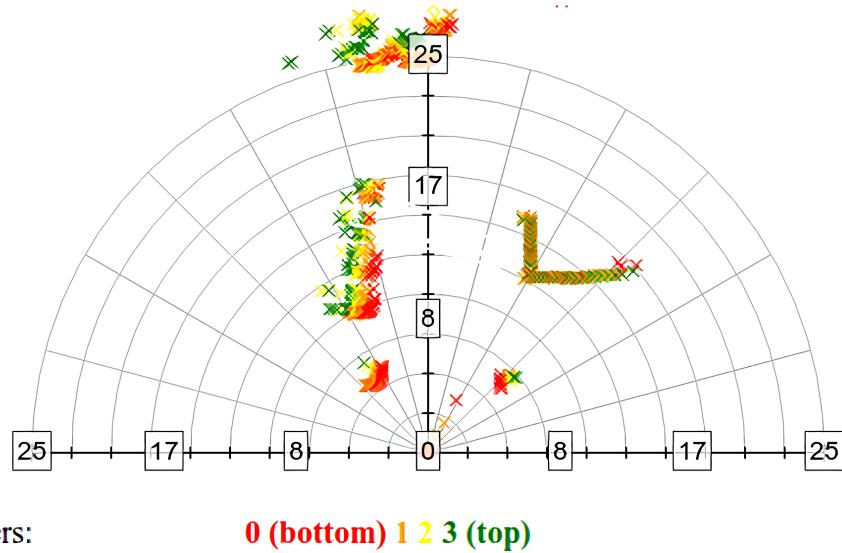


Fig. 6.3 LiDAR plot showing the detected parked vehicles at the position where Fig. 6.4 was captured. The graph axes measure distances in metres.

Road-edge detection is achieved through the use of the LiDAR data. The LiDAR, aimed at an angle below horizontal beyond the front of the car, provides four layers of depth information with a horizontal angle of 85 degrees. Its output is visualised as Fig. 6.3. Road edge detection is achieved by analysing the depth information in one of the layers and checking it for both smoothness and slope. The central data points and those near them are considered and checked to confirm that they meet the slope criteria (the road should be relatively flat so no great changes in depth should be noted in a line) iteratively further and further points are considered in a stepwise process where the correlation coefficient is considered at each point. The road edge is the point at which the correlation coefficient is the highest whilst the slope condition is still being met. This approach was improved with the implementation of a Kalman filter which creates a time-averaged estimate of the road edge-position assisting in the prediction of the current road edge.

6.2.6 Visual Navigation

The SAE vehicle uses visual information as one of its references for driving decisions. It mainly uses an off-the-shelf monocular camera to collect images then applied through OpenCV and SegNet [106] for road edges detection. OpenCV provides many modules, such as image processing, video, and video input/output, that is useful for road edges detection. However, using OpenCV alone for image recognition is limited by variations in image quality, brightness, and contrast. SegNet is an image semantic segmentation approach. It can classify road scene objects, such as the pedestrian, lane marking, traffic light, vehicles

etc., that complement the insufficient of single image processing scheme. SegNet is a pixel-wise semantic segmentation in deep learning framework. Semantic segmentation is used to apply for understanding the visual scene and object. This has been widely adopted in autonomous driving. The architecture of SegNet is a convolution encoder and decoder which is a pixel-wise classifier. The objects classify from SegNet is including following classes, sky, Building, column-pole, road-marking, road, pavement, tree, sign-symbol, fence, vehicle, pedestrian, and bicyclist. The accuracy of classify result is 65.9% for classes average [106]. The input images utilise SegNet to perform visual scene classification. This will produce results whereby road, road-marking, and pavement are classified (see Fig. 6.4), which is useful for road edges detection. OpenCV is simultaneously used to perform image processing. The first step for image processing is camera calibration to get an undistorted image. This is achieved using a chess board image and finding chess board corners to get two accumulated list — 3D point in real world space and 2D points in an image plane. Then we use the camera calibration function in the OpenCV library to obtain the camera calibration and distortion coefficients. This scheme will remove camera distortions.



Fig. 6.4 Segmentation results from a parking area on campus grounds.

The road edges detection scheme detects lane-marking at two sides of autonomous SAE vehicle. The lane-marking detection can also be performed solely by the OpenCV library. Finding the edges of the whole image will reduce the image complexity because numbers of colour and gradient of the image would make image processing more difficult. Canny edge detection [300] is a convenient approach found in the OpenCV library that can be applied for this purpose. Then, Hough transform [301] can be applied onto the image to detect the lanes on both sides of autonomous SAE vehicle. The marking of lanes is detected then using perspective transforms to get a bird's eye view-like image. It can easily find the four points to represent the lane marking pair, where a second order polynomial method can be applied to fit the points. The lane distances are obtained using pixel values that are converted into metres. The scaling factors are according to the Australian road width standard of 3.3 to

3.5 metres. However, the image processing approach might fail because the lane markings are not clear or when there are no lane markings. Therefore, using SegNet's results can effectively circumvent this drawback due to its ability to robustly detect and classify road and lane markings, whereby the same road distance calculation can be applied to find the vehicle's distances to the road edges.

6.2.7 Safety Trip Monitor

The safety trip monitor was designed with an observer-notifier structure. Any of the objects responsible for performing a safety-crucial function in the software can call a trip on the trip state monitor. This includes the low-level safety software, the controller, the GPS software, the web interface, the heart beat and the car network. The trip state is stored by the trip state monitor and pushed to any object which implements the trip state observer class and has registered itself with the monitor. The observer class ensures that the trip state does not produce any irregular operations while driving. The observers which receive the trip state upon each change are attached to the monitor after its instantiation, which means that the set of objects in the software which can change the trip state and which need to track to trip state can be completely reconfigured without needing to make changes to the trip state monitor or observer classes. This is an ideal structure for the software of a research autonomous vehicle, as the continuous development of ongoing research will frequently modify the structure of functions of portions of the software while seeking to maintain the integrity of safety features, like the safety trip.

6.2.8 Controller

The controller class is the high-level interface for the drive-by-wire functionality of the vehicle. The actual drive-by-wire controlling of the vehicle is done by separate software on an Arduino microcontroller. The controller class is the only high-level software which communicates with the low-level controller. The program utilizes three PID controllers to set the throttle, brake, and steering values. The controller class provides a high-level interface with methods to begin to stop autonomous control of the throttle, brake, and steering, as well as methods to set the bearing or speed of the vehicle with desired values. This interface is utilized by the Control program, which handles path planning, and the Fusion program, which provides fused IMU-GPS data in order to facilitate waypoint-based driving. Fig. 6.5a and 6.5b illustrates the base frame and curvature output by the path planner using the Control program based on our evaluation path in Section 6.3. A large change in curvature is present where the U-turn was made.

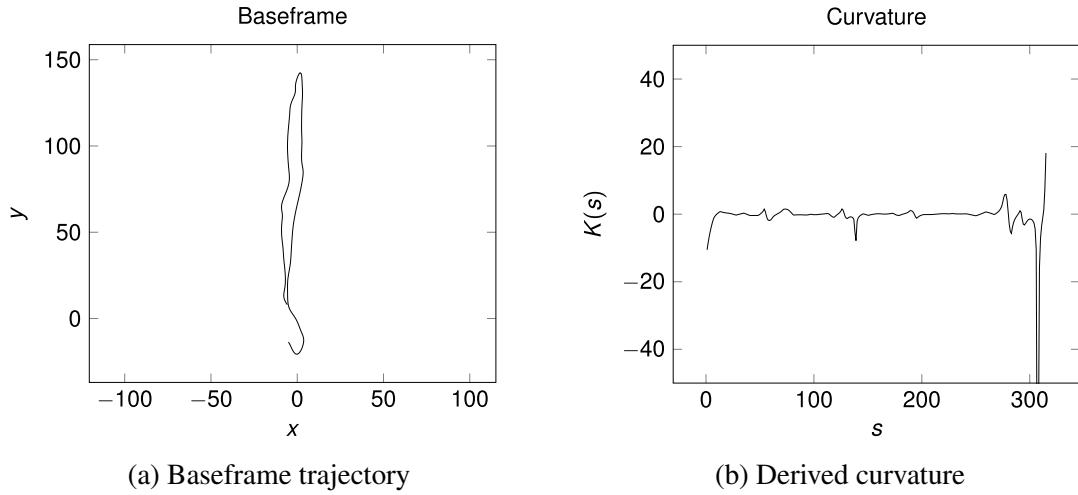


Fig. 6.5 The trajectory base frame generated by the path planner (a) and its derived curvature (b).

6.3 Implementation on SAE Vehicle

The software for the autonomous driving system is programmed onto an Nvidia Jetson TX1 embedded computer that is mounted on the chassis of the vehicle. The environmental sensors namely the LiDAR, GPS, IMU and camera connects directly to the camera via Ethernet (LiDAR) and USB 2.0 (GPS, IMU, camera) respectively. This software can be implemented onto another vehicle so long as the same sensors are used, as the system outputs drive commands through the Control module, which can be configured according to the vehicle's hardware specifications. To test our system, we collected driving data with the vehicle driving in a parking area at the University of Western Australia (shown in Fig. 6.6) by recording readings from the GPU, IMU, LiDAR and camera. The GPS and IMU plots waypoints for the car, LiDAR performs obstacle detection, and the camera performs semantic segmentation. The test drive begins from the southern end and then driving towards the northern end before making a U-turn back to the vehicle's station position. The recorded waypoints are passed to the path planner, which generates a trajectory base frame as shown in Fig. 6.5a. Subsequently, the curvature is obtained from the derivative of the base frame as Fig. 6.5b, which can then be used to determine the steering angle for autonomous driving.



Fig. 6.6 Map showing the path taken by the vehicle in with a solid red line.

6.4 Results

Fig. 6.4 was captured while the car was driving northward as it approaches the end of its path. The input image is displayed on the left and its semantic segmentation result is displayed on the right image. Results from semantic segmentation showed that the road is properly classified, along other elements in the frame. Its LiDAR readings at that position is as illustrated in Fig. 6.3, whereby the parked vehicles are detected on the left, along with the vegetation in the distance and the wall on the right side of the vehicle. The combination of LiDAR and semantic segmentation enables the vehicle to understand its position on the road, along with the obstacle types and the distances to each obstacle. For further results on road and lane marker detection, we processed a drive recording from Udacity’s Self-Driving Car Nanodegree [135]. From Fig. 6.7, the input image (Origin) is used for both semantic segmentation (SegNet), and bird’s eye view transformation (Bird view). The system was able to identify objects on the road scene, and the curvature of the road can be calculated using the bird’s eye view. The distance between the centre of the vehicle and the left and right lane markers are calculated as shown in the output. This is accompanied with a confidence value whereby a successful detection of the road lane markings will be denoted with a “1”.



Fig. 6.7 Original and Jetson TX1 output showing segmentation, bird's eye view, vehicle distance to left and right lane marks.

6.5 Conclusion

In this paper, we have designed and demonstrated the functionality of our software framework that is implemented on our autonomous SAE vehicle. This framework is designed to cohesively interface with the vehicle's camera, LiDAR, GPS, IMU and wheel speed sensors while being capable of performing navigational tasks such as path planning, image processing, odometry, localisation, safety checks, speed and steering control. Each sensor and navigation task is programmed as a separate module to ensure modularity and scalability, allowing for each module to be changed independently. Protocol buffers handle intermodular communications, whereby each process parses its output as a protobuf to be sent to another module. With this framework implemented on the Jetson TX1, our test drives on the autonomous SAE vehicle was able to achieve results that are adequate for fully autonomous driving. Future works will include further testing of the autonomous navigation software and refinements to the control and path planner classes to ensure that the system is capable for road drives using full automation.

Chapter 7

Evolution of a Reliable and Extensible High-Level Control System for an Autonomous Car

The reliability of autonomous vehicles is heavily dependent on their software frameworks, which are required to interface and process data from many different sensors on board the vehicle, perform navigational processes such as path planning and lane keeping, take action to ensure safety and display data to an operator in a useful fashion. These sensors can include a combination of cameras, LiDARs, GPS, IMU, and odometric sensors to achieve positioning and localisation for the vehicle and nearby objects in their environment and can be challenging to integrate. In this paper, we present a hybridised software framework that combines sensor and navigational processing for autonomous driving. Our framework utilises a modular approach for interfacing and safety functionality, whilst navigation and sensor interfaces are implemented as nodes in the Robot Operating System (ROS). Our testing results verify the suitability of our framework by integration with a hardware-in-the-loop simulation system and for fully autonomous field driving.

7.1 Introduction

The Renewable Energy Vehicle Project (REV) at UWA has developed an intelligent autonomous test vehicle, utilising a Formula-SAE race car (see Fig. 7.1) as a platform (Formula SAE [289] is an annual student competition organised by the Society of Automotive Engineers with events worldwide). Using such a vehicle allows us to target driving applications, both on- and off-road, in structured and unstructured driving environments. We have incorpo-

rated full drive-by-wire control of the electric vehicle's throttle, steering and the hydraulic braking system. The use of a Formula-SAE car provides several advantages for such a project as the vehicle is mechanically simple. Formula-SAE cars with similar designs are common at universities worldwide and the size of the vehicle makes testing accessible. Furthermore, the use of an electric vehicle makes the project significantly more practical for student work and the hardware installed in this project can take advantage of the large amount of electrical energy already available on the vehicle.



Fig. 7.1 Autonomous SAE Vehicle.

For the driverless FSAE project, our goal was to be able to autonomously drive a vehicle around a race track. Initially, this was achieved by placing waypoints along the ideal driving line, as well as "fence points" to lock out non-driving areas. Maps can either be recorded by human or remote-controlled driving or specified through a Google Maps driven web interface. During autonomous driving, a laser scanner and camera are used for detection of road edges as well as any obstacles on the track. Our current work involves increasing the level of automation to drive a semi-structured (traffic cone or road edge delineated) race track by first automatically driving and mapping the path without prior knowledge of the track and then redriving it, optimised, at greater speed with the assistance of inertial navigation.

Safety systems are essential for a driverless system, as the car weighs more than 250 kg and is capable of driving at a speed of 80 km/h. Both the low-level drive-by-wire, as well as high-level navigation system have independent safety systems built in. These include active geofencing, remote intervention, remote heartbeat and remote emergency stopping,

which are implemented through a fail-safe wireless link to a base station as well as through hard-wired buttons on the vehicle itself.

The previous revision of our framework [290, 302] had a heavy reliance on a central control module, which required all the sensors and their submodules to function. These submodules were developed over time using different programming languages and are partially redundant, which made integration difficult. First, this was streamlined in an approach whereby each module is programmed with a C++ interface that communicates with either a path planner or a drive control system. Here, we present the integration of this approach with the Robot Operating System [303] which allows the further separation of functions into ROS nodes and provides a consistent application programming interface (API) for implementation of additional sensors and higher level automation, whilst the original program, now running as a ROS node, deals with critical interfacing and safety functions.

Additionally, this approach presents a long-term advantage whereby our framework is made fully open and contributable by students and enthusiasts looking to implement our framework onto their custom-built vehicles. When compared against other autonomous driving frameworks such as Apollo [291] and Autoware [292] that mostly target commercial vehicles requiring expensive hardware, our approach leverages hardware and fabrication methods that are more affordable. The framework has been integrated with a simulator which provides the ability to test software modifications and algorithms prior to deployment to the vehicle.

Our contributions in this paper are demonstrated through the proposal of our high-level autonomous control system that interfaces through standard, off-the-shelf sensors and equipment. This system is made holistic through the integration of the following features — real-time localisation through odometry and dead-reckoning; object segmentation and detection using LiDAR and camera; real-time path planning via waypoints or object positioning; visual navigation with odometry, object recognition and tracking, and semantic segmentation; and a hardware-in-the-loop simulator for prototyping verification.

The remainder of this paper is organised as follows. Section 7.2 introduces the system framework with an overview. Section 7.3 describes the sensors that are used. Section 7.4 presents the path planning approaches used in the system. Section 7.5 explains how visual navigation is performed on the system. Section 7.6 highlights our driving simulator. Section 7.7 describe our system validations, before the concluding remarks are drawn in Section 7.8.

7.2 System Overview

In order to satisfy the requirements of resiliency, flexibility, extensibility and simple integrations, a publish/subscribe software architecture was used. This allowed for highly decoupled software to be developed, with each component or series of components needing only to conform to the expected message type on the input and output topics. The use of a publish/subscribe architecture allows for nodes to be easily swapped in and out in order to test different solutions, as well as allowing multiple components to make use of the same data sources without modifying the source, providing simple methods of logging, data capture and data replay.

It was decided to use ROS as the publish/subscribe layer of the application. This decision was influenced heavily by the usage of ROS in the Apollo Auto [291] project, as it is shown to be reliable and performant. This also provides a path for upgrades, with a version of ROS modified for automotive use through the addition of shared memory transport for message passing, support for the Protobuf [293] message passing protocol, and the decentralisation of ROS to reduce single points of failure available on an open-source licence. This also allows for any components developed to be more easily ported to the complete Apollo Auto platform at a later stage. In addition, the popularity of ROS ensures that there are a large number of existing modules available for use, allowing the team to focus on the goals stated above, instead of on creating supporting code and utilities. Based on the desired functionality, it was determined that the publish/subscribe architecture would initially require the nodes and topics outline in Fig. 7.2.

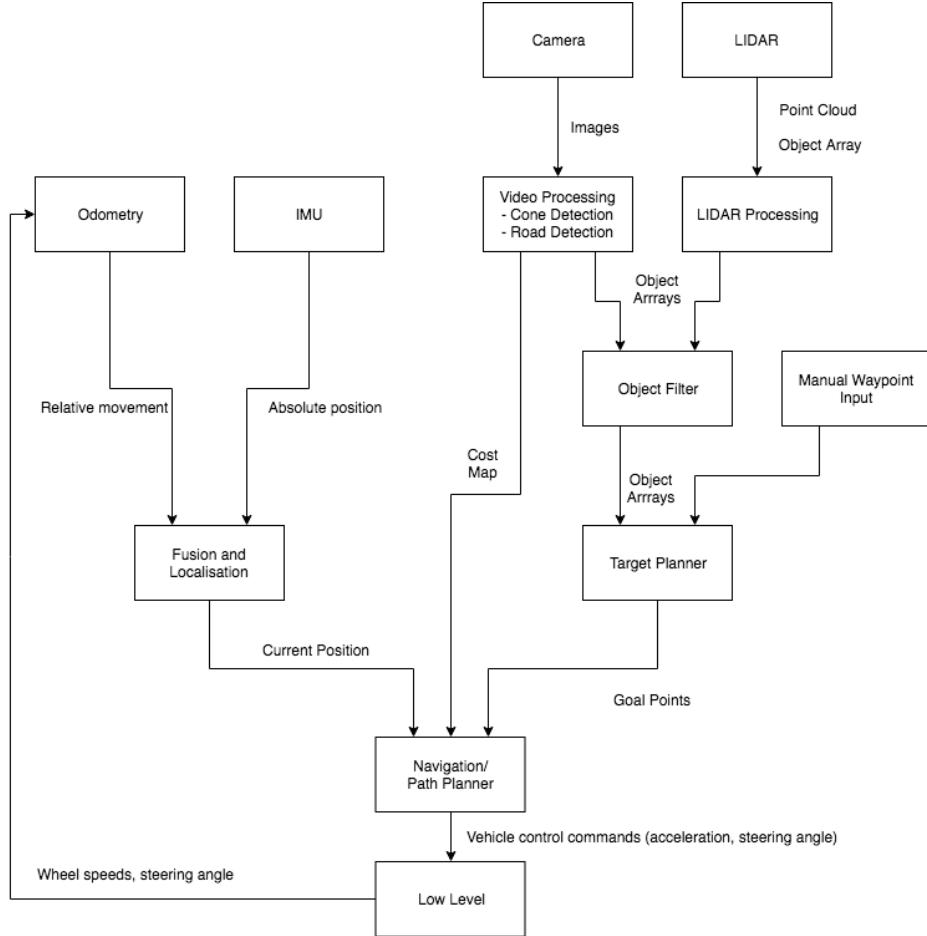


Fig. 7.2 SAE vehicle software framework.

We migrated the existing code base of the high-level software system to use the ROS framework in 2018 to reduce the development complexity of the software system. ROS provides low-level device control, implementation of commonly used tools, message passing between processes, and package management [303]. Instead of creating an independent system where a *broker* would manage the intercommunications between *modules* (programs that have a specific function) ROS readily provides these services. Hence, the user only has to create *nodes* (programs that perform a certain function) that listen and talk to other nodes. Using ROS therefore simplifies the integration process for each individual module in the system. By defining the topic information for messages to communicate, the individual nodes can work together without too much effort in integration.

More specifically, existing modules in our system presented in [302], including modules for logging, web server and serial communication were replaced with their equivalent ROS packages, as they are often more stable and better supported. All existing messages from the

system are converted into ROS messages. The testing for the individual modules therefore only requires minimal changes to the core software.

The ROS version used on the SAE vehicle is currently ROS Kinetic Kame running on Ubuntu 16.04 LTS which will be long-term supported until April 2021.

7.3 Navigation Sensors

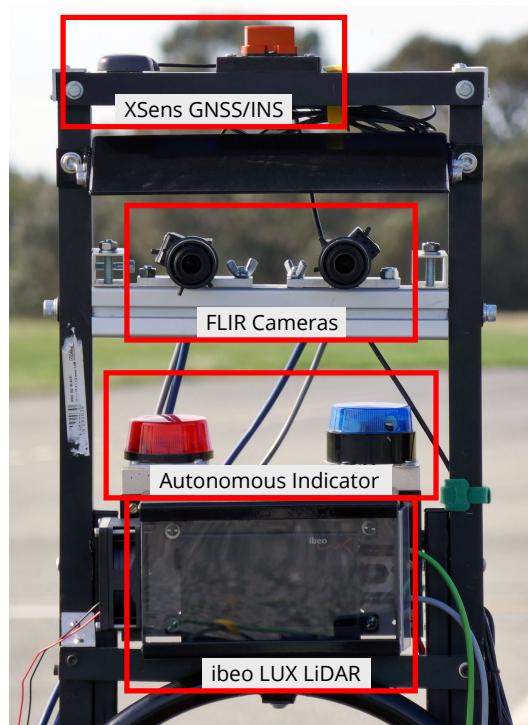


Fig. 7.3 The SAE vehicle's rack-mounted sensors.

The SAE vehicle performs autonomous driving through a combination of navigation sensors including LiDARs, cameras, wheel odometry and inertial measurement unit (IMU) (as shown in Fig. 7.1, with the sensor rack detailed in Fig. 7.3) which are categorised into four categories — odometry, dead reckoning, LiDAR and camera systems, which are elaborated upon in their individual Sections 7.3.1 to 7.3.4.

7.3.1 Odometry

The SAE vehicle has been fitted with Hall Effect sensors on each wheel which send data through a comparator and an OR gate, and generate a pulse train to an Arduino Nano [304]. The sensors count pulses for each wheel and report them to the low-level controller with

timestamps. The linear velocity and rotational velocity are then evaluated by a low-level controller with the feedback from the steering sensor. The accumulated pose information is then combined with the wheel odometry of the SAE car. The goal for implementing this wheel odometry is to provide basic offline localisation within a low-level system and to be further fused with other sensors such as IMU, LiDAR and cameras for a more accurate global positioning.

7.3.2 Dead Reckoning

Having an accurate state estimation is crucial for making optimal decisions for future control inputs to effectively navigate the environment. Dead reckoning on the vehicle is achieved through the Xsens MTi-G-710 [305], which is an inertial measurement unit (IMU) that is equipped with a global navigation satellite system (GNSS) running at 50 Hz. However, these sensors are susceptible to noise and imperfections which introduce uncertainty to the measurements. Hence, we introduce an extended Kalman filter (EKF) to fuse data from these sensors with that from odometry using a model of the car's dynamics to obtain a more precise estimate of its state. Since the computation of the car's movements requires direction, trigonometric functions are needed. The EKF linearises these non-linear functions using a first-order Taylor series approximation [306], where it is approximated according to [307] in equation (7.1).

$$f(u_k, x_{k-1}) \approx f(u_k, x_{k-1}) + \frac{df(u_k, \mu_{k-1})}{dx_{k-1}}(x_{k-1} - \mu_{k-1}) \quad (7.1)$$

where u and μ are the mean and the estimate of x , respectively. With an EKF, we calculate the fused covariance values P_k by predicting next state and the next error covariance using the current state and current estimate error covariance. x_k is the current pose at time instance k and f is a non-linear transition function that converts the past state to the current state; state x is composed of the car's x - y coordinates and orientation φ . We perform this as a three-step process, following the descriptions in Section III of [308]:

1. Compute the Kalman gain K
2. Perform the correction to find the current state x_k , and
3. Calculate the new process error P_k

This is used as the foundation for our experiments on sensor fusion, as described in Section 7.7.1.

7.3.3 LiDAR System

The vehicle utilises an array of Light Distance and Ranging (LiDAR) systems. This consists of a SICK LMS111-1010 [255] and an ibeo LUX 4 [254] connected through an Ethernet switch to the Nvidia Jetson TX1. The LMS111 scans a single layer at 50 Hz while the LUX scans four layers at 10 Hz featuring in-built object detection and tracking. The data is published by each of the LiDAR's ROS drivers and processed to achieve desired functionalities.

The LMS111 is mounted forward-facing on the front of the vehicle at 15 cm above the ground to obtain a 270° field of view, suitable for detecting close obstacles and scans a plane close to horizontal. The point cloud is sorted and then from left to right; each point is assigned a cluster identification number based on distance to the next point and the angle between it and the next point relative to the laser. This delivers accurate obstacle detection with features such as cluster size indicating the size of the obstacle, allowing for classification of obstacles (such as a cone). The LUX is mounted above the driver and pitched towards the ground slightly such that the lower layer scans the ground 20 metres ahead of the vehicle. It is utilised to achieve road edge and object detection at a distance.

Road edge detection is achieved by analysing the depth information in one of the layers and checking it for both smoothness and slope. The central data points and those near them are considered and checked to confirm that they meet the slope criteria (the road should be relatively flat so no great changes in depth should be noted in a line). Iteratively, further and further points are considered in a stepwise process where the correlation coefficient is considered at each point. The road edge is the point at which the correlation coefficient is the highest whilst the slope condition is still being met. This approach was improved with the implementation of a Kalman filter which creates a time-averaged estimate of the road edge-position assisting in the prediction of the current road edge. A detailed description of our methods is presented in [287].

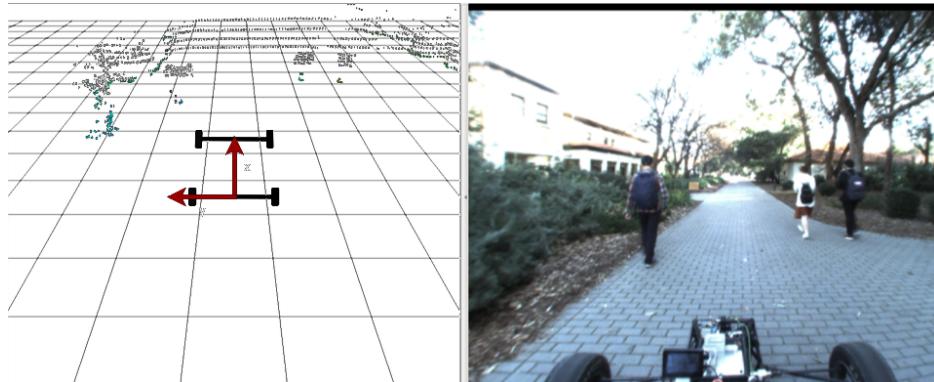


Fig. 7.4 Point cloud generated from the LMS111 (coloured) and the LUX (4-layers, white) (left) and the scene where it was captured (right).

The in-built object detection and tracking of the LUX will be used for fusion with the obstacles reported through processing of the LMS111's data. The comparison of positions of objects reported by the LUX and LMS111 will increase the likelihood of detecting an object. The LMS111 giving information on the objects on a low and horizontal plane and the LUX giving information of objects in the mid to far range. In turn, this object information will also be fused with that of the camera vision. We use both LiDARs to collectively generate a point cloud as illustrated in Fig. 7.4, captured during a test run. The LMS111 point cloud is coloured based on its intensity, retrieving information on the reflectivity of the surface struck by each point. The LUX is scanning layers onto the path in the distance while also picking up a large amount of detail from the surrounding vegetation as pedestrians walk past.

7.3.4 Camera System

A pair of FLIR Blackfly GigE [309] cameras are mounted on the vehicle's frame to perform tasks related to visual navigation, such as semantic segmentation and visual odometry. These cameras are fitted with Fujinon f/1.2, 2.8–8 mm wide-aperture varifocal lenses [310], and are individually capable of capturing a wide field of view. To suit our application, these cameras use 1.3 megapixel 1/3" global shutter CCD image sensors that will not be affected by any distortions caused by the rolling shutter effect [311]. The cameras are connected to a Gigabit Ethernet switch that connects to the Jetson TX1, interfacing them through Blackfly's ROS node where it is configured to record at 10 Hz per channel.

At the time of writing, the use of stereoscopic vision for autonomous driving is still subject to evaluation. Our application focuses on using monocular vision that is paired with measurements from the LiDAR system in order to achieve depth perceptions. The methods that we use for semantic segmentation and visual odometry for this paper currently do not

require a stereoscopic setup. Our implementation of these methods is further described in Section 7.5.

7.4 Path Planning

We have programmed the control system to deliver path planning routines to drive either through a series of predefined waypoints, or in between a series of traffic cones placed on either side of the vehicle.

7.4.1 Waypoint Driving

The underlying idea behind waypoint driving is to drive a set of predefined points in between the starting position P_1 ($x = 0$, $y = 0$ and orientation $\varphi = 0$) to the destination position P_2 , which can be obtained through the differences in GPS coordinates. These waypoints can either be stored in Cartesian coordinates in an array or in our test case, they are selected based on the driver's preference by selecting position points on RViz [312], which are then confirmed on the console. To ensure that all points can be driven smoothly with the consideration of the correct heading to the subsequent point, a spline approach has been implemented within this design to generate the desired path.

Once the waypoints are chosen, two static paths are shown on RViz (see Fig. 7.5). The first path (green) consist of straight lines that interlink all the points with the arrow at the end showing the destination heading. The second path (purple) is the desired path which consists of a smoothed curvature that passes through all of the waypoints, the generation of this path is based on the Hermite spline interpolation technique [313] with the required parameters which are the four vectors:

1. Current position, P_1
2. Target position, P_2
3. Tangent of departure from current position, T_1
4. Tangent of approach from target position, T_2

Along with four Hermite basis functions $H_n(u)$

$$H_1(u) = 2u^3 - 3u^2 + 1 \quad (7.2a)$$

$$H_2(u) = -2u^3 + 3u^2 \quad (7.2b)$$

$$H_3(u) = u^3 - 2u^2 + u \quad (7.2c)$$

$$H_4(u) = u^3 - u^2 \quad (7.2d)$$

where $0 \leq u \leq 1$ which represents the start to finish motion. We then construct the resultant path f by calculating the product between the vectors and the Hermite basis functions

$$f(x, y, \varphi) = H_1P_1 + H_2P_2 + H_3P_3 + H_4P_4 \quad (7.3)$$

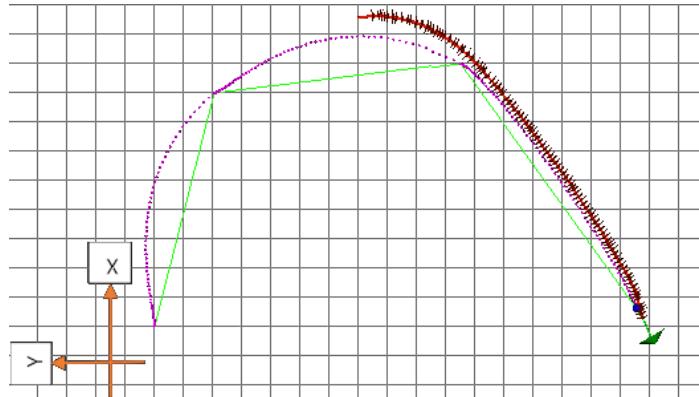


Fig. 7.5 Calculated waypoints on RViz with inputs from (7.2) and (7.3).

The actual simulated driving pattern (red line) is determined based on the distance measurement between the current position of the vehicle against the desired path (purple) with a predefined tolerance range, and limited maximum turning angle ranged between $\pm 25^\circ$. The desired path is constituted by a finite number of points generated from the Hermite spline interpolation. To avoid oversteering or understeering, we compute the slope differences for both the driving path and the desired path. With this logic in place, the vehicle is either turning left ($\varphi > 0^\circ$), right ($\varphi < 0^\circ$) or moving straight either at a constant speed or slowed speed for a sharper turn in order to reach the goal point ($\varphi = 0^\circ$).

7.4.2 Cone Driving

The current iteration of the path planning procedure uses obstacle detection of the cones to determine the correct path. This algorithm is similar to our solution in [302], but simplified

to allow for quicker calculation. Our cone driving module accepts cone locations from either the map, LiDAR or camera, classifying them as objects. Then, the vehicle navigates to drive within the track formed by cones safely without collision. Using a range of the maximum turning circle of the car, of both a left-hand turn and right-hand turn, it then looks at which predicted paths will intercept cones. The vehicle dynamics are thus limited during motion planning such that the steering angle does not exceed 25°. Our algorithm will iterate through all cones within the car's range and calculate the best collision-free path to undertake, as detailed in Algorithm 1.

Algorithm 1 Cone driving

```

1: procedure CONEDRIVE(cones in range)
2:   init steering_range to [-25,25]
3:   for all cones in range do
4:     evaluate collision_range with cone
5:     exclude the collision_range from steering_range
6:   end for
7:   if steering_range is empty then
8:     stop
9:   else if all steering_range ≤ threshold then
10:    select largest steering_range
11:   else if all steering_range > threshold then
12:    select steering_angle with minimum change in current direction
13:   end if
14:   drive toward centre of steering_range
15: end procedure

```

7.5 Visual Navigation

The vehicle is capable of performing visual navigation tasks through a combination of semantic segmentation, visual odometry and visual cone detection that is decided depending on the application requirements. These tasks commonly rely on the OpenCV [314] library, utilising functions such as handcrafted feature detection, general image processing and transforming. It achieves a visual perception of the driving environment through the camera system as described in Section 7.3.4.

7.5.1 Road and Lane Detection

Our system uses either semantic segmentation or edge detection to detect road edges and lane markings, depending on the environment's complexity. Using semantic segmentation also enables obstacle recognition which can be integrated with the LiDAR system. Environments are complex when there is a lack of uniformity in pose, features and illumination. While it is possible to solely rely on modules within OpenCV here, the performance of using handcrafted features alone for image recognition is restricted by variations in image quality, brightness, and contrast. In order to improve its performance under these environments, we selected SegNet [106] for semantic segmentation due to its high compatibility and ease of implementation. Its ability to perform pixel-wise classification for road scene objects complements the drawbacks of a single image processing scheme.

The architecture of SegNet uses a convolution encoder and decoder setup that classifies objects into one of the following classes — sky, building, column-pole, road-marking, road, pavement, tree, sign-symbol, fence, vehicle, pedestrian and bicyclist; with a class average classification accuracy of 65.9% [106]. Our application uses SegNet whereby road, road markings, and pavement are classified (see Fig. 7.6), which is useful for road edges detection. However, OpenCV is simultaneously used to perform image processing, with the first step being camera calibration to get an undistorted image. This is achieved using a chessboard image and finding its corners to get two accumulated lists — a 3D point in real-world space and a 2D point in an image plane. We then use the camera calibration function in the OpenCV library to obtain the camera calibration and distortion coefficients. Our experiments using SegNet for autonomous driving was performed in [315], where we have evaluated its segmentation accuracy in our test environment through the calculation of its pixel accuracy (PA).

$$\text{PA} = \frac{\sum_i n_{ii}}{\sum_i t_i} \quad (7.4)$$

where n_{ii} represents the number of classified class i pixels correctly classified to belong in class i , and t_i represents the total number of pixels in class i belonging in the ground truth.



Fig. 7.6 Semantic segmentation results during a test drive. This scene resulted in a pixel accuracy of 99.31%.

The road edges detection process finds lane markings at both sides of the car. We have noted that lane marking detection can also be performed solely using OpenCV functions, especially in non-complex, uniform environments with minimal illumination variations, thereby reducing its computation requirements. Algorithm 2 describes our approach.

Algorithm 2 Road and Lane Detection

```

1: procedure LANEDETECT(histogram_vector from camera)
2:   undistort image_frame from lens distortions
3:   Sobel filter image_frame as filtered_image
4:   threshold filtered_image as binary_image
5:   obtain histogram_vector for binary_image on y-axis
6:   split histogram_vector into left_half and right_half
7:   for each histogram_vector do
8:     find peak_position on y-axis
9:     init sliding_window at bottom of image at peak_position
10:    while sliding_window not at top_of_image do
11:      find mass_center of sliding_window as line_point
12:      move window to the mass_center
13:      move window iteratively on x-axis towards top_of_image
14:    end while
15:    fit line_point with second-order polynomial
16:   end for
17: end procedure

```

The lane distances are obtained using pixel values that are converted into metres, and its scaling factors are according to Australian road width standard of 3.3 to 3.5 metres. However, this image processing approach might fail when the lane markings are not clear or there are no lane markings. Therefore, using SegNet's results can effectively mitigate this drawback

due to its ability to robustly detect and classify road and lane markings, whereby the same road distance calculation can be applied to find the vehicle's distance to the road edges.

7.5.2 Visual Odometry

Our system implements ORB-SLAM2 [168] as our baseline algorithm for visual odometry based on its use of Oriented FAST and rotated BRIEF (ORB) features for mapping, tracking and place recognition. ORB features are similar to Binary Robust Independent Elementary Features (BRIEF) with an extra feature of introducing rotation invariance and noise resistance, while utilising Features from accelerated segment test (FAST) for corner detection. This results in a balanced compromise between accuracy and computation footprint, making it desirable for our hardware setup. Although initially proposed as a visual simultaneous localisation and mapping (SLAM) algorithm, our ORB-SLAM2 application focuses on visual odometry as we do not implement its loop closing feature. To further increase the efficiency of this algorithm for our specific needs, a new set of vocabularies were trained using the images collected from the cameras mounted on the car. This reduces the size of vocabularies, which results in improved memory footprint. The Jetson TX1's 256-core embedded GPU is being used to improve image processing through parallelisation. The original ORB-SLAM2 was not adapted for this acceleration. In order to boost runtime performance, the applied ORB-SLAM2 has been rewritten to adapt CUDA [207] and thus utilise the GPU on the TX1 [316]. Fig. 7.7 illustrates an experiment with ORB-SLAM2 on a path segment as shown in (a), with its generated path in (b) through the tracking of features along subsequent frames.

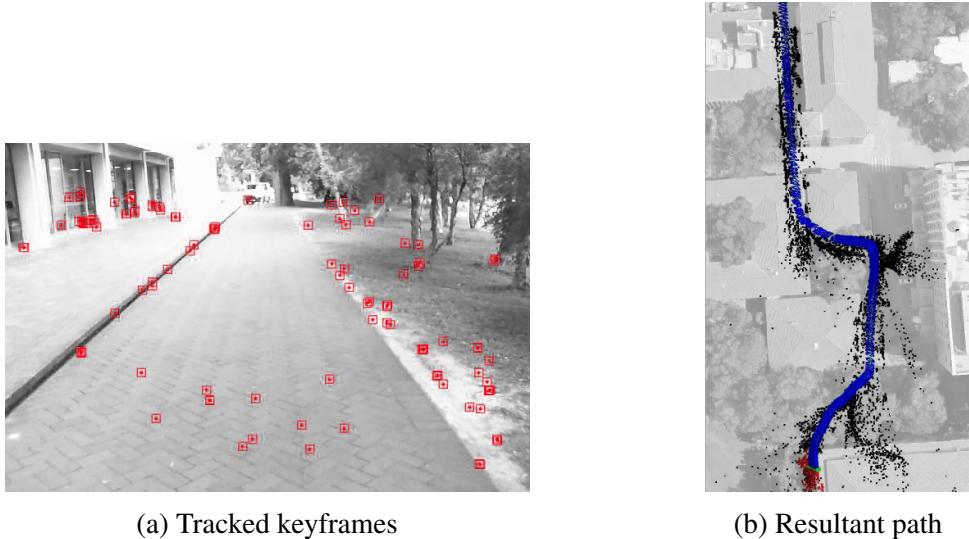


Fig. 7.7 ORB-SLAM2 experiment showing (a) tracked keyframes as red points that results in (b) the generated path in blue with the said tracked features as red dots, overlaid on satellite imagery.

7.5.3 Cone Detection

Handcrafted features that combine a linear classifier are utilised for cone detection using OpenCV. We use the histogram of oriented gradients (HOG) descriptor across an image to find and segregate regions of interest (ROIs) that may encompass a cone, which is then used as inputs for a support vector machine (SVM) classifier. For all regions that are positively classified, the hue layer is thresholded with an orange value, as our system is benchmarked using orange cones. We finally apply a histogram to the thresholded image and then obtain the position of the cone within the image frame. However, in order to fuse this classification result with other sensors, the detected cones must be presented in the global reference frame. This is done by applying a perspective transform to the image, and with the assumption that the vehicle is driving on a flat plane. The position of the cones in the global frame can then be obtained by projecting these cones onto a horizontal ground plane.

In order to reduce the effect from variations in brightness caused by the different sunlight angles, more samples must be included in the training process of the SVM. This significantly degrades the runtime performance of the entire system while offering only a minor accuracy improvement. Because of this, we designed a convolutional neural network (CNN) to provide a flexible feature extraction method to adapt this variation in the environment. This network has two convolutional layers and two fully-connected layers, which are used for detecting cones. In order to run the visual cone detection process smoothly along with all other modules

in the system, we designed the network to be simple with a small footprint. Using this CNN yielded increased detection accuracies as compared to the SVM approach while offering similar runtime performance [317].

7.6 Hardware-in-the-Loop Simulation

Simulation is a cornerstone of autonomous vehicle testing, allowing high-level software such as image processing and path planning to be tested in predefined scenarios on a much faster schedule than is possible with hardware testing. In addition, the use of autonomous driving simulations allows for testing to be performed in faster than real time, and for testing to be scheduled and the results reviewed at a later time.

We designed a hardware-in-the-loop (HIL) simulation system, whereby an interface between the high-level software of the autonomous SAE vehicle and the CARLA driving simulator [318] allows software components such as localisation and path planning to be tested in a simulated environment, resulting in faster iterations as tests can be conducted at any time that is convenient. This interface consists of a ROS node that retrieves environment data such as a camera feed and LiDAR point cloud from CARLA and sends movement commands to CARLA. Due to the message passing system used by ROS, this interface can easily generate and consume the outputs and inputs expected by other ROS nodes without requiring changes to the application, even across multiple devices, and allows for the application architecture in Fig. 7.8. This allows CARLA to be run on a more powerful computer more suited to generating a simulated environment while still allowing for the Jetson TX1 to be used to run the SAE vehicle software in order to maintain as realistic an environment and workload as possible, as the control hardware and software on the simulator is identical to the ones used in the real vehicle. In addition, the simulation node handles input from a Logitech G920 racing wheel [319] and simulates the low-level safety systems in order to allow overriding autonomous functions using manual inputs in a similar manner to what is possible on the SAE vehicle.

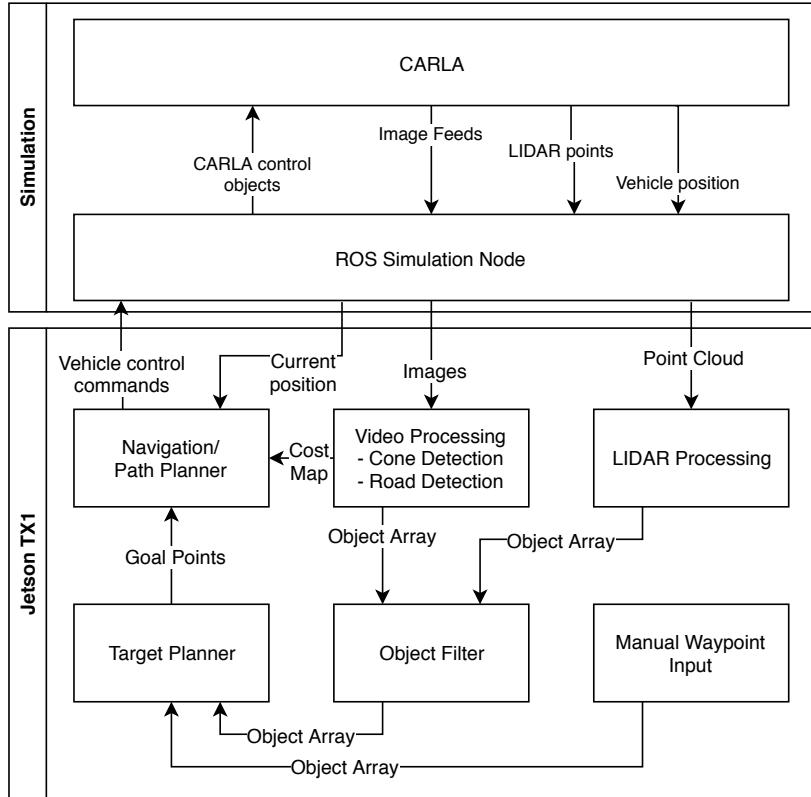


Fig. 7.8 Simulation software framework.

7.7 System Validation

In order to verify our system, we conducted experiments relating to sensor fusion for dead reckoning, waypoint and cone driving, and the driving simulator, which are elaborated individually in Sections 7.7.1 through 7.7.4.

7.7.1 Sensor Fusion

The odometry measurements are compared to the fused odometry and IMU position estimate using our approach described in Section 7.3.2. To gather the data, the vehicle was driven in a relatively straight path on an even plane and as a result the z coordinate was omitted. In Table 7.1, we hence measured and calculated the displacement of the car's state Δx_k and its error covariance P_k across three time instances k measured in seconds. We then compared the values obtained through pure wheel odometry (no IMU fusion) against that from the EKF (with IMU fusion); these values are expressed in Cartesian coordinates (x, y) and are measured in metres.

Table 7.1 Displacement & Error Covariance Comparison

IMU Fusion	k	Δx_k	P_k
No	50.88	(1.430, 0.030)	(0.0160, 0.0160)
	100.91	(41.070, 13.934)	(0.0160, 0.0160)
	150.89	(58.000, 27.680)	(0.0160, 0.0160)
Yes	50.88	(1.436, 0.025)	(0.0130, 0.0134)
	100.91	(41.073, 13.901)	(0.0132, 0.0132)
	150.89	(57.970, 27.658)	(0.0132, 0.0133)

The results in Table 7.1 shows an improvement in the certainty of positioning, whereby sensor fusion has performed corrections to the coordinates and improved the covariances in x and y . In our tests, the combination is sufficiently accurate for this application.

7.7.2 Waypoint Driving

We carried out our experiments for waypoint driving by measuring its path planning accuracy through the calculation of waypoints across several driving scenarios as shown in Fig. 7.9. Figure legends are as presented in Section 7.4.1. This accuracy is quantified by the car's projection error ε_P (the maximum deviation of the path projection from the ground truth), and its root-mean-square error ε_{rms} .

$$\varepsilon_{\text{rms}} = \sqrt{\frac{\sum_{i=1}^n (D_i - M_i)^2}{n}} \quad (7.5)$$

where n denotes the total number of records; D_i and M_i denotes the distance of the ground truth and the projected path, respectively at record i .

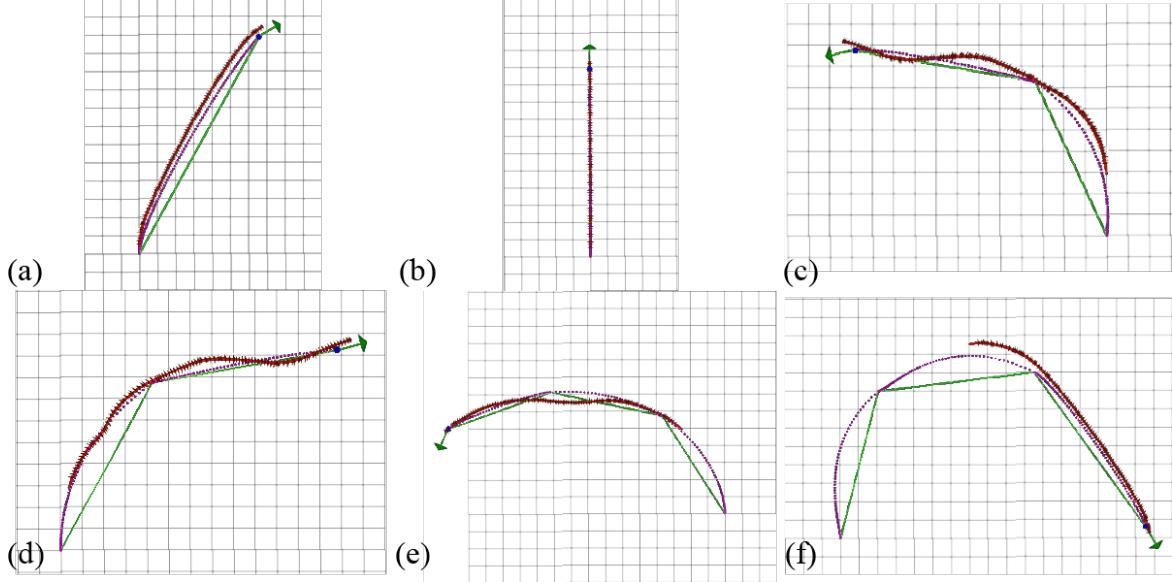


Fig. 7.9 Generated path projections M_i (red), its ground truth D_i (purple) and the linear displacements of waypoints (green). [Grid size: 1 m \times 1 m]

Distance error measurements are shown in Table 7.2, while heading angle deviations were found to be insignificant.

Table 7.2 Error and Distance Measurements from Fig. 7.9

Fig. 7.9	ε_p (m)	M_i (m)	D_i (m)
(a)	0.316	13.895	13.579
(b)	0.000	10.895	10.895
(c)	0.412	15.053	14.158
(d)	0.368	15.263	14.482
(e)	0.474	13.895	14.105
(f)	0.626	14.053	13.737

Using the values of M_i and D_i in Table 7.2, ε_{rms} was calculated to be 0.525 m, which is 85% accurate when compared to the average transverse track width of 3.5 m. This accuracy indicates that D is relatively close to M across the total i records. Additionally, the increase in track complexity (such as through the addition of sharper turns and more segments) contributes to a greater increase in ε_p , as compared to the increase of $(D_i - M_i)$. As expected, ε_p is non-existent when a perfectly straight path is generated as shown in Fig 7.9b.

7.7.3 Cone Driving

Cone driving on the car follows the setup as prescribed by the standards set by Formula Student Germany's Track Marking and Skidpad for Dynamic Events (DE6.3/DE6.4) [320]. Each cone measures $228 \times 228 \times 325$ ($l \times w \times h$) mm and they are placed as pairs 5 m apart, creating a track width of 3.5 m, as illustrated in Fig. 7.10.

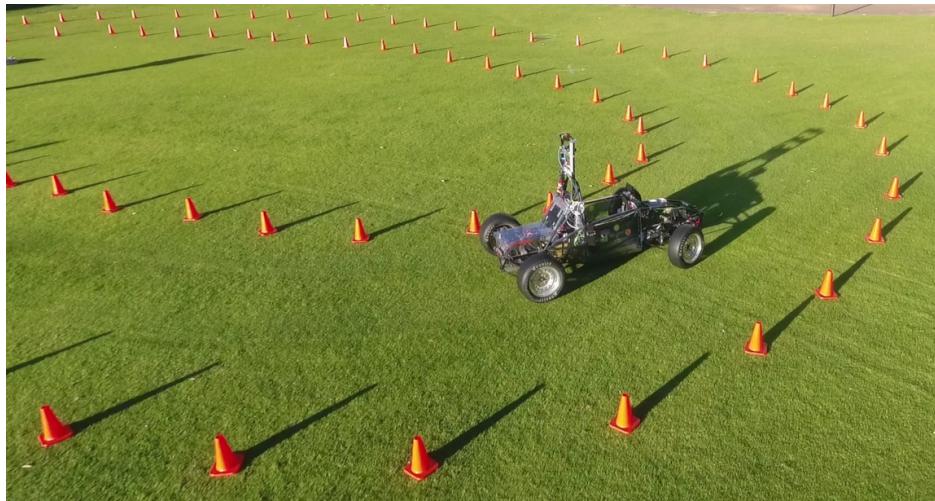


Fig. 7.10 Experimental setup for cone driving.

The system obtains the positions of the cones through the combination of LiDAR and vision processing. Data from the LiDAR is simultaneous to visual cone detection to provide a more robust solution to cone positioning. In our experiments, the speed of the car is limited to 5 m s^{-1} due to safety considerations.

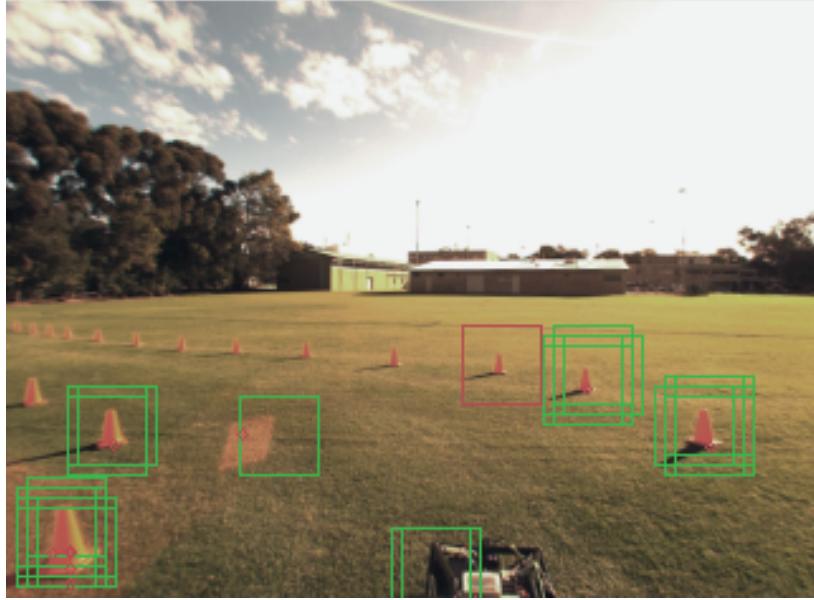


Fig. 7.11 Visual cone detection showing the detected cones in bounding boxes.

To detect cones in the vicinity, we first apply the process described in Section 7.5.3 to find a suitable path to navigate. Fig. 7.11 shows the detected cones in bounding boxes, which are red upon detection turns green as it passes the colour filter. The accuracy of visual cone detection for our initial training and test sets are good, at 96.3% and 92.1% respectively, with its F_1 score as calculated in Table 7.3, which are given in their mean, best and worst cases as measured from each frame across different lighting conditions. These results imply that our classifier is highly accurate under certain conditions and with a mean F_1 score of 0.85, our visual cone detection algorithm is therefore deemed suitable for the system.

Table 7.3 F_1 Scores for Visual Cone Detection

Case	Precision	Recall	F_1
Mean	0.9568	0.7644	0.8499
Best	1.0000	1.0000	1.0000
Worst	0.7500	0.5524	0.6362

Meanwhile, measurements from the LiDAR are used to accurately obtain the relative position of the cones. Its accuracy is verified by comparing it against their ground truth distances. We have thus calculated the mean distance error ε_d to be 21.30 mm with its standard deviation σ_d at 15.49 mm. By evaluating these results against the 5 m cone

distances, we have subsequently deduced that the LiDAR system is adequately accurate for dynamic cone positioning.

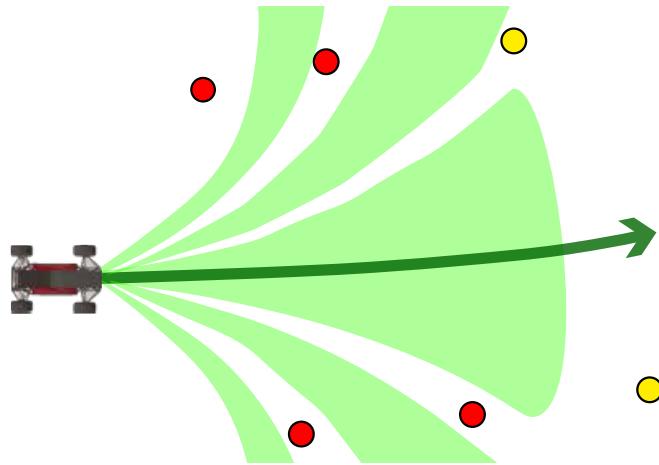


Fig. 7.12 Visualisation of the path planner on cone driving. Immediate cones are red and subsequent cones are yellow; green regions are viable paths.

Paths are generated through the clustering and filtering of LiDAR data. With reference to Fig. 7.12, the vehicle will drive straight following the green arrow in the green region as it has the largest range free of objects. Path planning for cone driving was tested across three sets of recordings N to verify the consistency of path generation across all subsequent frames Σn . Frames with false positives and negatives are considered erroneous frames n_e , where the error percentage e is then calculated with our results given as Table 7.4. With a mean false detection of less than 5% and the erroneous frames at less than 8%. We have therefore deduced from these results that our cone driving algorithm is adequate for autonomous driving. Note that these errors can be further remedied with frame coherence, which is capable of eliminating the classification of stray frames.

Table 7.4 Path Errors

N	n_e	Σn	e
1	7	84	7.14%
2	3	62	4.84%
3	1	24	4.17%

The runtime performances of individual nodes were recorded during our experiments, which are given in percentages as their means and standard deviations for CPU (avgCPU, stdCPU) and memory consumption (avgRAM, stdRAM), as tabulated in Table 7.5. The

sensors' driver nodes make up the largest utilisation percentage, with the image captures occupying over 60% of the CPU footprint to capture a series of 3-channel, 8-bit calibrated RGB image from the camera pair. The LiDARs collectively consume 18% of CPU. The path planning (ConeDetect) and high-level control nodes operate at 20 Hz, with 11% CPU usage.

Table 7.5 Runtime Performances for Cone Driving

Nodes	avgCPU	stdCPU	avgRAM	stdRAM
Camera	63.47%	3.862%	1.5%	0.0%
IMU	7.157%	0.5182%	1.9%	0.0%
LUX	9.955%	0.6802%	1.7%	0.0%
LMS111	7.586%	0.4830%	0.30%	0.0%
ConeDetect	7.605%	0.7858%	0.38%	0.040%
roscore	0.1354%	0.1909%	0.88%	0.61%
control	3.557%	0.2461%	0.30%	0.0%

7.7.4 Driving Simulation

The effectiveness of the simulation system was measured through the drawing of comparisons to results gathered from testing of the SAE vehicle. Given the current work involving autonomously driving a traffic cone delineated race track, a focus was placed on the relative accuracy of the LiDAR and visual cone detection systems compared to results gained from test drives of the SAE vehicle. Comparisons were made by recreating a cone track on a flat plane in the simulation system and performing visual comparisons of the results. A comparison of the tracks used is displayed in Fig. 7.13.

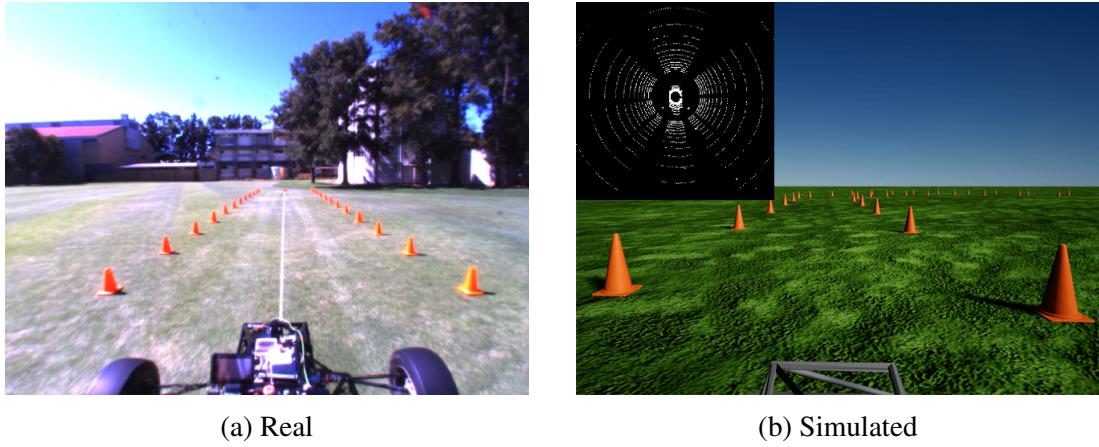


Fig. 7.13 Scenarios used for comparing (a) real and (b) simulated LiDAR and visual cone processing outputs.

From Fig. 7.14, the output from the simulated LiDAR is significantly more detailed than that on the SAE vehicle. As such, the original LiDAR output from the simulator (white points) was cropped to simulate the LiDAR available on the SAE vehicle through the use of ROS' “pointcloud_to_laserscan” package, resulting in the 2D laser scan data displayed in green. While these laser scans are not identical, with the real data displaying the ground on the far right as a result of the uneven terrain, these figures show that the cone locations identified are sufficiently similar to allow testing of higher level components (e.g. path planning) on the simulated system.

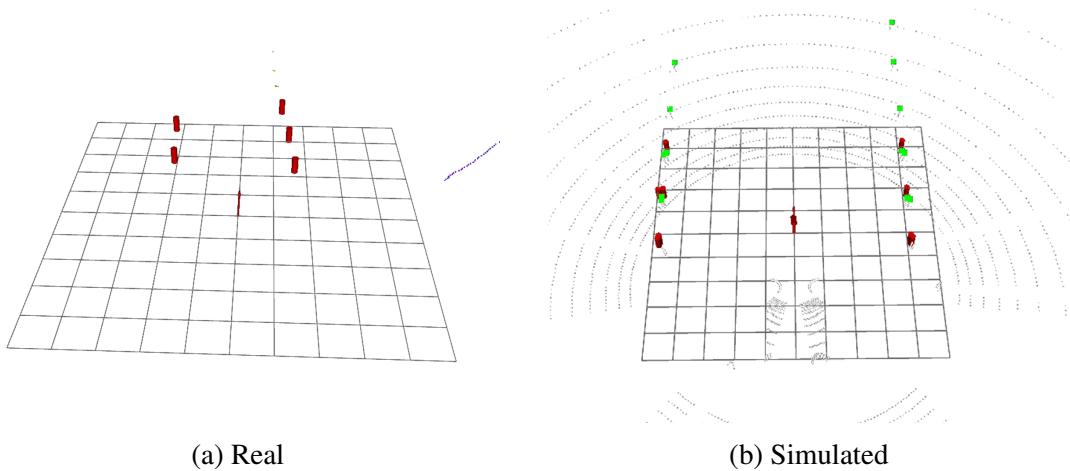


Fig. 7.14 LiDAR cone processing outputs from (a) real and (b) simulated scenarios.

7.8 Conclusion

We have presented a software framework for a high-level control system that is designed for autonomous vehicles that is both modular and scalable. Our design approach using open-source software with commercially available sensors and parts hopes to encourage similar projects especially in academia where we have fitted a student competition vehicle for full autonomous driving. These projects can therefore be low-cost while allowing users to adapt the software to the vehicle's and environment's needs. This system aims to be holistic by incorporating all the necessary modules required for autonomous driving, including sensor interfaces and fusion, localisation, path planning, visual navigation and road detection; as well as cone driving and an identical driving simulator for both real-world and simulated tests. It is therefore easily deployable while requiring minimal configuration. Experiments on path planning, cone driving and the simulator proved that this system is robust and adequate for implementation. We are eager to correspond with any entities who wish to incorporate our approach in their respective projects.

Chapter 8

Hardware-in-the-Loop Autonomous Driving Simulation without Real-Time Constraints

Simulation is a cornerstone of autonomous driving efforts, allowing testing to occur more rapidly and with significantly less risk than is possible with hardware platforms alone. Simulation systems must be able to emulate a variety of sensors including cameras and LiDARs in order to allow high-level software such as image processing and path planning to be tested. In this paper, we present a hardware-in-the-loop (HIL) simulation system without real-time constraints. It is based on CARLA to give access to the sensors required to test high level software, and incorporates compute hardware identical to that used on an autonomous vehicle platform in order to provide realistic constraints regarding available processing power. In addition, we explore the Robot Operating System (ROS) based software framework used on the Formula SAE (FSAE) vehicle and its integration with the driving simulator. Finally, we validate the sensor outputs and vehicle dynamics of the simulated system against a physical autonomous driving hardware platform.

8.1 Introduction

The Renewable Energy Vehicle (REV) Project at UWA is currently focused on the development of autonomous driving applications. This development has occurred predominantly on a hardware platform consisting of an FSAE [289] race car converted to an electric drive equipped with an ibeo LUX LiDAR, an Xsens MTi-G-710 combined inertial measurement unit (IMU)/global positioning system (GPS) and a number of cameras for sensing, an Nvidia

Jetson TX1 for compute, and full drive-by-wire capabilities. The goal for the driverless FSAE project is to increase the level of autonomy of the vehicle as it drives around a race track, from relying on waypoints placed manually through a Google Maps driven web interface, to relying solely on input from the variety of sensors available on the vehicle. This should result in the vehicle being capable of driving and mapping a semi-structured race track (with edges delineated by either cones, as displayed in Fig. 8.1, or road edges) with no prior knowledge before generating an optimised path and re-driving the track at a greater speed.

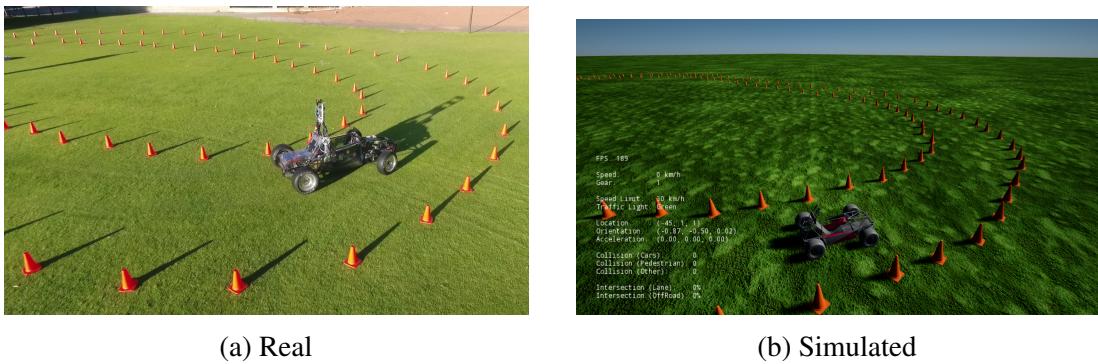


Fig. 8.1 Track setup used for comparing real (a) and simulated (b) LiDAR and visual cone processing outputs.

This platform is equipped with a variety of hardware safety systems and provides a number of advantages, such as being mechanically simple (resulting in low maintenance requirements and allowing for modifications to be made with ease) and being able to provide ample electrical power to the onboard sensors and hardware. However, given that it is a 250 kg vehicle capable of speeds up to 80 km/h, it is subject to onerous safety requirements. In addition, it is not a road-licensed vehicle, preventing testing on public roads. These issues provide the motivation for the development of an HIL simulation system without real-time constraints, designed to allow more frequent testing in a wider range of environments with minimal risk while still presenting the same constraints on the available compute hardware.

The viability of the HIL simulation system outlined in this paper, especially in terms of ease of integration and maintainability, was influenced greatly by the current software framework utilised on the FSAE vehicle. As such, this software framework will also be explored in this paper.

In order for simulated testing to be effective, a high degree of similarity in vehicle dynamics and sensor outputs must be maintained between the simulated and real systems. As a result, validation of these aspects of the HIL simulation system against a physical autonomous driving platform was undertaken.

The contributions of this paper are: developing an integration between CARLA and ROS; verifying the correlation between real and simulated sensor measurements; establishing and verifying that neither simulation nor real vehicle require hard real-time constraints to operate; and quantifying the time saved through use of simulation experiments as an initial step before experiments on real vehicles.

The remainder of this paper is organised as follows. Section 8.2 introduces the current software framework utilised on the FSAE vehicle. Section 8.3 introduces the open-source software used as the basis for the driving simulator, and details the integration with the software framework utilised on the FSAE vehicle. Section 8.4 outlines the sensors available on the FSAE vehicle and the equivalent sensors available in the simulation software, along with the cone detection and path planning algorithms currently available on the FSAE vehicle. Section 8.5 presents our experiments and results, followed by potential future works in Section 8.6. Finally, concluding remarks are drawn in Section 8.7.

8.2 Software Framework

This section introduces the software framework currently utilised on the FSAE car. Given the use of this software as a development platform in an autonomous vehicle, it was imperative that it be flexible and extensible, allowing for new components to be easily integrated, as well as resilient to software failures. This set of requirements led to a publish/subscribe software architecture being utilised, as it allows for highly decoupled software to be developed with a minimal set of shared dependencies, with each component (or series of components) needing only to conform to the expected input and output message types. By extension, this also allows for components to be developed in different languages depending on their importance and required performance characteristics, reducing the development time for simple, non critical components. The use of a publish/subscribe architecture also allows components to be swapped in and out, simplifying the testing of different solutions, and providing simple methods of logging, data capture and data replay that do not require modifying each component individually.

Based on the success of the Apollo Auto project [291], it was decided to use ROS [303] as a base system to provide the desired publish/subscribe functionality due to the resilience and performance that it displays. In addition, this provides a series of potential future upgrades, from transitioning to the Apollo platform [321] for improved performance due to shared memory transport for message passing, Protobuf message support, and decentralisation to reduce single points of failure, to adopting the complete Apollo Auto platform should access to a supported hardware platform eventuate. The usage of ROS now ensures that

any components developed will be compatible at any stage in this upgrade path, while also providing access to a large library of existing components and libraries, minimising the amount of supporting code and number of utilities that must be developed by the group to support common activities.

The nodes and message passing required to meet the current goals of the REV Project were heavily influenced by the software modules and interactions defined in the software architectures of two open-source autonomous driving platforms, Apollo Auto [291] and Autoware [292]. Apollo Auto is a Baidu-led project with partners including major automotive manufacturers such as Ford and autonomous driving hardware suppliers such as Nvidia and Velodyne. The project is aiming to implement full autonomy on highways and urban roads by the end of 2021, and is already used in production on over 100 autonomous shuttles operating in closed venues [322]. Autoware is a similar project which has been adopted by over 100 companies, and which is qualified to operate driverless vehicles on public roads in Japan.

The high-level architectures of these projects, one of which is presented in Fig. 8.2, display significant similarities in the flow of data and controls throughout each system. This starts with environment data from sensor suites, followed by localisation and perception modules, with the processed data then being utilised by planning and control modules. The software framework developed for the FSAE vehicle was designed around the commonalities of these architectures, with appropriate adaptations being made to ensure compatibility with the hardware available on the FSAE vehicle. For example, due to the lack of sufficiently high resolution LiDARs, adaptations have been made to remove the need for HD mapping.

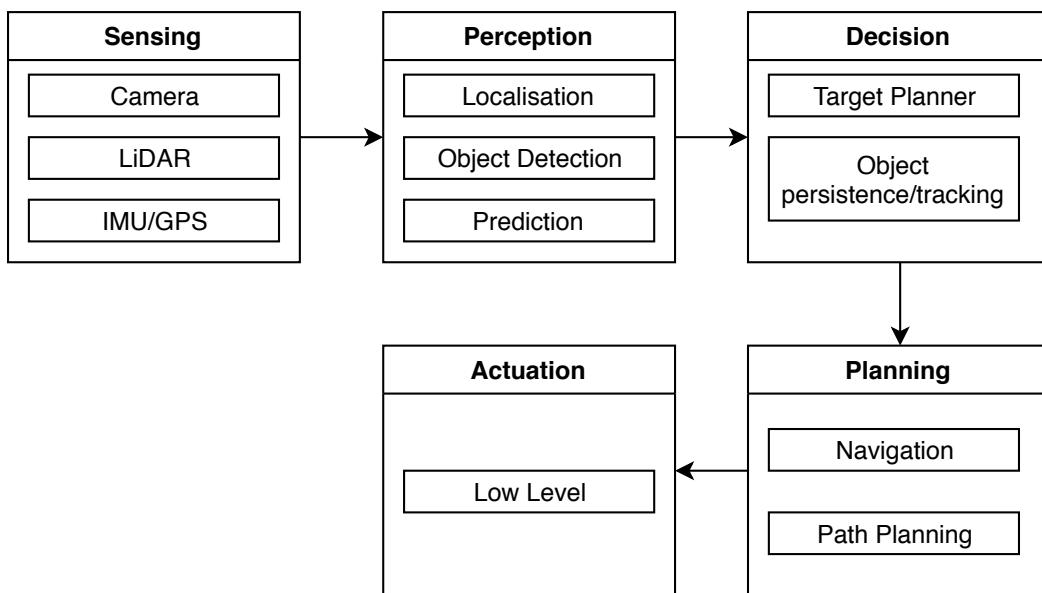


Fig. 8.2 High-level architecture of the Autoware project, adapted from [323].

Due to the flexibility of the publish/subscribe-based framework, this is an outline of the required functionality only, with the potential for some of the nodes displayed to be broken down into a series of smaller components. For instance, development of a variety of “Object Filter” nodes can be hastened by dividing it into two nodes as demonstrated in Fig. 8.3, one to simply combine the incoming object arrays into a single array (“Object Array Concatenation”), and another to perform the filtering (“Object Filter”). Development of nodes in this manner provides additional benefits, such as requiring only a single component to be modified should a new source of object data be introduced, such as a radar system.

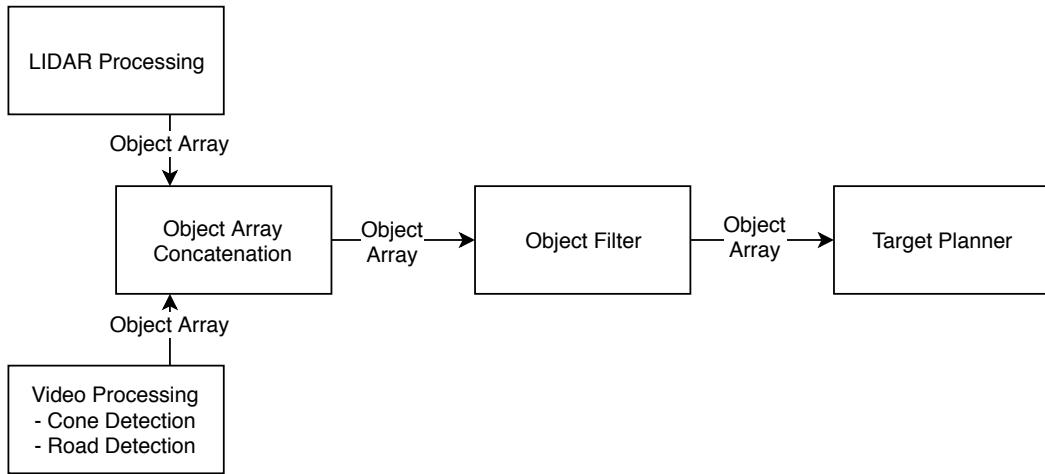


Fig. 8.3 Division of ROS node responsibilities.

8.3 Driving Simulator

Driving simulation systems have long been a cornerstone in efforts to lower development costs for advanced driver assistance systems [324–326] and are used extensively by major automotive manufacturers [327–329].

Given the potential for uncertainty in the regulatory landscape to slow down progress into autonomous vehicles [330], companies such as Waymo [130], Cognata [331], rFpro [332] and Nvidia [333], as well as an open-source collaboration between Intel, Toyota and the Computer Vision Center [318], are extending on these ideas by developing autonomous driving simulators in order to allow autonomous vehicles to be trained and tested without regulatory hurdles. In addition, driving simulators allow testing of a series of predefined scenarios to be performed quickly [334], as well as allowing testing to occur for scenarios not regularly encountered during driving. These systems have become cornerstones of autonomous vehicle testing, evidenced by Waymo having simulated 4.3 billion kilometres of driving in 2017 alone.

While the REV Project has made use of an autonomous driving simulator in the past [335], it was decided that the lack of support for LiDARs as sensors, the outdated graphics, and the complexity involved in developing integrations with external systems provided sufficient reason to move to a more modern simulation platform. As such, the driving simulator developed is centred around the CARLA open source driving simulator [318], due to its providing the desired sensors and customisable scenarios without prohibitively expensive licensing or hardware requirements. By default, CARLA provides access to data from a configurable suite of sensors including cameras and LiDARs along with information regarding the current pose, velocity and acceleration of the simulated autonomous vehicle through a Python API. In addition, CARLA provides access to information regarding other simulated agents, allowing for the automated verification of results, and is developed in Unreal Engine [336], a popular gaming and simulation engine, which ensures that tools and resources are available for any future modifications to the system.

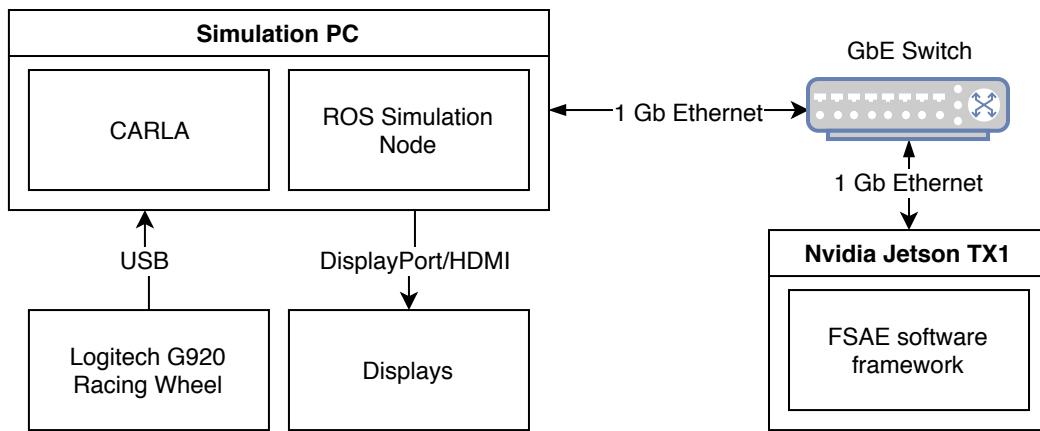


Fig. 8.4 Autonomous driving simulator hardware diagram.



Fig. 8.5 Autonomous driving simulator setup.

The autonomous driving simulator consists of a computer (Intel i7-4770 CPU, 8GB DDR3-1066 RAM, Nvidia Titan X (Maxwell) GPU) running the aforementioned CARLA driving simulator and ROS simulation node which receives input from the Logitech G920 racing wheel over a USB connection, and performs bidirectional communication with an Nvidia Jetson TX1 over a network connection, as shown in Fig. 8.4. The inclusion of HIL allows a much narrower “reality gap” than a purely software-based system, as all vehicle responses are “real” in a sense that they come from the actual drive computer at the correct “real” vehicle timing conditions. With this set up, we are able to run CARLA at upwards of 30 frames per second (FPS) without impacting the performance of the FSAE software running on the Nvidia Jetson TX1 for testing in simple environments. Note that this simulation system was established without the need for real-time constraints, beither on the simulation side, nor on the real vehicle control side. Sensor data will be provided to the connected hardware when calculated and does not have guaranteed response times. This setup is displayed in Fig. 8.5 along with the displays and racing seat used to provide a realistic driving environment.

The interface between the FSAE vehicle software framework and the CARLA driving simulator was greatly simplified due to the choice of ROS as a basis. ROS allows for inter-device communication over a network connection, and the software framework detailed in Section 8.2 allows for components or groups of components to be trivially swapped in and out. This resulted in the interface consisting of a single ROS node written in Python which retrieves sensor and environment data from the CARLA application programming interface (specifically, two camera feeds, a LiDAR point cloud and pose and velocity information of the vehicle), and published this information to the topics expected by the video processing and LiDAR processing nodes. The node then receives control data from the navigation/path planning node, which is used to create the control objects expected by CARLA for driving

the simulated vehicle. This node acts in the place of the fusion and localisation, camera, LiDAR and low level nodes presented in Fig. 8.6, resulting in the application architecture seen in Fig. 8.7.

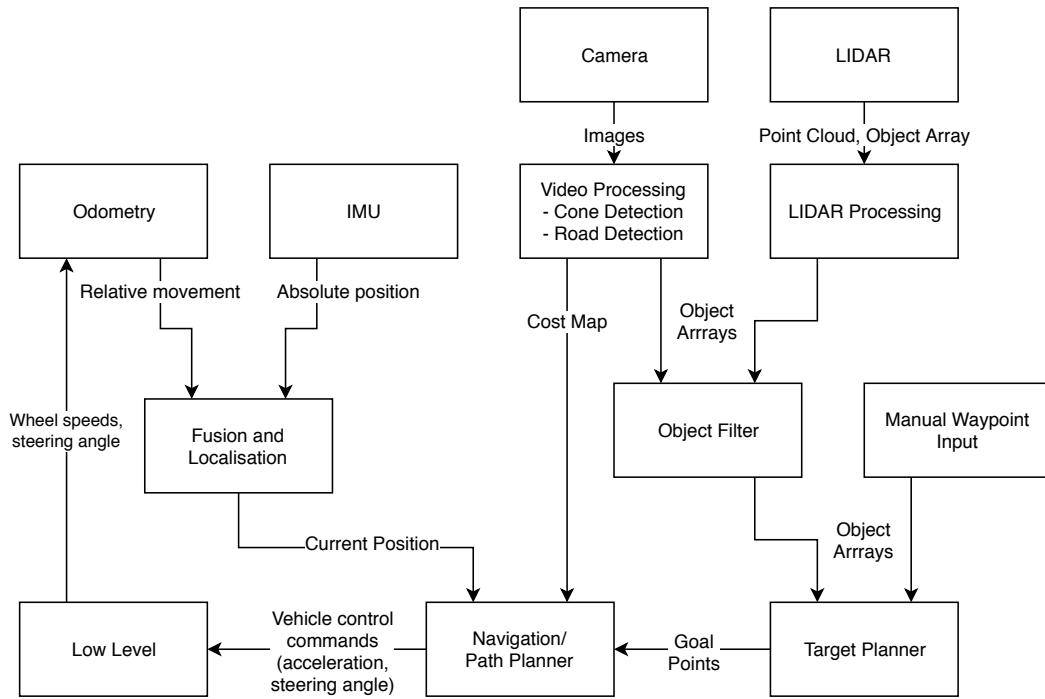


Fig. 8.6 FSAE vehicle ROS node structure.

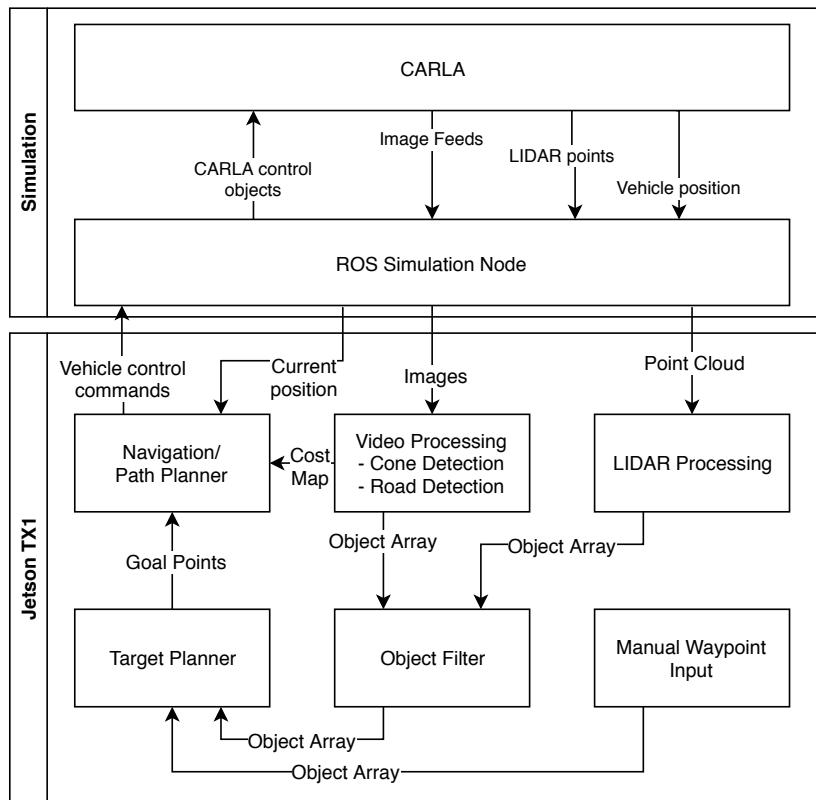


Fig. 8.7 Software framework architecture with CARLA simulator interface.

The inter-device communication enabled by ROS is critical in allowing realistic compute hardware to be used in the autonomous driving simulator. This allows the simulation software and results validation to be performed on a secondary device while the FSAE software is run on an Nvidia Jetson TX1, identical to the system installed on the FSAE vehicle. This allows for a high quality simulation to be run without negatively impacting the high-level software performance, while still presenting similar performance constraints to the high-level software. This ensures that software tested successfully on the simulated system will be able to perform almost identically on the real FSAE vehicle, which is verified in Section 8.5.4.

In replacing the low-level node, the simulator interface also assumed responsibility for emulating a number of the safety systems provided by the low level controller, such as allowing for manual intervention to override the autonomous systems. This was achieved by handling manual input through either a keyboard or Logitech G920 racing wheel [319], and replicating the low level system's response to these inputs in the simulator interface. It is also possible to connect a small touch-screen display to the Nvidia Jetson TX1 to provide an interface to the autonomous systems identical to that found on the FSAE vehicle, allowing for user interface testing to occur in a safe environment.

The open-source nature of the CARLA simulator provides additional benefits, such as an active community to provide troubleshooting and contribute features and performance and stability improvements to the project. To date, the primary benefit has been in the ability to create custom scenarios and import custom object meshes, however there is scope for actions such as creating new types of sensors should the need arise.

8.3.1 Performance and Suitability for Wall-clock Time Operation

The control system on the physical FSAE vehicle uses ROS and is not a hard real-time system in a sense that it does not guarantee fixed reply times. This has not been necessary for controlling the vehicle, as all sensor data is received asynchronously at various update rates. Consequently, there are no real-time requirements for the simulation system either, provided that the simulated sensor data can be provided at a similar and sufficiently high frame rate. It is worth noting that both Apollo Auto as well as Autoware are ROS-based and are not real-time systems either. Instead, it is simply demonstrated that the simulation system is capable of generating data at a rate which exceeds the rate at which the software framework consumes it, ensuring that the software framework receives new data with every request. The rates at which data is generated by sensors and consumed by various key nodes from the software framework are presented in Table 8.1.

Table 8.1 Comparison of real and simulated update frequencies of sensors and key ROS nodes.

Sensor or Node	Update Frequency (Hz)	
	Physical	Simulation
ibeo LUX LiDAR	10	45
SICK LMS111-1010	50	45
FLIR Blackfly GigE camera	10	45
Vehicle odometry	30	45
Control node	30	30
Cone detection node	10	10

By comparison, the simulation system on average generates and publishes data at a rate of 45 FPS, with 95% of frames generated at a rate of at least 24 FPS, as seen in Fig. 8.8. While the ibeo LUX LiDAR generates data at a higher frequency, this data is consumed by the cone detection node at a rate of only 10 Hz, significantly lower than the rate at which the

simulation system provides sensor data. As such, the simulation system is able to operate effectively without a hard real-time requirement.

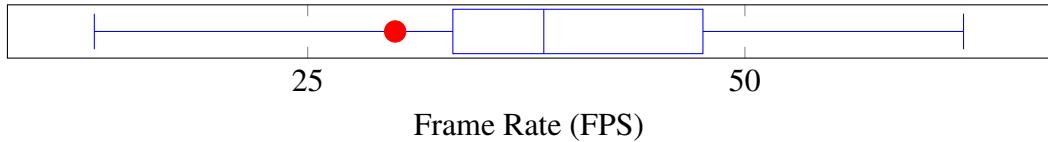


Fig. 8.8 Comparison of frame rates achieved by CARLA (blue) with the control node update frequency (red).

8.3.2 Time Synchronisation

Within ROS, the header of each message sent contains a timestamp generated based on the source computer's clock, moving the responsibility of time synchronisation to the operating system. At present, rough time synchronisation between the simulation and autonomous driving software nodes relies on a shared, remote NTP server. This has resulted in a clock error of 35ms between the simulation and autonomous driving nodes as measured by Ubuntu's `clockdiff` utility [337].

8.3.3 Simulation Benefits

The establishment of a reliable and accurate simulation system provided significant time savings during the development of high-level perception and planning modules by significantly reducing the time required to test each iteration of a software module. It achieved these reductions by significantly reducing the time to setup and breakdown test scenarios. The approximate time savings that resulted from the availability of a simulation system are presented in Table 8.2. From this, it can be seen that the simulation system saves around 110 minutes or almost two hours per test, allowing tests to be performed more regularly. In addition, the use of the simulation system allows testing to be carried out by an individual where physical testing requires a minimum of three persons to satisfy safety requirements, resulting in a saving of around 330 man-minutes per test, and was unaffected by any hardware faults such as sensor failures. In essence, each software error caught by the simulation system saved at least five man-hours of testing overhead, as it prevented issues that could render a physical test drive ineffective.

Table 8.2 Approximate autonomous driving test times.

Task	Time Required (minutes)	
	Physical	Simulation
Moving vehicle to appropriate test location	60	0
Setting up test track/environment	15	5
Breaking down test track/environment	15	5
Returning vehicle to storage	30	0
Total	120	10

8.4 Sensors, Navigation and Path Planning

The FSAE car uses a combination of sensors (displayed in Fig. 8.9) including LiDARs, cameras, wheel odometry and an IMU, which are divided into four categories: a camera system, dead reckoning, a LiDAR system and odometry. The driving simulator provides replacements for the LiDAR and camera systems, and makes the dead reckoning and odometry systems obsolete by providing exact positioning in the simulated environment, allowing focus to be placed on camera and LiDAR systems with the guarantee that the data regarding the cars positioning will be correct. This section will demonstrate the sensors that are made available by CARLA, and draw comparisons to the sensors currently installed on the FSAE vehicle. It will also introduce the path planning algorithm which has been used for testing the simulator.

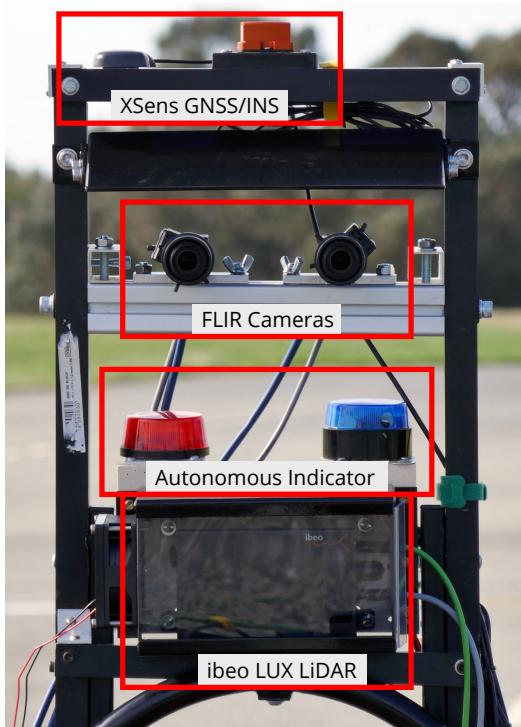


Fig. 8.9 Autonomous FSAE sensor rack.

8.4.1 LiDAR System

The cone detection algorithm currently utilised on the FSAE vehicle, detailed in Algorithm 3, relies on a single front-mounted 2D SICK LiDAR [255], providing a 270° horizontal field of view with an angular resolution of 0.25–0.5° and a range of up to 20 m. CARLA provides a 360° LiDAR with a configurable range, number of channels, rotation frequency, and upper and lower field of view limits [338]. In order to emulate the LaserScan [339] data that is available through SICK’s LMS1xx ROS driver [340], the `pointcloud_to_laserscan` ROS package [341] was used in conjunction with a CARLA LiDAR configuration with a narrow vertical field of view. This combination allows a LaserScan to be produced with a configurable horizontal field of view and angular resolution, hence matching the output produced by the physical LiDAR.

Algorithm 3 LiDAR cone detection

```

1: procedure LiDARCONEDETECTION(laserscan)
2:   crop laserscan to [-90,90] degree range
3:   for all point in laserscan do
4:     if point is first point then
5:       init a cluster with point
6:     else
7:       from (this_point, last_point)
8:       evaluate magnitude, orientation
9:       if (magnitude or orientation) > threshold then
10:        start new cluster with point
11:      else
12:        add point to current cluster
13:      end if
14:    end if
15:   end for
16:   for all cluster in laserscan do
17:     cone_center = center_point of cluster
18:     cone_size = point(nearest) - point(furthest)
19:   end for
20: end procedure

```

8.4.2 Camera System

The FSAE vehicle is currently equipped with two FLIR Blackfly GigE [309] cameras. Each of these cameras has a maximum resolution of 1288×964 pixels and is capable of working at 30 FPS. The cameras make use of a global shutter, removing the need for the compute hardware to perform compensation for rolling shutter effects [342], such as those presented in [311]. CARLA provides a direct analogue for these cameras in the form of the “scene final” cameras [338]. These are also global shutter cameras, and CARLA provides facilities to configure the field of view, resolution and position of the cameras. Using this feature, the simulation is configured to output two image streams at a resolution of 1288×964 pixels each, placed the same distance apart to mirror the FLIR camera setup displayed in Fig. 8.9. The frame rate of the cameras provided by CARLA is tied to the frame rate that the CARLA simulator is run at. While it is possible to set a static FPS target in CARLA, this prevents the simulator from running in wall-clock time which is incompatible with our FSAE software

framework. Instead, images are published to the software framework at the same FPS that CARLA runs at, with only the latest received image being stored. The visual cone detection node, detailed in Algorithm 4, then retrieves this image as needed at a frequency of between 15 Hz and 30 Hz.

Algorithm 4 Visual cone detection

```

1: procedure VISUALCONEDETECTION(image)
2:   crop image to 64 x 64 patches with a stride = 4
3:   for all patches in image do
4:     evaluate feature_vector
5:     pass feature_vector to SVM_classifier
6:     if not patch has cone then
7:       continue
8:     end if
9:     threshold the hue_layer to match color
10:    apply histogram to hue_mask on (x, y)
11:    histo(peak) = center(cone) in camera_frame
12:    project center_point onto ground_plane
13:    projected_point = cone_location in global_frame
14:   end for
15: end procedure
  
```

8.4.3 Path Planning

The FSAE control system has been implemented to deliver path planning routines to allow either driving through a series of predefined waypoints, or in between a series of traffic cones placed on either side of the vehicle. This paper will focus solely on the cone driving scenario, as this presents a higher level of complexity and processing power in order to thoroughly test the simulation system.

The current iteration of the path planning procedure uses obstacle detection of the cones to determine the correct path. The current code uses the same as [302], but simplifies it to allow for quicker calculation. Our cone driving module accepts cone locations from either the map, LiDAR or camera, classifying them as objects. Then, the vehicle navigates to drive within the track formed by cones safely without collision. Using a range of the maximum turning circle of the car, of both a left-hand turn and right-hand turn, it then looks at which predicted paths will intercept cones. The vehicle dynamics is thus limited during motion planning whereby the steering angle does not exceed 25°. Our algorithm will iterate through

all cones within the car's range and calculate the best collision-free path to undertake, as detailed in Algorithm 5.

Algorithm 5 Path planning

```

1: procedure CONEDRIVE(cones in range)
2:   init steering_range to [-25,25]
3:   for all cones in range do
4:     evaluate collision_range with cone
5:     exclude the collision_range from steering_range
6:   end for
7:   if steering_range is empty then
8:     stop
9:   else if all steering_range  $\leq$  threshold then
10:    select largest steering_range
11:   else if all steering_range  $>$  threshold then
12:     select steering_angle with minimum change in current direction
13:   end if
14:   drive toward centre of steering_range
15: end procedure
  
```

8.5 Experiments and Results

Verification of our simulation system came in the form of experiments related to the accuracy of simulated vehicle dynamics, LiDAR and vision-based cone detection, processor load monitoring, and system response times, detailed in Sections 8.5.1 through 8.5.5.

8.5.1 Vehicle Dynamics

This experiment aimed to verify that the dynamics of the simulated vehicle were comparable to those of the physical FSAE vehicle. This was achieved by having both the real and simulated vehicles perform a prescribed set of actions, and recording the path driven by the vehicle. Two simple scenarios were performed: driving in a straight line for a fixed period of time, and driving in a circle with a fixed steering angle and speed. In the straight line scenario, the vehicle started from stationary, accelerated to a fixed speed of 3 m/s, and fully applied the brakes at the five second mark until the car returned to stationary. During this manoeuvre, the FSAE vehicle travelled a distance of 13.4 m on average. In comparison,

the simulated vehicle travelled, on average, a distance of 13.0 m, which is within 3% of the FSAE vehicle. In the steering angle scenario, the vehicle was driven at 3 ms^{-1} with a number of fixed steering angles, and the radius of the turning circle measured. The results of these scenarios for both the real and simulated vehicles are presented in Table 8.4, in which it can be seen that the radius of the turning circle of the simulated vehicle is within 5% of the physical vehicle on average. While there is some error present in the vehicle dynamics exhibited by the simulation system, the constant feedback loop during autonomous operation works to minimise the effects of these differences.

Table 8.4 Real and simulated turning radii for varying steering angles.

Steering Angle	Turning Radius (m)	
	Real	Simulation
10	10.3	10.4
15	7.0	7.3
20	5.3	5.7
25	4.3	4.6

8.5.2 LiDAR Cone Detection

The aim of this experiment was to verify that the LiDAR output obtained from the simulation using the method described in Section 8.4.1 was sufficiently similar to that generated by the SICK LiDAR available on the FSAE vehicle. This is required in order to allow cone detection algorithms to be tested on the driving simulator with a high degree of certainty that the results will be transferable to the SICK LiDAR. This was achieved by simulating a scenario mimicing that of a previous test of the FSAE vehicle, and verifying that a similar set of cones was detected by the same algorithm used in the FSAE vehicle test. Using the real and simulated scenarios presented in Fig. 8.10, the outputs of the cone detection performed on each scenario, displayed as red cylinders in Fig. 8.11, are sufficiently similar to allow testing of higher level components such as path planning and object avoidance on the simulated system.



Fig. 8.10 Scenarios used for comparing real (a) and simulated (b) LiDAR and visual cone processing outputs.

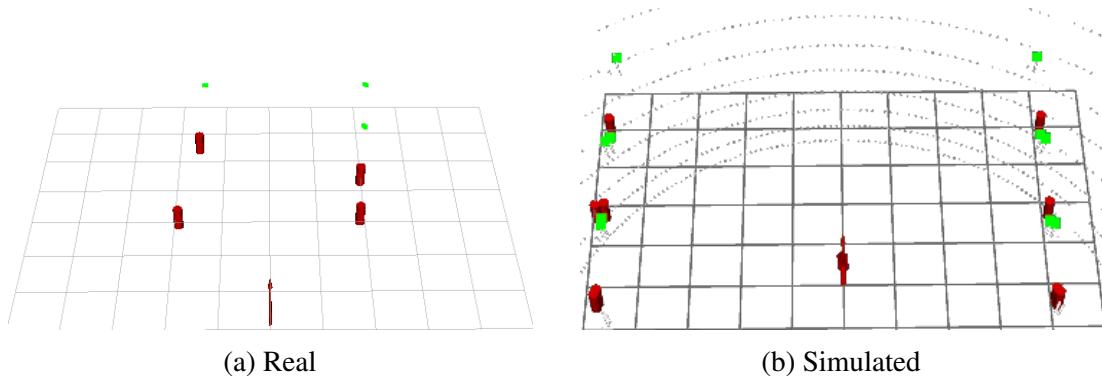


Fig. 8.11 LiDAR-based cone detection results from real (a) and simulated (b) scenarios. The raw LiDAR data is displayed in green, and detected cones are displayed in red.

In addition, the LiDAR accuracy was verified through a direct comparison of LiDAR point clouds from the real and simulated LiDARS on a known cone layout, the results of which are displayed in Fig. 8.12. With the exception of some minor inconsistencies which can be attributed to errors in measurement of the physical cone layout and minor differences between the physical cones and simulated cone models, the simulated point cloud is seen to directly overlay the real point cloud.

As seen in Algorithm 3, the LiDAR-based cone detection operates by clustering the LiDAR points, and classifying them as cones based on the number of points in the cluster, which has been tuned to the maximum distance at which cones are detected reliably based on the resolution of the SICK LiDAR. In order to ensure that this algorithm is also effective on the simulated LiDAR output, the number of points returned by objects of various sizes and reflectivities at set distances was measured and compared. As can be seen in Fig. 8.13, the number of points returned by the simulated LiDAR is within $\approx 15\%$ of the physical



Fig. 8.12 Comparison of LiDAR point clouds from real (green) and simulated (red) scenarios.

LiDAR, and demonstrates a realistic decrease in the number of points returned by an object as distance increases.

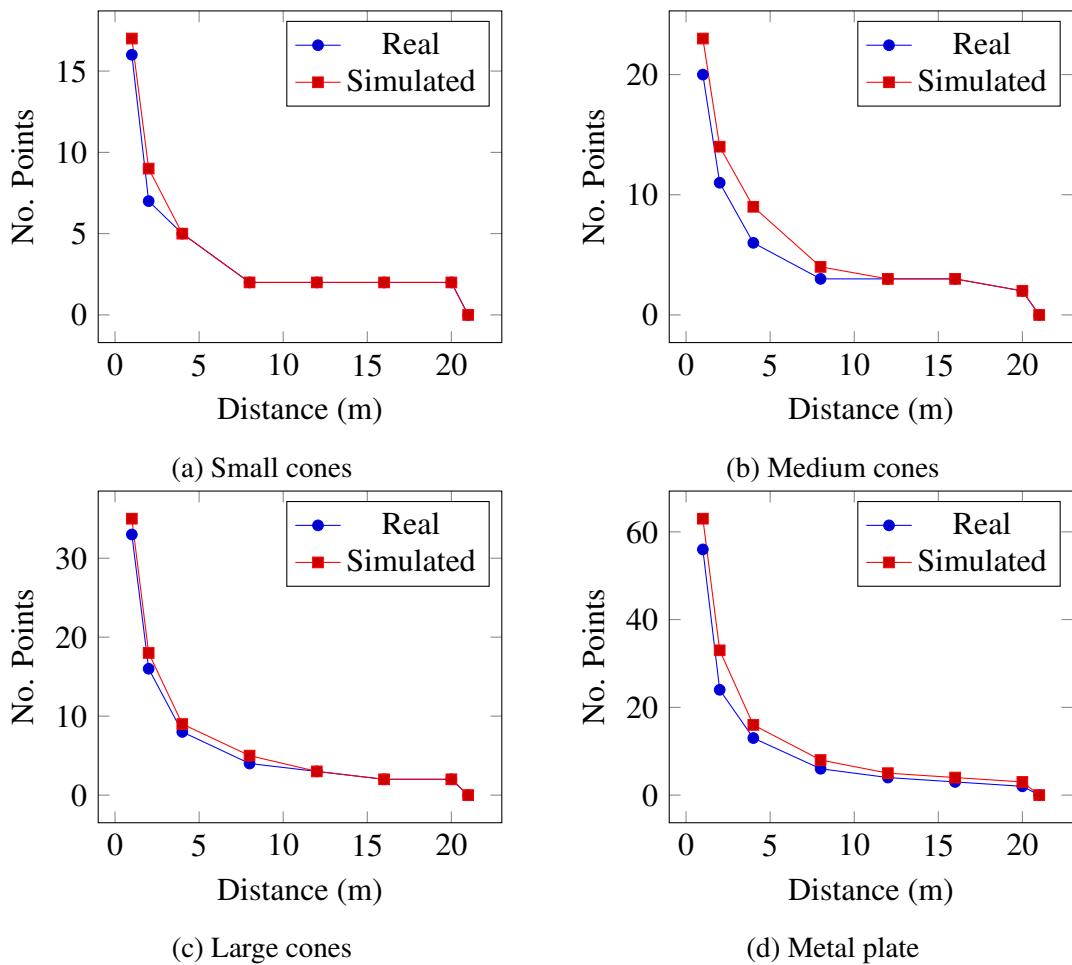


Fig. 8.13 Number of LiDAR points returned by small cones (a), medium cones (b), large cones (c) and a metal plate (d) at varying distances.

The influence of LiDAR noise on the cone detection algorithm was tested by comparing the detection rates of the algorithm on the default simulated LiDAR output, which shows perfect precision for well constructed collision boxes, and the same output to which artificial noise has been introduced. This was achieved by passing the LiDAR data through a filter which modifies the distance of each point over a Gaussian error distribution typical of LiDAR sensors. In both scenarios, an average detection rate of $\approx 95\%$ was observed, with a minimum detection rate of $\approx 92\%$. This conforms to the expected behaviour, as the typical error of the SICK LiDAR (± 30 mm) is significantly smaller than the cutoff magnitude used in the cone detection algorithm, and so would not be expected to affect the point clustering.

8.5.3 Visual Cone Detection

This experiment aims to verify that the images available through CARLA's camera sensors (described in Section 8.4.2) are sufficiently similar to those generated by the FLIR Blackfly GigE cameras installed on the FSAE vehicle that a visual cone detection algorithm is capable of producing similar results on both images. A comparison of these results is given in Fig. 8.14. From this, it can be seen that cones are identified successfully, however with a decreased range on the simulated image. It is expected that the detection range on simulated images could be increased by incorporating some simulated images into the training set.

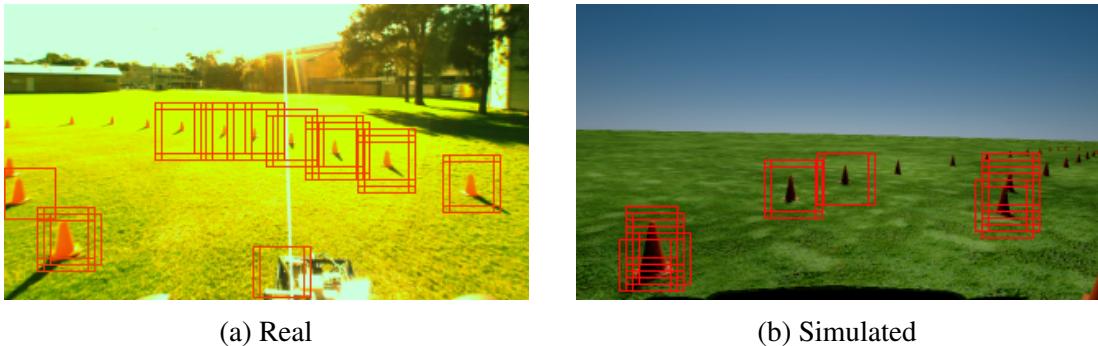


Fig. 8.14 Computer vision based cone detection results from real (a) and simulated (b) scenarios.

The experiment also aims to demonstrate that the performance of computer vision code is similar between real images and those generated by the simulator, which was achieved by monitoring the per-frame runtime performance of the cone detection algorithm currently in use on the FSAE vehicle. These times are presented in Fig. 8.15, from which it can be seen that the average processing time for the computer vision algorithm on simulated images is on average only 0.005% faster than the average processing time for real images, and the

median of the processing time on simulated images is 0.003% slower than for real images, confirming the similarity in the computer vision code performance.

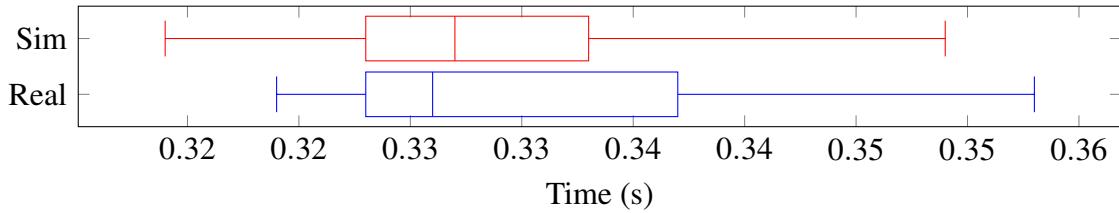


Fig. 8.15 Cone detection algorithm processing times for real and simulated images.

The detection rates of the computer vision based cone detection algorithm for real and simulated scenarios is presented in Table 8.5. This includes both the raw simulated output, along with images that have had post processing applied to introduce Gaussian-distributed additive noise. This was achieved by adding values from a Gaussian distribution with $X \sim \mathcal{N}(0, 20^2)$ to each of the RGB channels of each pixel of the image. The divergence of the detection metrics of simulated images from real images is presented in Table 8.6. Averaged across all frames, the average divergence of the simulated images from the real across the three detection metrics is 8.3%, which increases to 8.7% when noise is introduced.

Table 8.5 Comparison of computer vision based cone detection rates for real and simulated images.

Image Source	Case	Precision	Recall	F_1
Real	Mean	0.9568	0.7644	0.8499
	Best	1.0000	1.0000	1.0000
	Worst	0.7500	0.5524	0.6362
Simulation	Mean	0.9017	0.8867	0.8773
	Best	1.0000	1.0000	1.0000
	Worst	0.6000	0.5000	0.6667
Simulation (w/ noise)	Mean	0.8246	0.8000	0.7845
	Best	1.0000	1.0000	1.0000
	Worst	0.5000	0.5000	0.5000

Table 8.6 Divergence of detection metrics from real images for simulated images.

Image Source	Case	Precision	Recall	F_1
Simulation	Mean	5.8%	16.0%	3.2%
	Best	0%	0%	0%
	Worst	20.0%	9.5%	4.8%
Simulation (w/ noise)	Mean	13.8%	4.7%	7.7%
	Best	0%	0%	0%
	Worst	33.3%	9.5%	21.4%

8.5.4 Compute Hardware Load

This experiment was designed to verify that the use of identical compute hardware (an Nvidia Jetson TX1) in the simulation loop resulted in similar performance constraints to those presented by the FSAE vehicle platform. This verification comes in the form of a comparison of the system resources used in the real and simulated systems while performing a similar task, in this instance, LiDAR-based cone detection. Results were gathered by running the `sysstat` performance monitoring tools for Linux [343] while the cone detection algorithms were operating on both systems. This utility captured the percentage of the processor utilised by user and system processes at a frequency of 1 Hz for 120 seconds to allow an average to be computed, the results of which are displayed in Fig. 8.16. From this figure, it can be seen that the hardware in the simulation loop had consistently higher processor utilisation for user space processes, with an average of $\approx 25.2\%$ (Fig. 8.16a) compared to $\approx 20.6\%$ (Fig. 8.16b) for user space processes on the real system. Given that in both scenarios more than 60% of processor time is spent at idle, it is unlikely that this difference in processor utilisation by user space processes would significantly impact application performance.

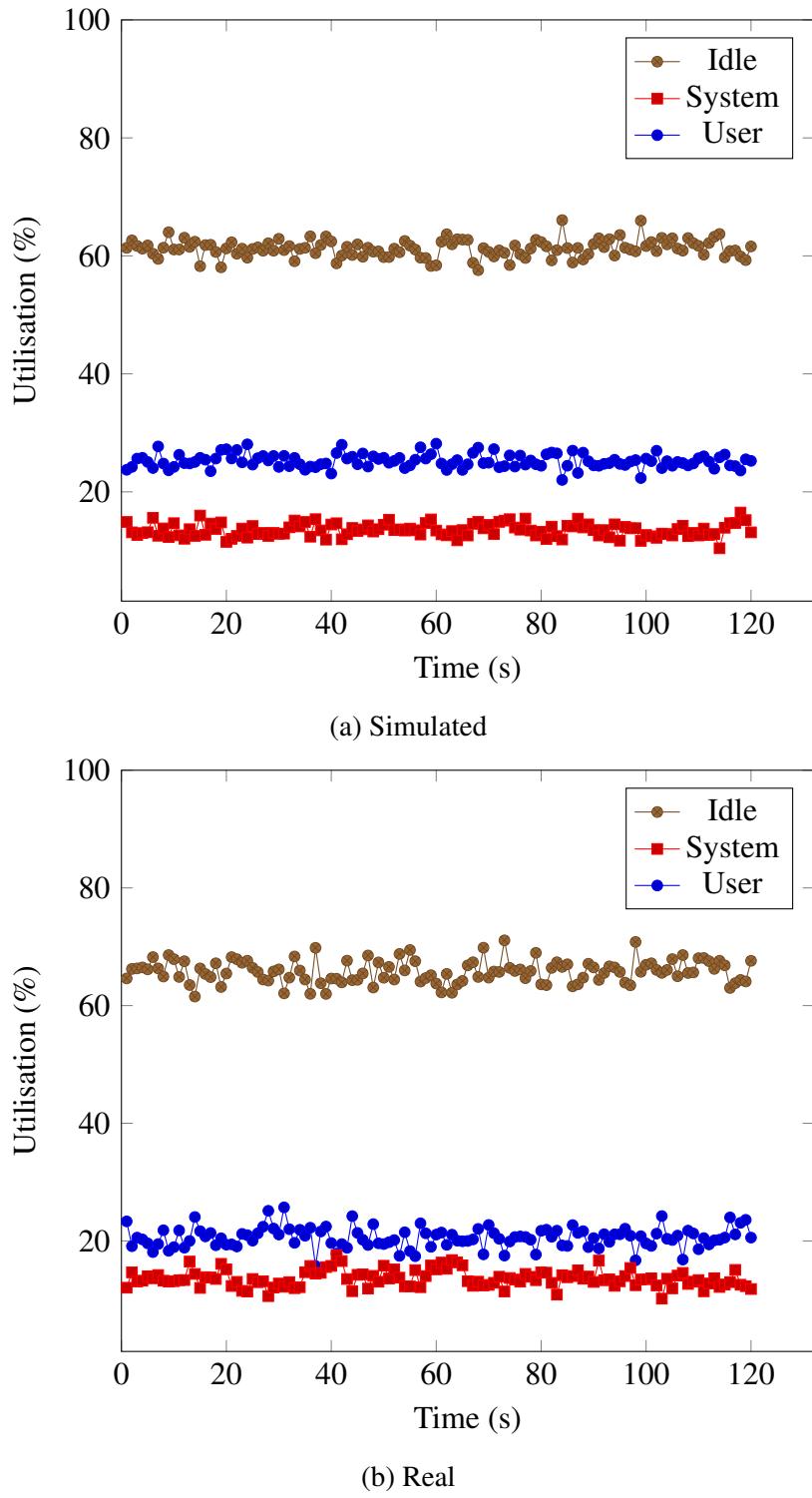


Fig. 8.16 Processor utilisation during LiDAR-based cone processing based on simulated (a) and real (b) input.

8.5.5 Response Time

Given that the simulation involves transferring streams of images and other sensor data across a network connection, verification is required to ensure that this does not introduce significant delays to the response time of the system. This was achieved by sending a control command from the simulation computer, measuring the time taken for the system to respond, and comparing this to the same measure taken on the FSAE vehicle. The results of this experiment are presented in Fig. 8.17, where it can be seen that the simulator had a significantly lower average response time. As testing currently occurs predominantly at low speeds this is seen to be insignificant for current use cases, although an artificial delay could be trivially added to the simulation system to better mimic the FSAE vehicle.

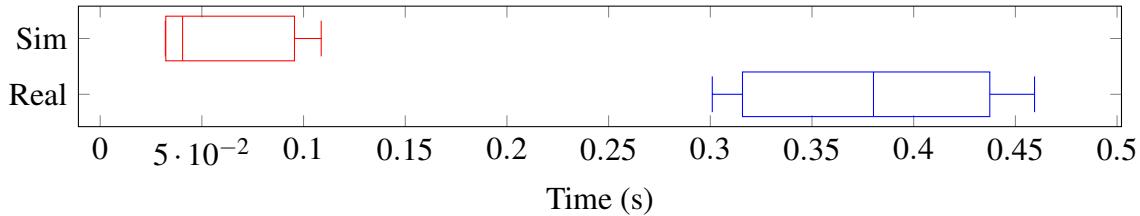


Fig. 8.17 Vehicle response times for real and simulated vehicles.

8.6 Future Work

In the current implementation, all vehicle dynamics are handled purely by CARLA through Unreal Engine’s vehicle physics. Some qualitative efforts have been made to match the stability of the FSAE vehicle during acceleration, braking and cornering actions through adjustments to existing vehicle parameters. This could be further reinforced by the completion of a quantitative comparison of the real and simulated vehicles’ poses during common driving scenarios. In the event that these results differ significantly, more accurate vehicle physics could then be implemented.

The response time of our simulation to a control command is significantly lower than that of the FSAE vehicle. In order to improve the accuracy of the simulator, it is proposed that a processing delay be added to control commands.

In order to reduce the synchronisation error, we plan to employ a local Network Time Protocol (NTP) server, which has a typical accuracy in the tens of microseconds over a local area network. Improving time synchronisation will also allow the software framework itself to be distributed over multiple compute nodes as the performance limitations of a single node are reached, and for the software framework as a whole to be synchronised with a reference clock provided by the available GPS receiver. While it is expected that this level of

synchronisation will be sufficient, time synchronisation could be further improved through the use of a Precision Time Protocol (PTP), which supports sub-microsecond synchronisation accuracy [344].

The HIL simulation system described has been used to simulate near ideal weather conditions only, in order to replicate the Western Australian climate, which averages approximately 110 clear and 130 partly cloudy days per year [345]. As such, a key area of focus for future enhancement of the system is the development of scenarios in less ideal weather conditions, and verification of the accuracy of the simulated sensors in these situations.

8.7 Conclusion

We have presented an HIL autonomous driving simulation system that is capable of simulating the various sensors available on the FSAE vehicle without hard real-time constraints. We have detailed the architecture of the software framework utilised on the FSAE vehicle, and how the use of this framework was able to simplify the development of the HIL simulation due to its high degree of flexibility and extensibility. Our design approach of utilising actively developed, open-source projects on commodity hardware results in a relatively low-cost simulation solution that is nonetheless capable of generating sensor data at a frame rate greater than that required for our FSAE vehicle's software framework. We have shown that this system allows for software to be tested in an environment that presents similar performance constraints as the FSAE vehicle platform. Most importantly, we have verified that results generated through use of this simulation system are transferable to the FSAE vehicle for the group's current use cases.

Chapter 9

REView: A Unified Telemetry Platform for Electric Vehicles and Charging Infrastructure

Charging station networks and connected vehicles play a pivotal role in the advent of smart cities and smart grids. A cornerstone of these infrastructures is often a platform or a service that handles the copious amounts of data generated, processing and storing them for monitoring and analyses. REView is a software platform to automatically collect, analyse, and review live and recorded data from electric vehicles (EVs) as well as electric vehicle supply equipment (EVSE or “charging stations”) to introduce a unified monitoring platform for these infrastructures that is both modular and scalable. The data described in this paper has been collected live from the Western Australian Electric Vehicle Trial and the WA Charging Station Trial. A secure web portal was designed with different viewing entries for electric vehicle users, charging station users and charging station operators. It includes informative statistics about a user’s driving efficiency and energy use and is compared to the average of all other users. It further includes a smartphone application for live monitoring and itemised billing. In this chapter, we discuss the development of REView, including mechanisms used to generate and collect the information. Finally, we show and discuss the visualised data itself, which includes the charging time, duration, energy used, as well as utilisation metrics of the charging infrastructure. We promote an open source approach to charging station software development. This will allow a single-software back end to handle multiple stations from different manufacturers, promoting competition and streamlining the integration of charging technologies into other devices. The results from this network and platform have ultimately enabled us to perform quantitative investigations towards the driving and charging behaviours, as well as the overall electric vehicle trends around Perth.

9.1 Introduction

The REV Project at The University of Western Australia (UWA) operates one of the largest electric vehicle (EV) charging station networks in Western Australia, where it is also the largest network operated by an academic institution in the country. This network includes 23 units of 7 kW AC chargers (Level 2) and one 50 kW DC fast charger (Level 3). All of the energy generated for on-campus charging is provided through a 20 kW-peak solar photovoltaic (PV) array. Additionally, our successful run of the Western Australian Electric Vehicle Trial 2011–2014 has enabled us to establish a vehicle fleet-tracking platform for EVs. Being able to collect individual user data for EVs and EVSEs and relay them back to the user in both itemised and statistical form, provides a substantial added value for individual mobility.

Our telemetry platform REView is a web-based software package that receives, processes and stores incoming telemetry data from connected infrastructures and more importantly, it facilitates data sharing from the various streams of incoming data, thereby enabling it to directly influence system automation without external intervention. This data is then visualised on REView’s front end where it then performs automatic statistical evaluations and presents the results to users in a meaningful and informative way, letting each user know about his or her own individual mobility costs, as well as their ranking alongside other EV users of this software platform. We find that by illuminating the user’s patterns and contrasting them with data from other users, we can motivate individuals to reduce their energy consumption and carbon emissions, as well as, in general, educate people about zero emission transportation. The approach of competing users against each other is known as gamification and has already been used in related areas [346].

The statistics generated allow drivers and fleet managers to monitor their vehicles’ efficiency and energy use. It allows charging network operators to monitor the effectiveness and usage of their stations, and it allows station users to monitor their energy usage and costs. This information is made available in several ways, including live mobile phone web applications, desktop web applications, data exporting and printing.

REView was developed as part of two different projects:

1. The Western Australian Electric Vehicle Trial, Australia’s first EV trial, consisting of eleven locally converted EVs based on the Ford Focus and owned by various businesses and government agencies [347].
2. The installation of the Western Australian charging station network, a set of 23 AC and one DC electric vehicle charging outlets, which are made available to the public.

REView has helped in analysing driving and charging behaviours of EV drivers [348–350] and statistics generated from this system have been used in setting up an acceptance study among EV drivers [351].

REView is currently providing real-time monitoring of vehicles, charging stations and solar installations around Western Australia, with new statistics generated every half hour. The software is presented as a full stack project to be entirely web based, utilising a combination of Python servers, cron batch scripting of Python for statistical processing, PHP server side and PostgreSQL back end with JavaScript, CSS and HTML for the user interface. As these open source languages (along with several open source libraries) make up the system, it has the ability to be used extensively and be available to any educational or not-for-profit organisation. We hope this will promote further research into charging station infrastructure and show that it is possible to fill the void between a research organisation's need for data collection and the government/corporate sponsors needed for investment return.

That said, the motivation for proposing REView mainly stems from the lack of availability of an open and scalable telemetry and data monitoring platform that comprehensively incorporates multiple sources of data across various infrastructures. Throughout the development of this software there were several lessons learned from stakeholders' requirements, acceptance of features by station users, and general possibilities for charging station software. We believe that the features that were developed in this system help form a baseline for future software developments in vehicle tracking and charging station monitoring. As many of these infrastructures become more integrated, telemetry platforms are required to become more centralised as an effective yet holistic solution to convey its information to the user. However, we have found that many existing solutions are exclusive, only supporting certain and often proprietary products and are usually impossible to integrate with other infrastructures. Through the EV Trial we found that commercial software which is sold bundled with charging stations often lacks vital functions and can be very awkward to operate, also in all cases we have seen, such software is limited to the associated company's charging hardware, and does not support interoperability with other stations. With many different charging station manufacturers in the market, this makes management, analysis and billing cumbersome.

The solution proposed by REView is to store and organise telemetry data. The significance of its contribution is the ability to handle, process, store and visualise this data effectively, which can occur on demand and in real time. The platform is scalable whereby it is able to process data across different manufacturers and infrastructures. Monetisation options are also included in REView, as monthly bills are automatically generated for users, station operators and administrators to quantify the running cost and usage of both the charging stations

and the vehicles. In addition to monitoring and visualisations, we also use this collected data to analyse charging behaviours and the local EV adoption rate. The combination of these features aims to present a centralised solution to inform and encourage EV market penetration.

We organise the remainder of this chapter as follows. Section 9.2 introduces the background into REView’s development. Section 9.3 presents the overview of the system design. Section 9.4 describes the telemetry design for charging infrastructures. Section 9.5 describes the telemetry design for vehicle fleets. Section 9.6 outlines the data management from energy generation. Section 9.7 features usage billing from telemetry data. Section 9.8 introduces REView’s mobile application. Section 9.9 evaluates data interpretations and their results. Finally, Section 9.10 draws the concluding remarks.

9.2 Background

In this section, we present the background research that resulted in the development of REView. These topics cover EV adoption, environmental impacts and telematic platforms, which are elaborated in the following subsections.

9.2.1 Local and International Adoption of Electric Vehicles and Charging Stations

When starting the WA EV Trial and the charging station trial in 2010, there were no OEM-built (commercially built) EVs available in Australia and no there were no charging stations in Western Australia. Since then, several manufacturers have released new electric vehicles and plug-in hybrid vehicles into the Australian market, including Mitsubishi, Nissan, Hyundai, Holden, Renault, Jaguar, BMW, Porsche and Tesla Motors. Globally the number of Electric Vehicles has grown from 700,000 in 2014 to over three million in 2017 [352]. Electric vehicles are no longer a pipe dream, but a reality, and every year we will see more on our roads. For fleet managers, tracking and logging of energy usage, as well as localisation and utilisation are valuable tools to reduce carbon emissions and expenses.

Meanwhile the number of charging station manufacturers has increased significantly, and many governments around the world are subsidising their installation in order to meet the desire to reduce the dependence on oil. While charging station manufacturers currently have incompatible customer identification methods and competing management software, the Open Charge Point Protocol (OCPP) [353] has been introduced as a possible new standard for EVSE communication. OCPP is an open, uniform communications protocol that can be used

across all charging stations. Already, many manufacturers are supporting this protocol—now being the most popular protocol for new stations. This means that external companies can access the data and control of the stations via an application program interface (API), no matter which manufacturer.

The number of EVs in Perth, Western Australia, has grown from 15 in 2010 to over 700 in 2018 with exponential growth. Various consulting firms and governments have forecast the growth of EV sales in Australia. In 2009 the Department of Environment and Climate Change commissioned consulting firm AECOM to study the economic viability of electric vehicles [354] and they projected that supply constraints would limit the sales of EVs (including hybrids) in Australia until 2020 and in each of their three projections over 60% of new cars would be plug-in hybrids or pure EVs by 2040. AECOM released another report in 2011 for the Victorian Department of Transport where through the use of a vehicle choice model, they concluded that sales of mild hybrid vehicles in Victoria will be more predominant in the short term (up to five years), plug in EVs in the medium term (five to 10 years) and pure battery EVs in the long term (more than 15 years) [355]. They also found evidence that high levels of charging infrastructure will significantly increase the adoption of EVs.

In 2012, ABMARC performed a survey of motorists in Australia with a conservative estimate of EV uptake. They concluded that without a breakthrough in battery technology the adoption of EVs by 2020 would likely be 0.4% of new car sales [356]. However, plug-in hybrid electric vehicles would constitute a much large proportion of 6.4% of the new vehicle market. The Energy Supply Association of Australia reviewed several different forecasts for Australia, showing that they all had several factors in common that controlled EV uptake, with a major factor being available EV charging infrastructure [357].

9.2.2 Importance of Measuring Environmental Impact

The environmental impact of running any vehicle needs to be analysed from its source. The environmental benefit in terms of CO₂ emissions of EVs relies quite heavily on the way the electricity is generated. In 2018, 21% of all energy generated in Australia came from renewable sources. This is made up as 35.2% from hydro, 33.5% from wind, 19.6% from small-scale solar PV, 7.1% from bioenergy, 3.9% from large-scale solar PV, and 0.8% from medium-scale solar PV [358]. The Union of Concerned Scientists released a report in 2012 in the US that related the source of electricity generation directly to the environmental benefit of a Nissan Leaf EV [359]. Their report showed that in some regions the difference in carbon emissions in electricity generation varied as much as three times, where a nuclear and renewable energy mix of generation is compared to a coal and gas driven power generation.

This meant that EVs in areas with high electricity emissions were comparable to highly efficient petrol vehicles (17% of all Americans live in these areas).

Table 9.1 Efficiency and Theoretical Emissions of Electric Vehicles

Model	Efficiency (Wh/km)	Range (km)	CO ₂ (g/km)
Hyundai Ioniq EV	115	280	104
Hyundai Kona EV	131	557	118
Renault ZOE	133	403	120
BMW i3	137	335	123
Tesla Model S (70)	185	455	166
Tesla Model X (60)	208	363	187
Jaguar I-Pace	230	446	207

Model, kWh/km and range from greenvehicleguide.gov.au [360], CO₂ emission calculated from SMEC 2008.

A report by SMEC in 2008 for the Department of Transport states that in Western Australia the amount of kgCO₂ Emissions per kWh is 0.936 [361]. From this information and the efficiencies of the models from the Australian Department of Infrastructure and Regional Development [362] one could calculate theoretical CO₂ emissions per km of the EVs available in WA, and Table 9.1 lists this according to the EVs that are currently available for purchase in Australia. However, this would assume that these cars are charged entirely from the average grid without any renewables, which is clearly not the case. Many early EV adopters also have solar PV generation at home and are able to charge their cars completely emission free. Also, just assuming the local energy mix for EV charging is too simplistic, as it does not take into account the time of charging:

- Public charging stations are mostly used during sunshine hours with a usage pattern quite similar to solar PV. This means EV charging can make use of excess solar energy during the middle of the day
- Home charging can be shifted to convenient hours of the night after the evening peak. At these times EV charging can make use of excess wind energy and use the otherwise wasted baseload energy of coal-fired power stations, which run through the night.

Further, the focus on CO₂ values ignores more harmful emissions, such as carbon monoxide and particulate matter (PM10 and PM2.5), which can be better controlled in power stations than in combustion engine cars.

In 2013, the Australian National Transport Commission released a report discussing the carbon dioxide emissions of new Australian vehicles [363]. They found that the average gCO₂ per km was 199g/km, meaning that EVs in the worst-case scenario generate less emissions than the average new petrol car. To reduce or remove CO₂ emissions for EVs, they must be charged (or arguably offset) from a renewable energy resource.

It is important to note that air quality in metropolitan areas will improve through the use of EVs, even when charged from a “dirty grid”. EVs produce zero local emissions and power stations are typically located in less populated areas outside a city. Also, many emissions can be better dealt with at a power station that at thousands of ICE (internal combustion engine) cars.

9.2.3 Telemetry Platforms and Networks

While we note the scarcity of a cohesive solution to telemetry monitoring for public charging stations, vehicles and solar PV systems, exclusive solutions for these systems are well documented and established. Charging station networks have since relied on the expanding EV market for their own expansions in installation base and coverage area. Commercial global networks such as Chargepoint [364] and Tesla’s Superchargers [25] have tens of thousands of installations to track usage and billing. Additionally, governmental networks such as China’s GB-T aims to achieve more than 200,000 installations nationwide by the end of 2018 [365]. Locally in Australia, networks such as ChargeStar [366] operate a system where stations owners can participate as part of their network to bill customers. Billings from these networks are often outsourced to external companies such as Go Electric Stations [367].

Telemetry for connected vehicles became popular when automotive manufacturers installed CAN buses onto their products, thereby allowing third-party access to vehicle data. By connecting cellular-enabled GPS tracking devices to a fleet of vehicles, a connected fleet telematics system (FTS) can thus be established to monitor these vehicles. These systems are often controlled through an application service provider (ASP) such as OpenGTS [368], Traccar [369] and GPSGate [370].

There are several examples of commercial GPS tracking software packages for fleet vehicles. Commercial vehicle tracking is used in many different industries including mining, trades, utilities, transportation, and government agencies. There are a number of products available in Australia, such as EZY2C [371], Fleetmatics [372] and Linxio [373]. These systems claim to provide solutions that will reduce fuel costs, improve productivity, reduce labor costs, and increase accurate reporting. These services install GPS tracking devices into the fleet vehicles to monitor them remotely. The major drawback of such commercial systems is that they do not record energy usage or the status of charging, air conditioning,

heating and headlights but rather exclusively rely on a GPS unit. Additionally, they don't include information from other devices, such as charging infrastructure. All these systems are aimed at the petrol fleet market.

Similarly, PV monitoring systems are also readily available. Many solar PV inverters such as SolarEdge [374] have Wi-Fi connectivity, which allows for remote monitoring on PCs and smartphones. Grid-connected PV systems can utilise the smart grid to manage solar feeds and power allocations. Locally, Synergy [375] offers such a system, which enables customers to monitor their PV system and solar feeds, as well as managing solar tariff returns.

In the academic scene, works that incorporate telemetry monitoring for connected vehicles [376–378], charging stations [379–381] and solar PVs [382, 383] are not uncommon. Many institutions around the world have started research into tracking and monitoring of electric vehicles and charging stations, using their own GPS systems and charging infrastructure. In the North East of England, Blythe performed a study tracking 15 electric vehicles and a charging station network [384]. They concluded with stating their ability to use the data from tracked vehicles to derive the state of the charging station network. From there they are able to predict possible future problem areas for electricity power generation.

While individual applications for charging network management and connected fleet or vehicles do exist, these are typically closed sourced and proprietary. Although we may not refute the availability of proprietary or undisclosed software, at the time of writing, we are unable to find any work relevant to a telemetry monitoring system that combines vehicle tracking and charging station usage.

9.3 System Design Overview

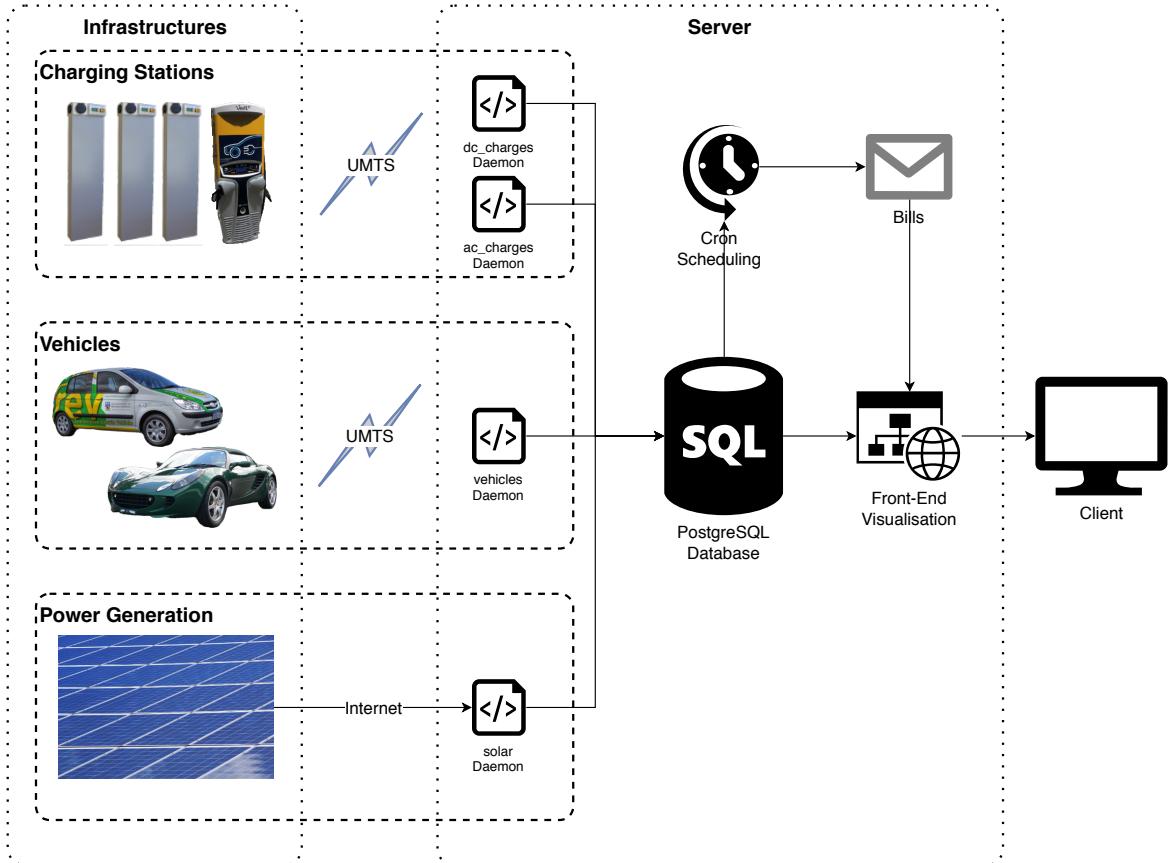


Fig. 9.1 Network architecture of REView.

REView's overall architecture can be summarised as Fig. 9.1, consisting of seven components — an EV server, charging station server, a solar downloader, data processing scripts, a database, a web server and its web interface.

The charging stations and vehicles are all fitted with machine-to-machine (M2M) modems, which are perpetually connected to Telstra's 3G Universal Mobile Telecommunications System (UMTS) network on the 850 MHz frequency. These modems then connect to our local server to transmit data over the Transmission Control Protocol (TCP) across two dedicated ports — one for the charging stations and another for the vehicles, as the charging stations and vehicles package data through different protocols. The server handles these transmissions through a SocketServer framework on Python 2.6.

The Python daemons that run the SocketServer function receive and parse the incoming telemetry data from the charging stations and vehicles before appending these data onto a PostgreSQL database. This process also verifies data integrity and consistency by filtering any

duplicates and non-events. This PostgreSQL database is installed onto the same server, which stores all data relating to telemetry, charging stations, vehicles and users. The platform's Apache front end can then access this data for visualisation and analysis.

The REView website features several pages including vehicle tracking, vehicle statistics, charging station status, charging station statistics, billing, heat maps, journey lists, charging lists, mobile tracking, and more. Depending on the type of user (station operator, station user or EV tracker) some pages are restricted or hidden. The website is a secure HTML 5 site with live information, interactive maps, graphs and customisable time scales. The supported browsers are Chrome, Microsoft Edge, Firefox and Safari, allowing access from computers, tablets and smartphones. At the time of writing, we are running PostgreSQL 8.4.20 and Apache 2.2.15 on a Red Hat Enterprise Linux (REHL) 6.10 server platform.

9.4 Charging Infrastructures

Our charging station network was established in 2011 as part of our research initiatives into the Western Australian EV landscape, which enables the collection of charging data to quantify charging trends and behaviours. This began with the installation of the Level 2 AC stations in the Perth metro area, followed by the installation of the 50 kW DC fast-charging station at UWA in 2014. We offer these charging stations free of charge to EV owners in return for research and data collection. The AC and DC stations transmit data through different protocols but are monitored for the same information. In other words, we collect information pertaining to charge times, duration and energy consumption from the stations for data visualisation and modelling. The following subsections detail the functions of these charging stations.

9.4.1 DC Charging

The 50 kW DC fast-charging station at UWA is a Tritium Veefil-RT [385] (see Fig. 9.2), which supports charging over the CHAdeMO [386] and SAE Combined Charging System (CCS) Type 2 (IEC 62196-3) [387] standards. This station performs telemetry through the Open Charge Point Protocol (OCPP) 1.6 over a 3G UMTS network. Data from the station is first pushed onto Tritium's server before it is pushed back to our local server, this allows Tritium to collect and consolidate data from its charging stations, and to streamline their maintenance and support on the station. The simplified class diagram in Fig. 9.3 summarises REView's functions on the DC station.



Fig. 9.2 UWA's Tritium Veefil-RT DC fast charging station.

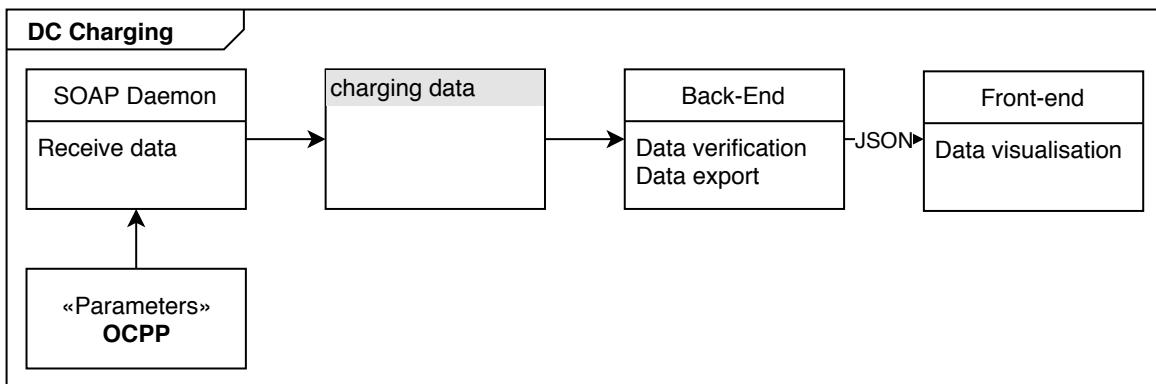


Fig. 9.3 Simplified class diagram of REView's DC station management.

9.4.1.1 Communication Protocols

OCPP data from Tritium's servers are transmitted through the Simple Object Access Protocol (SOAP). We run a PHP SOAP server with its service functionality described in a Web Service Description Language (WSDL) file, following the service list as described by e-laad.nl [388], to which we have ensured its compatibility with the Veefil charging stations. The charging station constantly transmits timestamped SOAP messages relating to heartbeats, start/stop charging events, status notifications, user authorisation and energy meter values.

By referencing the WSDL file, our SOAP server receives and logs incoming data, subsequently appending it into the dccharge table. Each charging station user is assigned a unique

identification (ID) the charging vehicle, and each station broadcasts a unique station ID that corresponds to its installed location; for each charging event, the charging station measures energy consumption in Wh units, along with its start and end times; finally, the station also broadcasts the charging standard used (CHAdemo or CCS2) for each charging event.

From the series of SOAP messages, the server appends charge data following Algorithm 6. The combination of this data allows us to perform time series analysis and modelling with regards to charge frequencies, duration, energy consumption and charge standard used. Additionally, we have instigated measures to maximise user participation through the station's proximity to the city centre, while providing the service free of charge. These measures provide us with reliable data to model EV trends and fast-charging behaviours around Perth.

Algorithm 6 DC charges database append

```

1: procedure DCAPPEND(incoming header)
2:   open Connection to database and table dccharge
3:   read incoming header
4:   if header is StartTransaction then
5:     append to dccharge with values TransactionID, UserID, StationID,
      StartTimestamp, StartUnits, ConnectorID
6:   end if
7:   if header is StopTransaction then
8:     update dccharge with values UserID, StopTimestamp, StopUnits,
      TransactionData where same TransactionID
9:     update status = ChargeComplete
10:   end if
11:   close Connection to database
12: end procedure

```

9.4.1.2 User Authentication

To improve data integrity, our DC station supports user authentication based on near-field communication (NFC). Users are given the flexibility to use any compatible NFC card in their possession, including those for public transportation, work or school. Fig. 9.4 shows a non-exhaustive example of NFC cards that the station accepts. The charging station reads these cards for a four-byte unique identification number (UID), which is presented as eight hexadecimal numbers. The UID is then crosschecked with its database entry for charging authentication. In addition to the tag number, users are required to register with UWA for

their names, email, vehicle model and registration number; whereby these fields are stored in a separate table in the database.



Fig. 9.4 RFID cards supported by the DC station. Pictured are a Transperth Smart-Rider card, a UWA Student ID and a general issue MIFARE RFID card, commonly used by station users.

9.4.1.3 Data Visualisation

Data visualisation for the DC stations is performed through a series of SQL queries which are parsed through a series of JSON encode/decode processes over PHP. REView visualises these data in the form of time series graphs and pie charts, as well as a page delimited table with the option for a *.csv file export.

A back-end PHP script performs data parsing by connecting to the PostgreSQL database, which we use to send SQL queries. For the visualisation of charts, these arrays are appended with their headers, and then combined with chart parameters such as graph types, chart title, axes titles and legends; which are subsequently encoded into a JSON representation. For a table visualisation, the JSON representation will also include the total number of charging events to enable the page delimiter to determine the number of pages for the table.

Likewise, a front-end PHP script *gets* the JSON representations from the back-end script and decodes it. This script interfaces with the web browser and is therefore also programmed with HTML and JavaScript as a webpage. Users of this webpage can specify visualisation periods with a start and end date, which is then used as part of the SQL statement to generate the query. The front-end script then parses the decoded JSON representation to determine all chart parameters, and subsequently plots them in a table using the Google Visualisation API [389] as illustrated in Fig. 9.5.

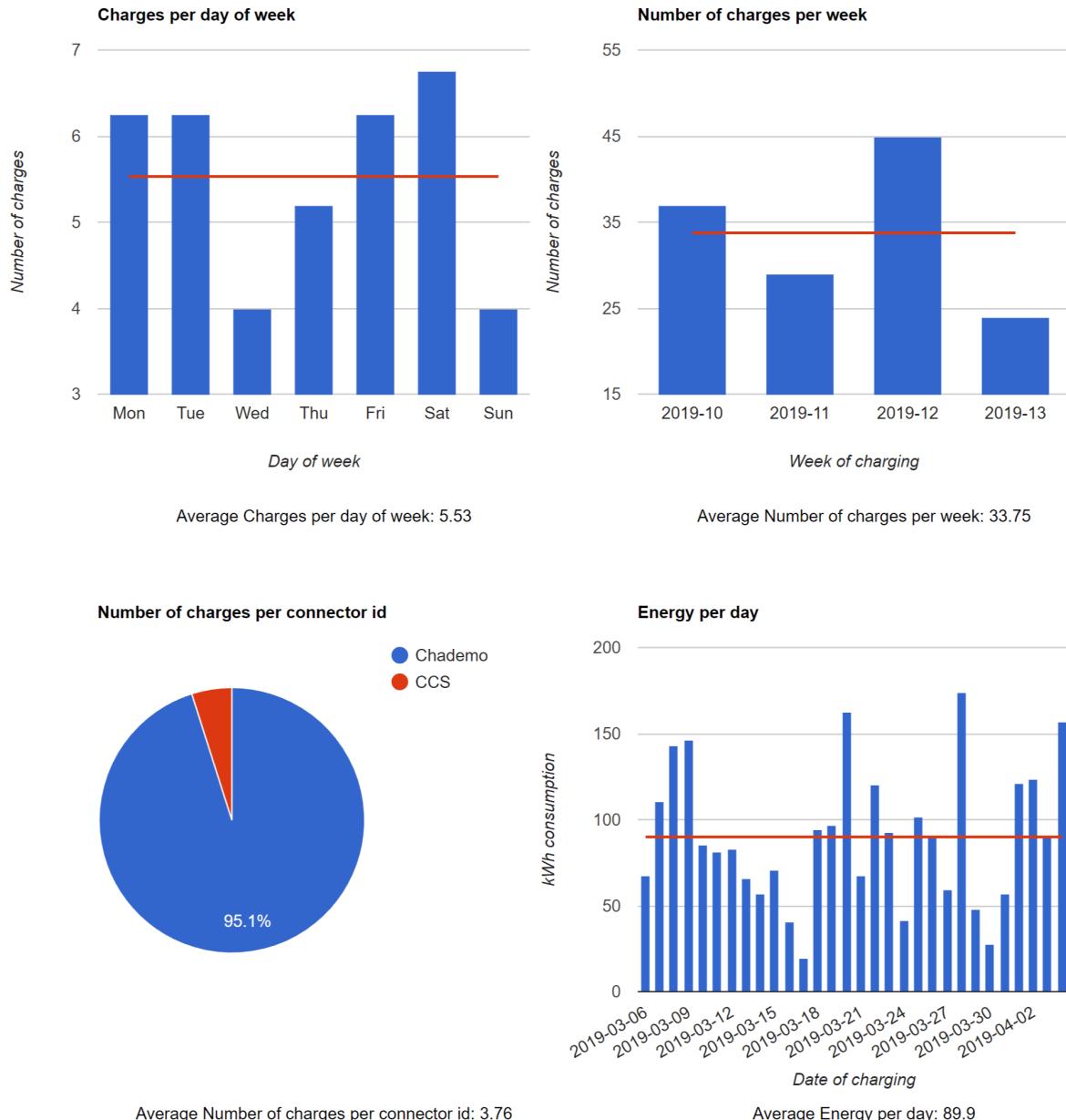


Fig. 9.5 Examples of visualisations for the DC station given the month ending 5 April 2019.

On the other hand, the table visualisation front end as illustrated in Fig. 9.6 draws an HTML table and fills it with the array obtained from a query that returns data from all DC charging events with the columns storing start and end times, charge duration, energy consumption and type of connector used. Each page is limited to 50 entries as defined in the query. This page supports *.csv data exports, whereby the file will be downloaded with the same table headers.

Count	ID	Start Time	End Time	Duration	Energy (kWh)	Connector
1	13974	2019-04-05 12:34:04	2019-04-05 12:55:04	00:21:00	13.978	CHAdeMO
2	13973	2019-04-05 12:08:01	2019-04-05 12:34:00	00:25:59	16.715	CHAdeMO
3	13972	2019-04-05 02:59:00	2019-04-05 04:02:02	01:03:02	38.824	CHAdeMO
4	13971	2019-04-04 22:01:02	2019-04-04 22:27:04	00:26:02	4.069	CHAdeMO
5	13970	2019-04-04 16:59:01	2019-04-04 17:04:05	00:05:04	1.051	CHAdeMO
6	13969	2019-04-04 15:32:04	2019-04-04 15:58:00	00:25:56	17.593	CHAdeMO
7	13968	2019-04-04 14:36:05	2019-04-04 15:30:01	00:53:56	35.025	CHAdeMO
8	13967	2019-04-04 13:13:00	2019-04-04 14:10:05	00:57:05	41.922	CHAdeMO
9	13966	2019-04-04 12:01:03	2019-04-04 13:09:03	01:08:00	49.971	CHAdeMO
10	13965	2019-04-04 09:29:02	2019-04-04 09:41:02	00:12:00	7.606	CHAdeMO

Fig. 9.6 Screen grab showing the DC charges table on REView with the last ten charges.

9.4.2 AC Charging

Our AC charging station network consist of 11 dual outlet (see Fig. 9.7) and one single outlet Elektromotive Elektrobay charging stations, totalling 23 stations. These charging stations act as individual clients and connect to a central server node in a many-to-one configuration. Each charging station is powered through a 7 kW three-phase AC supply and are either wall mounted or floor mounted. The charging outlets support the IEC 62196-2 (Type 2/Mennekes) standard, which is compatible with all recent EVs sold in Australia. The stations are water resistant and fitted with overcurrent protection and RCD switches. The functions of REView on our network are as summarised in the simplified class diagram in Fig. 9.8, and are elaborated in the respective subsections.



Fig. 9.7 UWA's dual outlet Elektromotive Elektrobay AC charging station.

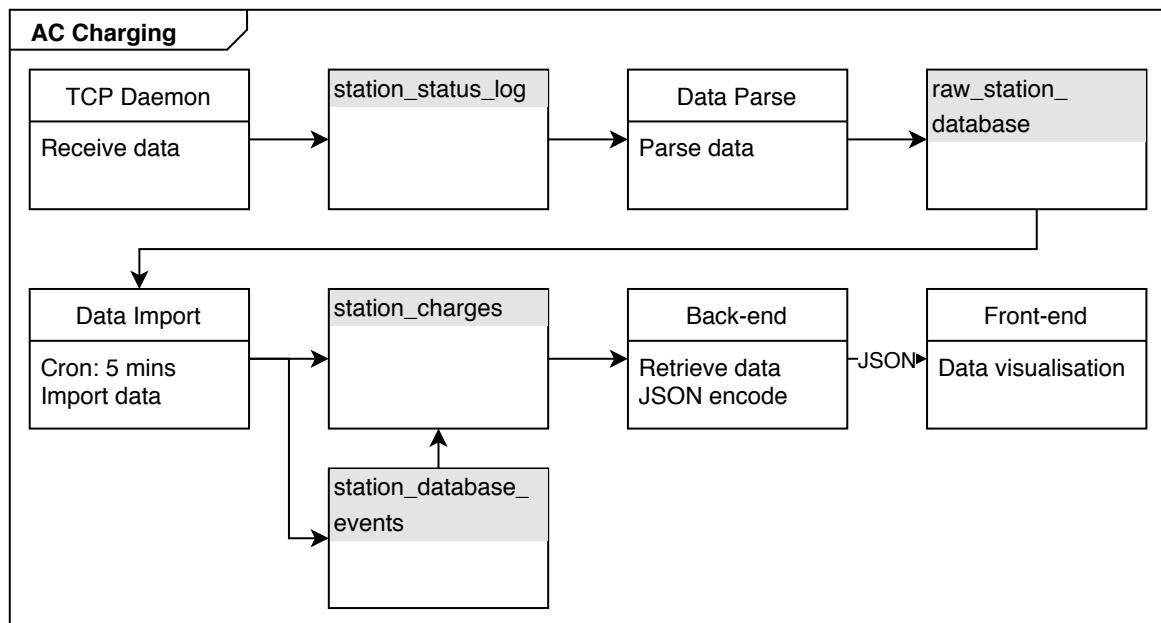


Fig. 9.8 Simplified class diagram showing REView's AC station management.

9.4.2.1 Communication Protocols

The AC station network achieves telemetry through the RS-232 serial standard, wherein for each station, a DE-9 cable connects to a Four-Faith F2414 [390] High-Speed Downlink Packet Access (HSDPA) M2M modem (see Fig. 9.9), transmitting charging data directly to our local server. We configured these modems using AT Commands, which instructs them to connect to our server through its IP address and port number, using TCP for data transmission. These modems establish a perpetual cellular connection upon power on, enabling them to transmit data from the stations on demand. Unlike the DC station, each AC station can be configured directly using the Elektromotive EB Connect software to set station parameters such as authorisation control, energy metering, charge limits, charge event and energy tracking. It also gives the administrator the ability to remotely login to the station or disconnect a user or reset a station. To allow this software to connect to the station, the server can open a Secure Shell (SSH) tunnel between the station and the administrator PC. This can be used to either remotely configure the station's modem or the charging station itself.



Fig. 9.9 A Four-Faith F2414 M2M modem used in AC charging stations.

Telemetry data from the AC charging stations follow Elektromotive's proprietary protocol that transmits a series of concatenated, timestamped hexadecimal strings. In order to append the database, we run a Python-based data parsing script as a daemon that first splits the string into parts that correspond to the database variables, and then converts these parts into intelligible ASCII texts. This script logs all incoming telemetry data from the charging stations, which for every charging station, transmit at five-minute intervals. It also checks and sets the stations real time clock, and requests information from the stations internal database.

The charging station modems are configured to connect to the server, allowing them to use dynamically allocated IP addresses, which are generally a cheaper option to using more convenient static IPs. To distinguish the charging stations, each modem is programmed to transmit its Modem ID as a header for all outgoing TCP messages. The Modem ID is a four-byte alphanumerical variable that is appended into every database entry to uniquely identify the station and its charging outlet (left/right side). In other words, a dual outlet station will consist of two modems, one for each charging outlet. The server then parses all messages and groups them according to their charging stations based on this Modem ID.

Each station keeps a record of several types of events, including charging, disconnect, power failure and reset. When the number of recorded events at the server is less than that at the station, the excess records are downloaded and stored for later statistical analysis.

9.4.2.2 Telemetry Parameters

In addition to the Modem ID, other relevant parameters are further detailed in their respective paragraphs.

Clock Each AC charging station keeps a clock which timestamps telemetry messages as they are produced. The stations' clock differs from the server's clock whereby the server's clock is used to timestamp the telemetry messages as they are received. This clock redundancy is particularly useful at times when the UMTS network is unreliable, in which case the charging station will store these messages in memory before transmitting them in series as soon as the UMTS network re-establishes. While the charging stations' clocks support over-the-air synchronisation, this process occurs intermittently on demand, which implies that the server's clock, being consistently connected to the Internet, is more accurate. To accommodate this issue, the data parsing script checks for any time discrepancy between the server and the station, setting the message timestamp to the server's time if it is less than 12 hours apart.

Station Attributes A *flag* variable transmits station attributes through an array of 16 binary objects that correspond to various station statuses. These array elements are, in sequence, “data erased”, “station restarted”, “no return timer activated”, “no power drain”, “excess power drain”, “power failure”, “door jammed”, “door forced open”, “record trip”, “mains status”, “door status”, “charge time exceeded”, “transaction start”, “transaction end”, “remote event”, and “force end charge cycle”. The values presented in this array are important for data collection and station diagnostics. For example, we use a combination of these station attribute flags to decipher station logs and append them as charging events into the database.

Station Status The combination of short telemetry intervals with its contained station attributes enables REView to determine the status of each station, where they are classified as “In use”, “Not in use” or “Unknown”, and subsequently visualised on the front end for users to check on the station bays’ occupancy. While the “in use” and “not in use” states can be ascertained from the telemetry logs, an “unknown” state is set for a station when more than 15 minutes have elapsed since its last telemetry message was received, considering that each station is programmed to transmit messages every five minutes. A station that is “in use” will also display its current charging time and allowing registered users to track their charging status on their phones via a mobile website.

9.4.2.3 User Authentication

AC charging station users have been supplied with RFID tags for identification to allow monitoring and future billing. This also reduces the risk of cable theft, as only the correct tag can release the charging cable. Other identification methods used elsewhere include smartphone login, credit card swipe and in-vehicle identification, but these require higher security standards (in case of credit card readers) and constant Internet connection, which makes these methods more expensive.



Fig. 9.10 An RFID token used for our AC charging station network. Pictured here is the tag for the User ID REV182.

User authentication on our AC charging station network is established through Elektromotive-supplied RFID tokens (see Fig. 9.10). Each charging station is fitted with an RFID tag reader that conforms to the MIFARE DESFire [391] standard. These RFID tokens also communicate through the 13.56 MHz band and are also compatible with the DC charging station. Users intending to charge their vehicle at our charging station network are required to register their details and consent to having their charging data collected for research purposes. We issue the Electromotive RFID tokens to all registered users, and each RFID tag number uniquely identifies the user through the storing of the tag’s user ID.

9.4.2.4 Database

The database stores charging data from the charging station network in a normalised manner across three redundant tables shown in Figure 1.8, namely `raw_station_database`, `station_database_events` and `station_charges`. This redundancy not only protects the data from accidental deletions, it also enables us to experiment and modify filter and merging rules for the tables while preserving data integrity.

Noting the longer average charge duration of Level 2 stations (the location of AC stations sees most users leaving their vehicle charging while at work), we have set up REView to feature per-hour and per-day-of-week data recordings for charging duration and energy consumption. For instance, a size 24 array represents each hour of the day; a size 7 array represents each day of the week. By segregating these analyses into a time series, we can therefore visualise and model the stations' utilisation as time progresses. The longer charge duration will often also imply that vehicles are left plugged into the charging stations even after they are fully charged. These vehicles are therefore subjected to a trickle charging state where their batteries are charged at their self-discharging rate, where the charging stations are **maintaining** charge at an **idle** state. To differentiate and analyse this charge maintaining state, we classify all charging instances that draw less than 1 kWh per hour as one that maintains charge, and that of more than 1 kWh per hour as one that is actively charging.

We present Algorithms 7 and 8 for the importation of data between tables, each representing a different parsing script.

Algorithm 7 station_status_log to raw_station_database

```
1: procedure ACAPPEND1(incoming telemetry message)
2:   open Connection to database and table station_status_log
3:   read incoming NumRecords, ModemID
4:   fetch lastEntry from station_status_log
5:   while incoming.NumRecords < lastEntry.NumRecords do
6:     amount = incoming.NumRecords – lastEntry.NumRecords
7:     fetch past amount record from station_status_log
8:     for each amount record do
9:       append into raw_station_database with values StationID, Flags,
   ServerTime, StationTime, MeterReading, EnergyReading, Index, UserID,
   ModemID
10:    end for
11:   end while
12:   close Connection to table station_status_log
13: end procedure
```

Algorithm 8 raw_station_database to station_charges

```

1: procedure ACAPPEND2(incoming header)
2:   open Connection to database and table station_database_events
3:   for each ModemID in station_database_events do
4:     lastEvent = max(endtime) from station_database_events at ModemID
5:     get first event
6:     for each event where StartTime < lastEvent do
7:       if not stationRestarted and transactionStart and serverTime < currentTime
8:         then
9:           if not chargeStart then
10:             chargeStart = event
11:           end if
12:           lastEvent = event
13:         end if
14:         if chargesStart and (transactionStart and transactionEnd) or
15:           (chargeRestart and not transactionStart) then
16:             kWh = EndMeterReading - StartMeterReading
17:             append into station_database_events with values stationID,
18:               StartTime, EndTime, kWh, UserID, rightSide, ModemID
19:           end if
20:         end for
21:       open Connection to table station_charges
22:       for each appended event do
23:         append into station_charges
24:       end for
25:     end for
26:   close Connection to all tables and database
27: end procedure

```

We improve data integrity by setting charging limitations on the charging stations. These include a 12-hour charge duration limit, and an energy threshold between 3 Wh and 14.4 kWh. The charge duration limit ensures that users who forget to tag off after a charge, or subject their vehicles to extended periods of trickle charge are not accounted for data collection; similarly, current limits protect the charging stations from overloading and the charging event is properly terminated as soon as the vehicle's battery is fully charged. On

the database, we have programmed the data import script to filter for extremes and negative values. These filters ensure that

1. Energy values are between 0 kWh to 200 kWh. This upper limit is, at the time of writing, higher than the battery capacity of any commercially available EV.
2. Charging durations are between 0 to 12 hours, to ensure that the charge duration limits are coherent with those set at the charging stations.

Any entry that does not satisfy the filter boundaries are not imported into the `station_charges` table.

9.4.2.5 Data Visualisation

Unlike the DC charging station's architecture, data from the AC charging station network is stored entirely on the same local server and does not involve any communication between servers. Having all data eventually consolidated into a single table also enables us to condense, query and retrieve all data that is required for visualisation into a single SQL query that combines data from charging events with its tag owner. The result is encoded as a JSON representation that is subsequently sent to the front end that calls it.

Data visualisation at the front end is similar to the DC charging station's whereby it generates a series of charts and a page delimited table. However, the added complexity of a charging station network, along with the increased variety of collectable data from it implies a differentiation from our descriptions in Section 9.4.1.3. The Google Visualisation API is again used to render charts for the AC stations. The difference, however, is that data parsing is performed directly on a single SQL query result, as compared to sending multiple queries for the DC station. A JavaScript function is written to parse the decoded JSON representation and group them into multiple charts based on their headers. Users can select a specific time period for visualisation periods, as well as specific charging stations. In addition to time series charts describing energy usage and charge duration, having per-station and per-user data also enables cost patterns and energy usage to be visualised for each user and station in the form of pie charts. An example of these visualisations is given as Fig. 9.11.

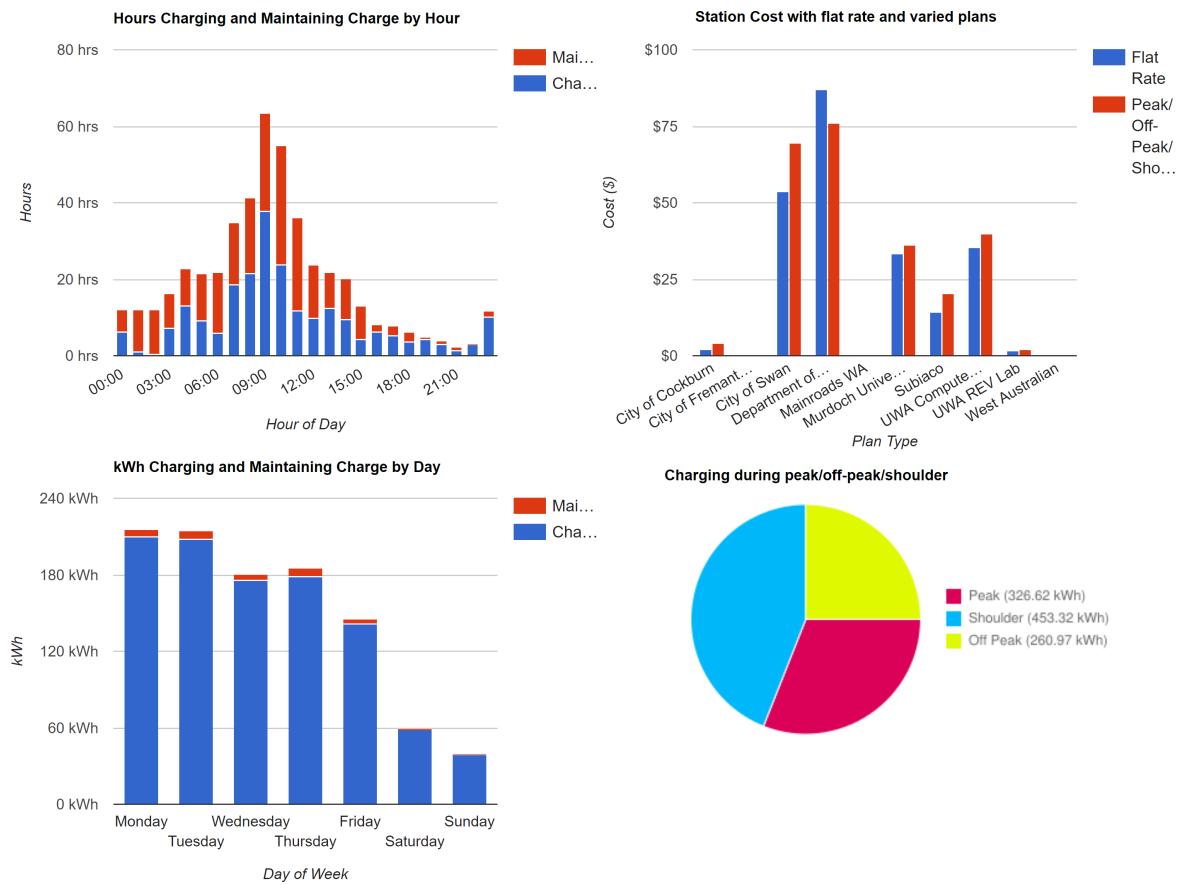


Fig. 9.11 Visualisation examples for the AC stations given the month ending 5 April 2019.

Furthermore, using this query approach along with a visualisation period selection enables the front end to include a summary table for each charging station and its network. This table summarises, for any selected period, its total energy consumption, estimated running cost (peak and off peak), number of charges, total plugged-in time, total charging and maintaining charge time, average charge duration, average charging and maintaining charge duration, percentage of time in use and power used for peak, shoulder and off-peak times. Each of the statistics generated is useful to charging station operators. The table shows:

- Which electricity plan is more useful, displaying the cost of a flat rate of electricity price (e.g. 21.87¢ per kWh) and a tiered rate (peak/shoulder/off-peak, e.g. 42.15¢, 21.44¢ and 11.32¢ per kWh respectively), which charges more during peak times and less during off-peak times.
- Time spent charging the vehicle (drawing more than 1 kW) and the time spent plugged in and not charging as a percentage. This shows if a station is more used for charging or if a location is more used as a parking spot.

- Time spent on a transaction (how long the vehicle is plugged in on average).
- Amount of time the station is actually in use versus its total time installed. Showing how often the stations are utilised as an average over all locations.

9.5 Vehicle Monitoring

REView's electric vehicle tracking functionality was first established and presented as part of The REV Project's WA Electric Vehicle trial in 2011, whereby a fleet of 11 electric-converted Ford Focuses were fitted with mobile GPS tracking units, which tracks the vehicles' movement patterns, battery level, charging, headlight, heater, air conditioning and ignition status. 2.7 million data points were collected over the trial period, which includes 5,000 independent journeys averaging 9.3 km per trip and 1,600 charges. Since then, our vehicle tracking platform has expanded to include our electric Jet Ski, electric boat and autonomous driving projects. The incorporating of EV tracking onto REView enables us to better analyse usage behaviours of EVs and to model future commute habits. Results stemming from our vehicle tracking system are published in [350]. Vehicle tracking on REView is performed according to the class diagram in Fig. 9.12, and is detailed in the subsections.

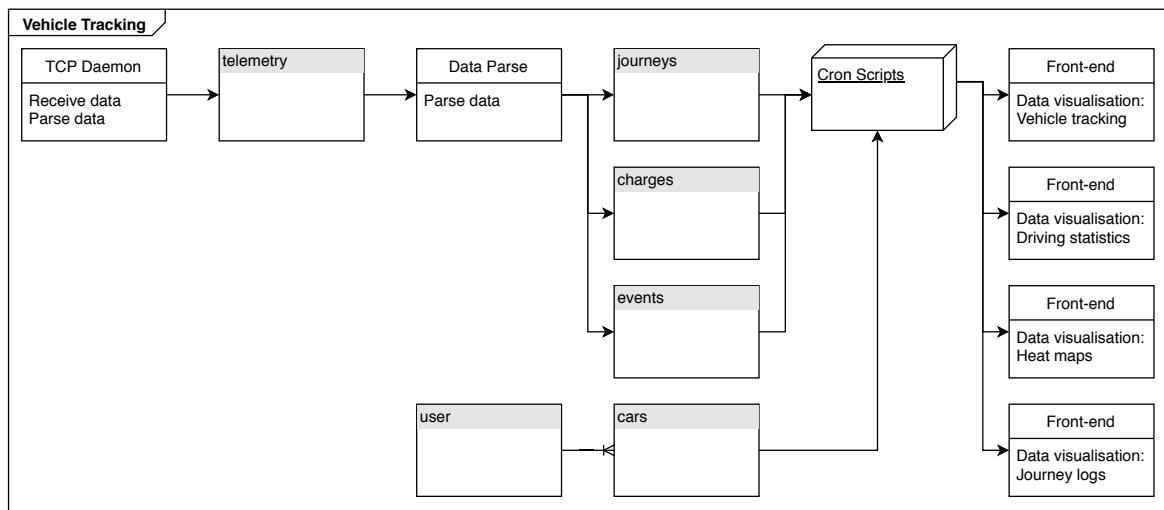


Fig. 9.12 Simplified class diagram showing REView's vehicle tracking system.

9.5.1 Communication Protocols

Our fleet network was initially fitted with Astra Telematics' AT110 [392] tracking devices, which were then upgraded to the AT240 (version 8.5) [393] to support the 3G UMTS network.

These tracking devices communicate with the server via an internal UMTS modem, using a SIM card with machine-to-machine (M2M) capabilities.



Fig. 9.13 An Astra Telematics AT240 vehicle tracking device used in our vehicle fleet network.

The AT240 (shown in Fig. 9.13) are capable of vehicle tracking through a Global Navigation Satellite System (GNSS) over the ublox EVA-M8M module [394], and are IP67 waterproof rated, making them suitable for our watercraft projects. Each tracking device is based on a Cortex M3 microcontroller [395], which is powered by a 510 mAh battery that is able to sustain three days of hourly updates, and a three-axis accelerometer to detect motion and driving behaviours. Input/output options include six digital inputs and five digital outputs, as well as two inputs connected to an analogue-to-digital converter.

The five digital input lines of the tracking devices are connected to the air conditioning, ignition, headlights, radio and, heater statuses. The analogue input line is connected to a battery level logging device that outputs the battery level percentage as an analogue voltage. The battery meter counts the energy flowing in and out of the main electric vehicle battery pack using a current sensor. Serial communications are facilitated through the RS232 standard. In the case of our road vehicles, these connect through OBD-II over a CAN bus, which transmits the vehicle's journey information to the tracking device. This journey information supplements the sensor readings from the device which are then packaged and transmitted over to our server.

The charging system monitoring is done with a Python daemon running the Threading Socket server library. This library listens for TCP connections on a defined port and creates a new thread for each one, which can handle the processing of the data received. The processing is done by parsing the incoming message from the byte stream into Python variables, connecting to the database and inserting the new data points. Each vehicle connects to our local server via a dedicated port over TCP, where the vehicles are identified through the devices' IMEI number. The tracking devices transmit telemetry messages to the server

using Astra's proprietary Protocol A. This script connects to the same database and deciphers each incoming packet from the vehicles.

The tracking devices' reporting frequency is determined by the vehicle's ignition state. In other words, a vehicle that is running (positive ignition) will transmit data every minute; whereas a vehicle that is parked (negative ignition) will transmit data every half hour. Charging is facilitated through the vehicle's 12 V rail, which is connected in line over a 1 A fuse to the vehicle's battery.

9.5.2 Database

Our server's vehicle tracking daemon receives and parses all incoming telemetry data from the vehicles, which is subsequently appended into the `telemetry` table in the PostgreSQL database. The following information is recorded in the database for each data point — latitude, longitude, time logged on device, time received at server, vehicle speed, vehicle heading, altitude, journey max speed, journey max acceleration, journey distance, journey idle time, ignition status, alarm line status (unused), air-conditioning status, headlights status, heater status, charging status, and car battery level. The GPS positions and line inputs are uploaded onto the server either at every minute or at every ten meters, whichever comes first.

The server runs a separate parsing script, scheduled as a cron job, to further process and classify the variables into three separate and redundant tables.

1. The `journeys` table records data pertaining to journeys made with the condition that the device records an increase in distance travelled. Entries are appended at one-minute intervals.
2. The `charges` table is appended from any telemetry data that whenever the vehicle is charging, are appended at one-minute intervals.
3. The `events` table logs and timestamps any appends pertaining to charges and journeys, summarising and batching each data parsing event according to the vehicle's IMEI number.

This is followed by ten cron-scheduled Python scripts that calculate and update the imported entries for these tables, activating every 30 minutes. These scripts perform the following:

- Generate vehicle journeys, charging events, idle events, missing data events.
- Generate charging station events.
- Combine similar charging station and vehicle charging events based on user tag, time and location.

- Compress data, such as air conditioning, heater, headlights into a data point for a journey.
- Compress charging data into charging/maintaining charge, divide into by-hour and by-week arrays.
- Generate heat maps.

As separate scripts are used for different functions, adding new functionality or statistics can be easily done by adding an additional script. This limits the need for modifying existing software and reduces integration problems and helps in isolating errors.

The following paragraphs detail notable information that the data import relates:

Charging places By assuming that each charging event will take place at either the trial participant's workplace, home or a public charging station, each participant nominates a work and home location for charging. Charging places are hence classified as charging stations, workplaces and homes with their data stored in a separate table in the database.

Data loss As each tracking device is programmed to transmit data at every minute when the vehicle is running, and every 30 minutes when the vehicle is parked or idle, any subsequent data packets that arrive later than 30 minutes will be identified as a data loss, these are recorded into a `data_loss` table.

Time series modelling Data from the telemetry table are parsed into day/week time series representation through scripts 8 and 10. This is done by selecting and grouping telemetry data from day and week segments. Travelling distances and energy consumption is summed as they are appended into the `charges` and `journeys` tables. Time series modelling is vital for the visualisation of our vehicles' data as they are disclosed to trial participants to gauge and understand their driving habits.

9.5.3 Data Visualisation

REView visualises vehicle data across four categories — vehicle tracking, driving statistics, heat maps and journey logs, as detailed in their subsections. All visualisations are presented on REView's front end, which is presented through a PHP script that connects to any of the three related tables on the database through SQL queries. Users wanting to view their vehicle's data on REView are required to log in with their credentials, which is linked to their registered vehicles. In this context, we establish a one-to-many relationship between

the User and Cars tables in the database. We use a one-month sampling period to represent the figures in this section.

9.5.3.1 Vehicle Tracking

The vehicle tracking page looks up a list of vehicles that corresponds to the user viewing it. The user can select an individual vehicle or all vehicles in a fleet and the time period to display. The graph is interactive, allowing the user to drag and zoom in on the time scale.

This page uses PHP scripts to supply the information to JavaScript code on the page using several free-to-use libraries including JQuery [396] for communication with the server, Google Maps for the map and Dygraph [397] for the interactive graph. The Dygraph library is open source and was modified for use in the website.

To display the GPS data, the map has the ability to show individual interactive points that can be clicked on for additional information or an image that is generated by a PHP script on the server. The number of interactive points is limited to 150, as too many points can cause instability in the browser. However, the image overlaid over the map can contain any number of points, which allows users to see data over longer time periods. Generating an image at the server is also useful, because the information sent from the server to the user is significantly less. The server caches all images generated and generates differently scaled images for different map zoom levels.

For performance and stability reasons, the graph below the map is limited to displaying 2,000 points at a time. To reduce the load on the server, the graph caches data and only requests additional information when necessary. When a user pans the graph to the left or right, only the missing information is requested. The granularity of the data is also important, and the server is designed to send sub-divided data when the time period selected has more than the 2,000-point maximum. When the user zooms on a section of the graph, the server is asked for sub-divided or raw information for this smaller time period.

A query reads all IMEIs associated to that user and retrieves a list of related vehicles from the Cars table, which is subsequently presented as a selectable radio button list. For each selected vehicle, the vehicle tracking page visualises the vehicle's position and its trajectory from the selected time period, which is displayed using the Google Maps JavaScript API. The vehicle's current location is given as a colour-coded pin that corresponds to the selected vehicle, whereas its trajectory is plotted through an array of markers obtained from the vehicle's historical location data from the telemetry table.

Additionally, the vehicle tracking page also includes a chart that allows users to check, by defining a sampling period, the vehicle's driving behaviour as a time series, which is also exportable as a *.csv file. REView plots multiple line graphs on this chart, which includes

battery levels, vehicle speed, travelled distance, as well as air conditioning, headlights and heater usage. Fig. 9.14 shows an example of this visualisation.

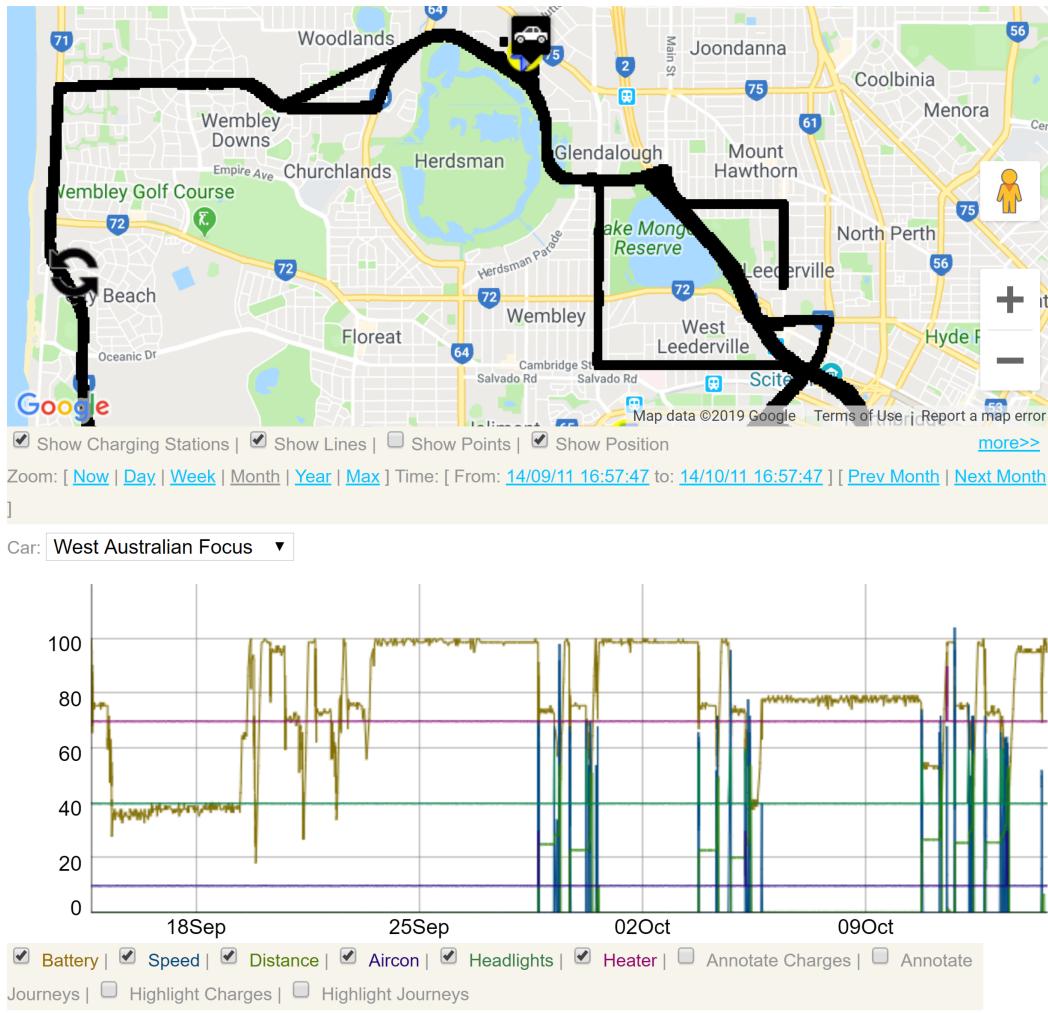


Fig. 9.14 Partial screen grab of the vehicle tracking page showing its historical trajectories on Google Maps, and its driving behaviours in the bottom chart.

9.5.3.2 Driving Statistics

Visualisation for driving stations is presented in two modes — one as an overview for all vehicles, and another for a detailed analysis of driving patterns. The overview page consolidates statistics across all vehicles, and for a specific range of dates, with examples as shown in Fig. 9.15.

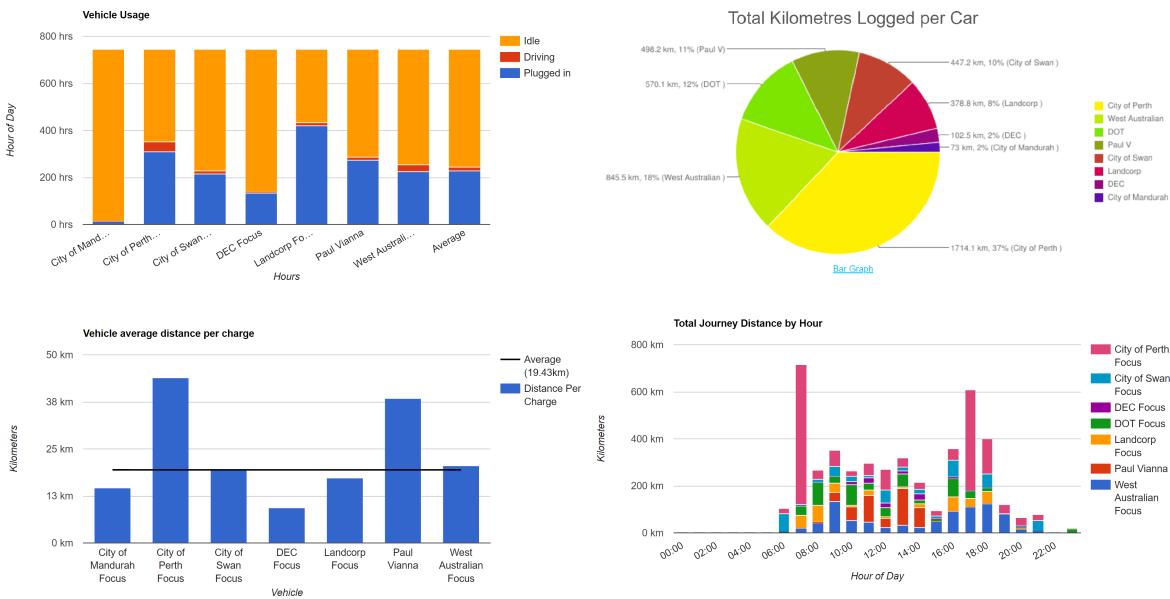


Fig. 9.15 Examples of charts showing driving data.

Leaderboard The leaderboard table ranks the vehicles according to their distance travelled and compares them alongside the other vehicles. The table also lists the vehicles' driving times, total number of journeys, average journey distance and average journey time. An example of the leaderboard is shown in Fig. 9.16.

Leaderboard Table

	Rank	Vehicle	Distance Travelled	Time driving	Total Journeys	Average Journey Distance	Average Journey Time
	1st	[Pixelated Vehicle Name]	1714.1 km	1 day 19:10:22	110	15.58 km	00:23:32
	2nd	[Pixelated Vehicle Name]	845.5 km	1 day 01:52:12	142	5.95 km	00:10:55
	3rd	[Pixelated Vehicle Name]	570.1 km	16:57:48	51	11.18 km	00:19:57
	4th	[Pixelated Vehicle Name]	498.2 km	14:17:07	65	7.66 km	00:13:11
	5th	[Pixelated Vehicle Name]	447.2 km	14:09:58	83	5.39 km	00:10:14
	6th	[Pixelated Vehicle Name]	378.8 km	12:22:07	39	9.71 km	00:19:01
	7th	[Pixelated Vehicle Name]	102.5 km	03:10:48	12	8.54 km	00:15:54
	8th	[Pixelated Vehicle Name]	73 km	01:24:39	4	18.25 km	00:21:09

Fig. 9.16 A sample leaderboard displaying driving statistics for the trial vehicles for an average month. Vehicle names have been pixelated to preserve privacy.

Chart legends Legends used in the charts are described according to their:

- Vehicle name: Name of vehicle e.g. UWA Lotus, UWA Getz etc.
- Location type: Type of charging station location e.g. home, business or public station.
- Vehicle state: Vehicle status e.g. idle, driving or plugged in.
- Plug type: Charging current e.g. 32 A, 15 A or 10 A.
- Daily distance travelled in km, with an average plot.

Likewise, driving statistics for the individual vehicles are presented across three sections — distance, charging and journeys, as shown in Fig. 9.17. For each statistic, results from each vehicle are compared against the average results from the rest of the vehicles (community). This, along with the leaderboard, is a gamification feature, which we find useful to keep monthly reports interesting for the users. These are mostly presented in bar charts of grouped pairs to illustrate the statistical difference between that vehicle and the community.



Fig. 9.17 Individual driving statistics are presented in distance, journeys and charging sections.

9.5.3.3 Heat Maps

REView generates heat maps of the tracked vehicles through the aggregation of their GPS coordinates and location type geofencing. They are automatically generated and can show

areas where vehicles drive, park and charge within certain time periods. The server periodically runs a heat map generating Python script as a cron job, which searches through the journeys, charges, telemetry and any low charges from the journeys table, eventually generating a *.kml file that contains the location of the heat maps.

The Python script imports Jeffrey Guy's heatmap library [398], which utilises the Python Imaging Library (PIL) to generate a heat map *.kml file along with a *.png file containing an image overlay that corresponds to the heat maps. Areas with increasing overlapping entries from the database will progress from a navy to a red hue, as shown in Fig. 9.18. These heat maps can be generated according to the user's selection, which includes the type of heat map (places parked, charged or driven), time frame (year or month) and time period.

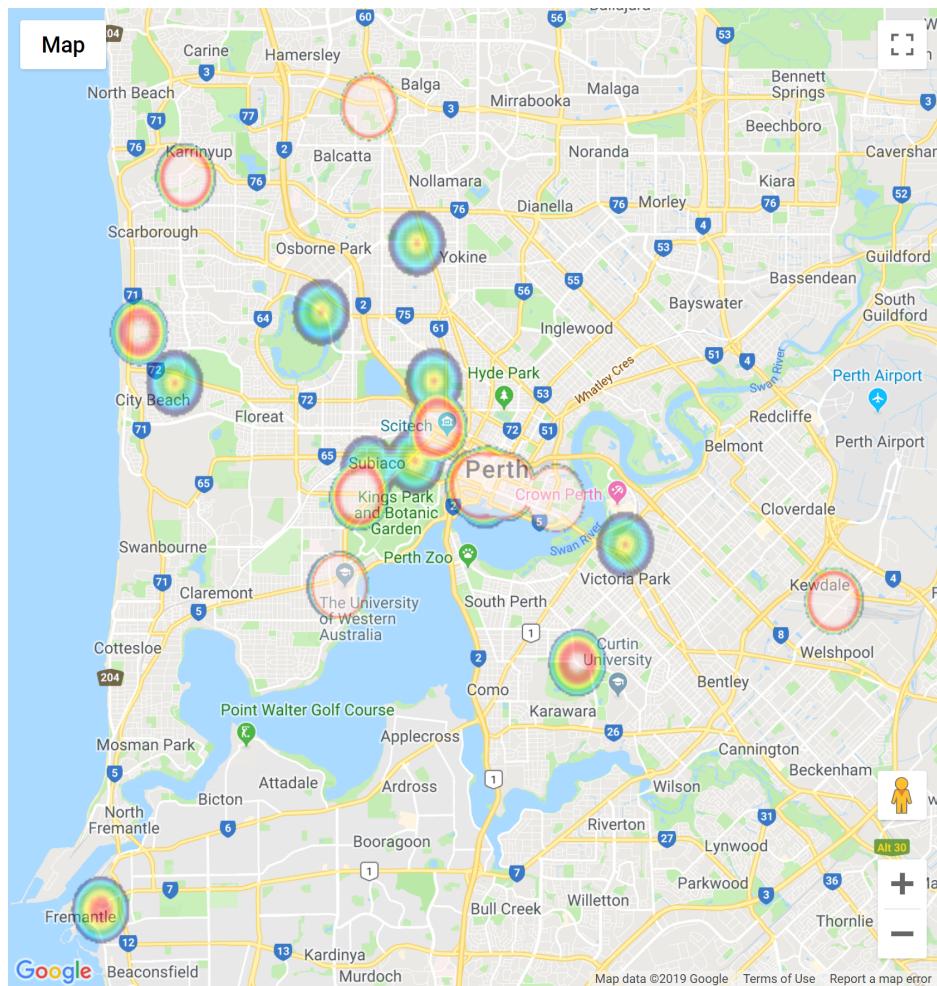


Fig. 9.18 Heat map of the vehicle fleet tracked over a month for possible charging locations that are utilised during the day (7 am to 6 pm).

9.5.3.4 Journey Logs

The completion of any drive routine from a tracked vehicle will result in the tabulation and display of its journey log onto REView as shown in Fig. 9.19.

The cost of equivalent petrol, electricity and carbon emissions are calculated based on the distance travelled, which is 10.43 cents, 3.47 cents and 168 g per kilometers respectively. These values were taken off the Carbon Dioxide Emissions from New Australian Vehicles 2013 information paper [363] by the Australian National Transport Commission along with current electricity tariffs as an average. The monetary savings are calculated as a difference between petrol and electricity cost. Importing options for *.csv data is also available.

Count	ID	Car	Distance (km)	Time	Battery Used (%)	Maximum Speed (km/h)	Idle Time (minutes)	Start Time	End Time	Emissions Saved (kg CO ₂)	Electricity Cost (\$)	Petrol Equivalent (\$)	Savings (\$)
4701	27505	[REDACTED]	39 km	00:36:22	0 %	112 km/h	3	2014-06-05 09:17:12	2014-06-05 09:53:34	6.591 kg	\$1.35	\$4.07	\$2.72
4702	27506	[REDACTED]	39 km	00:36:22	0 %	112 km/h	3	2014-06-05 09:17:12	2014-06-05 09:53:34	6.591 kg	\$1.35	\$4.07	\$2.72
4703	27503	[REDACTED]	14.9 km	00:22:43		80 km/h	6	2014-06-05 09:02:59	2014-06-05 09:25:42	2.5181 kg	\$0.52	\$1.55	\$1.03
4704	27504	[REDACTED]	14.9 km	00:22:43		80 km/h	6	2014-06-05 09:02:59	2014-06-05 09:25:42	2.5181 kg	\$0.52	\$1.55	\$1.03
4705	27501	[REDACTED]	13.3 km	00:20:29		80 km/h	2	2014-06-05 08:28:13	2014-06-05 08:48:42	2.2477 kg	\$0.46	\$1.39	\$0.93
4706	27502	[REDACTED]	13.3 km	00:20:29		80 km/h	2	2014-06-05 08:28:13	2014-06-05 08:48:42	2.2477 kg	\$0.46	\$1.39	\$0.93
4707	27500	[REDACTED]	0.2 km	00:12:25	0 %	30 km/h	11	2014-06-05 06:20:21	2014-06-05 06:32:46	0.0338 kg	\$0.01	\$0.02	\$0.01
4708	27499	[REDACTED]	0.2 km	00:12:25	0 %	30 km/h	11	2014-06-05 06:20:21	2014-06-05 06:32:46	0.0338 kg	\$0.01	\$0.02	\$0.01
4709	27498	[REDACTED]	0.1 km	00:01:40	0 %	20 km/h	0	2014-06-04 21:41:15	2014-06-04 21:42:55	0.0169 kg	\$0	\$0.01	\$0.01
4710	27497	[REDACTED]	0.1 km	00:01:40	0 %	20 km/h	0	2014-06-04 21:41:15	2014-06-04 21:42:55	0.0169 kg	\$0	\$0.01	\$0.01

Fig. 9.19 Table showing the journey logs of a tracked vehicle. Vehicle names have been pixelated to preserve privacy.

9.6 EV Charging Power Generation

UWA's AC chargers are powered through a solar PV array installed on the roof of the Human Movement building. This is a dual inverter PV plant that is rated at 20 kWp and collectively generates 114.1 kWh per day, which is greater than the combined usage of all our AC chargers. This plant uses two Sunny STP 10000TL-10 [399] 10 kW three-phase inverters with a maximum efficiency of 98.1% at a nominal 600 V. Monitoring on the solar plant is done through SMA's Sunny WebBox [400], which provides a web interface for configuration,

including that for telemetry data transmission to our local server at up to one-minute intervals. The average instantaneous measurement of the plant is tabulated in Table 9.2. REView processes data from this infrastructure according to the class diagram in Fig. 9.20.

Table 9.2 The average instantaneous measurements of the solar PV plant

DC Current	1 A
DC Voltage	510 V
DC Power	1000 W (2 inverters)
AC Grid Frequency	50 Hz
AC Grid Power	1600 W
AC Phase Current	1.1 A
AC Phase Voltage	225 V
AC Phase Power	500 W
AC Day Yield	56.67 kWh

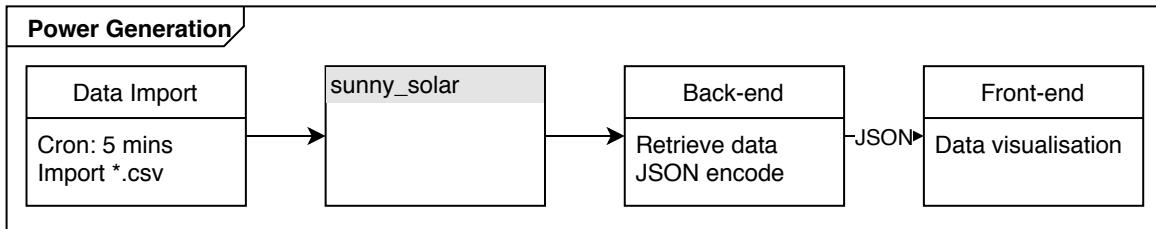


Fig. 9.20 Simplified class diagram showing REView's energy generation system.

9.6.1 Data Visualisation

Data from the Sunny WebBox is pushed daily to our server via FTP as a series of `*.csv` spreadsheets, containing timestamped entries at five-minute intervals that record each inverter's instantaneous measurements, similar to Table 9.2. We subsequently wrote a Python script that imports this data into our server's PostgreSQL database as the `sunny_solar` table, which runs on a five-minute cron schedule. This script reads the last event in the table and imports every successive data from the spreadsheet, synchronising the data between the two entities. The data from the solar system includes time stamps, power generated, voltages at the panels and grid and operation health flags. PV systems are connected to and scanned every 15 minutes for data download.

This data in the `sunny_solar` table is, in turn, visualised as a series of line graphs on REView's front end, which can be selected through a day, week or month period. Visualisations

are given in the solar plant's power and energy generation and consumption, as accumulated from the selected period. We achieve this visualisation by importing the JpGraph [401] library, whereby a back-end PHP script calls all related SQL queries which then encodes it as a JSON representation for the front end, as Fig. 9.21 illustrates.

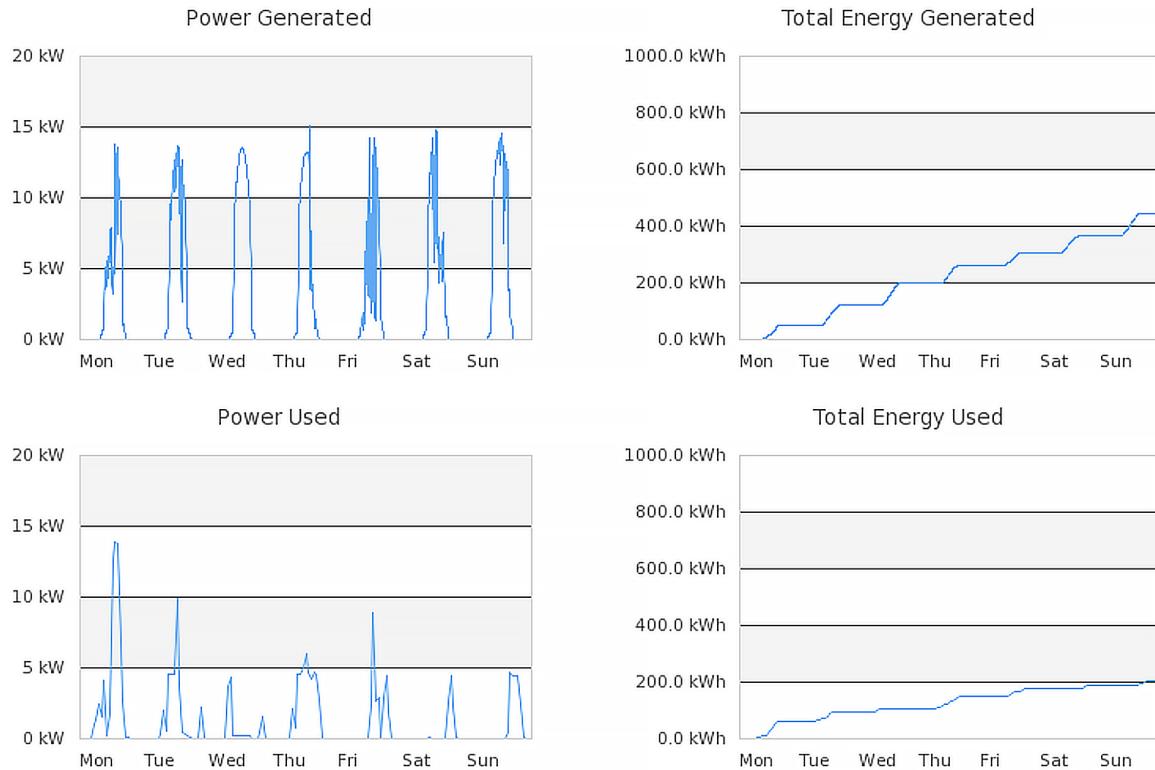


Fig. 9.21 Graphs generated from solar PV data over a typical week.

9.7 Usage Billing

The billing page allows an EV user to view his or her monthly mobility cost. It also lets station operators view utilisation and energy usage of their stations. In both cases summaries as well as itemised bills are generated. All bills are automatically generated with several informative graphs, including distance travelled, distance per charge and kWh per km of the individual versus the community. Also, a time-of-use energy graph is generated.

To provide users and charging station owners with a summarised update, bills are generated at every first Monday of the month through a scheduled cron job, which is then emailed to related users who opt into this service, where they are classified as either vehicle owners or station operators. For this purpose, we have configured a CSS script that enables the printing of the contents directly through HTML as a Tax Invoice. This is presented as a combination

of charts and tables that are dependent on the type of bill presented — Vehicle/User (User Billing), Charging Station (Station Operator Billing) or Summary (Network Overview). Each bill type is generated through individual PHP back ends that encode SQL queries as a JSON representation as detailed in the subsections.

9.7.1 User Billing

An itemised bill enables users to individually monitor and track their charging behaviours, which is displayed as a line graph followed by three tables (see Fig. 9.22). If the vehicle is part of our tracked fleet in Section 9.5, its driving statistics (total distance travelled, distance per charge and kWh per km) will be presented above the line graph and compared against the rest of the community. REView obtains these data by running a query for each graph or table for a given bill. This query is applied for the `station_charges` table according to the accessing user and its selected sampling timeframe, which returns the charging timestamps, duration, location and per-hour energy usage to the front end.

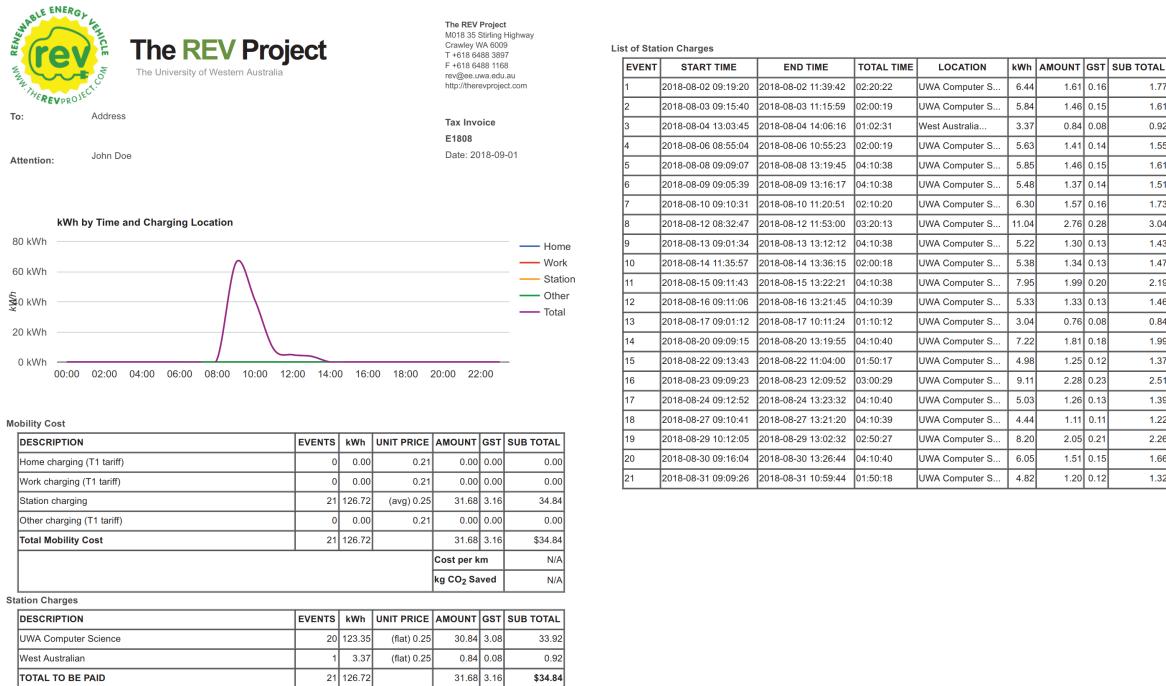


Fig. 9.22 Itemised bill generated for a station user for August 2018.

The line graph illustrates the cumulative charging energy consumption per hour for the vehicle according to the charging places from Section 9.5.2, which is individually labelled as different coloured lines. The data used is summed through the per-hour energy usage data obtained in the query as mentioned in the first paragraph. To achieve this, the system uses

an incremental function to individually increment the energy consumption for each location type at each hour of the day.

The mobility cost is the first table presented following the line graph, which tabulates the cost of charging based on the user's charging location. Headers are given as the charging locations, number of events per location, energy consumption in kWh, tariff units and totals with GST, which is calculated as 10% of the total price. We use a standard tariff of 25 cents per kWh for charging stations and 21 cents per kWh for other locations. When presented for a tracked fleet vehicle, it also calculates the cost per kilometres travelled and the amount of CO₂ saved.

The second table presents station charges using the same headers found in the first table, which itemises the charging cost according to the charging stations used, detailing the individual station charging events found there.

The third table further itemises the second one by listing each charging event of that vehicle on charging stations, listing each event using headers that correspond to its starting and ending timestamps, duration, station location, energy consumption, and amount totals with GST.

9.7.2 Station Operator Billing

Bills are also sent to station operators to summarise the monthly usage of their charging stations. Station operator bills consist of a community comparison, a line graph and an itemised table (see Fig. 9.23). Data is obtained from the `station_charges` table through a query that returns the charging user, starting and ending timestamps, energy consumption, per-hour charging and maintaining energy consumption, charging duration and connector side (left or right).

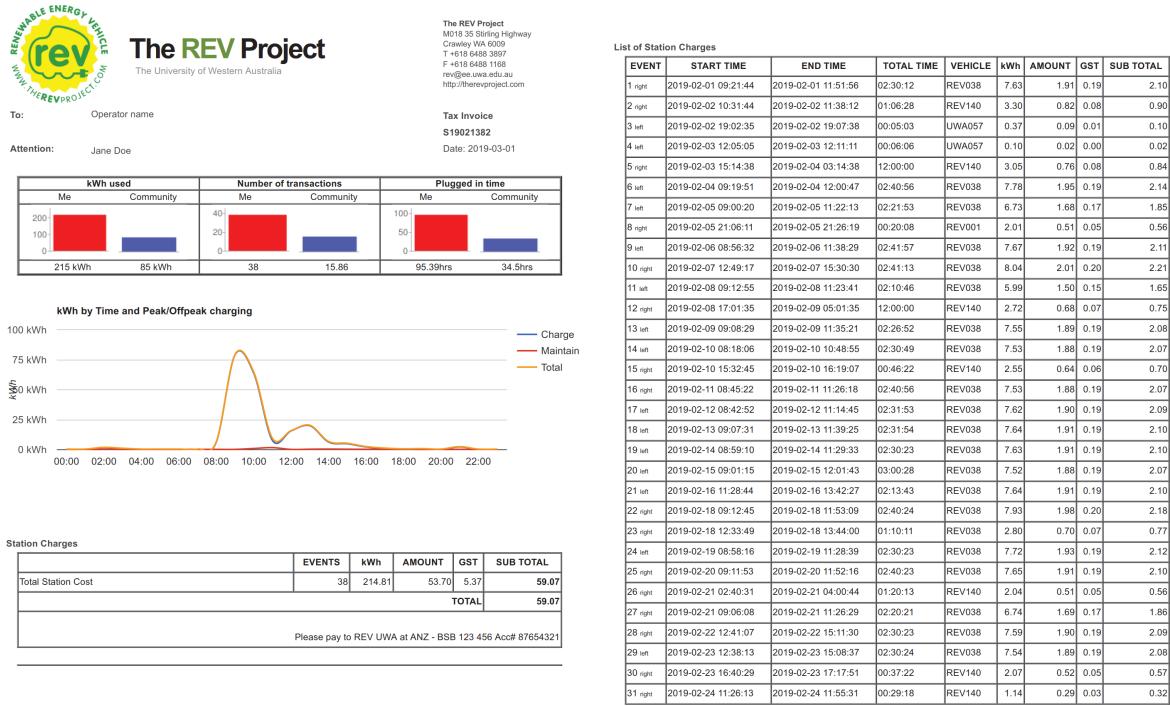


Fig. 9.23 Itemised bill generated for station operators for February 2019.

Community comparisons are drawn through the billing station against the average charging parameters of the other charging stations, where these parameters are given as their energy consumption in kWh, number of transactions and plugged in time. These are illustrated as a pair of colour bar charts for each parameter.

The line graph plots the per-hour energy consumption of the charging station on an average day, separated by its charging, maintaining and total plots. The total plot is the instantaneous sum of the “charge” and “maintain” energy consumption per hour.

Finally, the `station_charges` table begins by listing the total operating cost for the selected billing period, which lists the number of charging events and total energy consumption for that period, followed by the total energy costs on a 25 cents per kWh tariff. Another table chronologically itemises each charging event under the headers: connector side, starting and ending timestamps, charging duration, vehicle tag, energy consumption and charging price with GST.

9.7.3 Network Overview

REView displays an overview of the charging station network and vehicle’s usages, which is accessible only to project administrators. This bill is generated for a selected period through a drop-down menu that evokes a query that produces a usage summary for our entire AC

charging station network. Data on the network overview bill is presented as a line graph that is followed by two tables (see Fig. 9.24). Data for this bill is obtained through a query that returns individual events under the following headers: station name, vehicle tag, energy consumption and per-hour energy usage (charging and maintaining charge).

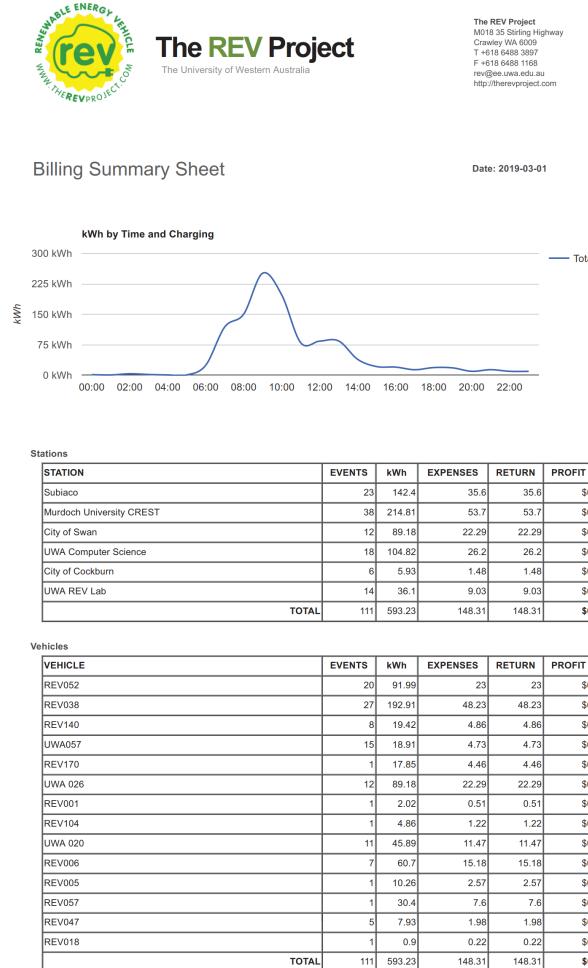


Fig. 9.24 Network overview bill generated for March 2019.

The line graph plots the cumulative per-hour energy usage across all stations for each hour of the day. This graph consolidates data under the per-hour energy usage header from the query, adds the energy usage for charging and maintaining charge across each hour of the day, and subsequently presents it as a sum of energy consumption for each event in the selected period.

Each of the two tables aggregate charging events, one for the charging stations, and the other for the vehicles. These tables tabulate, for each charging station or vehicle, its number of charging events and total energy consumption in kWh. Additionally, this bill also includes cost (expenses), revenue (return) and profit generated from each event. (At the time

of writing, our expenses and return tariffs are equally set as we are operating on a non-profit model). Data from the query is processed through a PHP back end that calculates values for an aggregated number of events and energy usage, as well as its associated costs. For each charging station and vehicle, an incremental function finds and increments the number of events and its associated energy usage, and subsequently its operating costs, revenues and profits, before encoding the entire table as a JSON representation.

9.8 Mobile Application

REView has two mobile phone applications for EV drivers and station users as illustrated in Fig. 9.25. The first allows users to view their vehicle status on their mobile phone, showing location, status and battery level. The second allows station users to see if their vehicle is still drawing power or if their EV is fully charged. It also allows users to check remotely if a station is occupied or free, allowing EV drivers to plan their trip ahead. These applications helped ease ‘range anxiety’ where drivers fear their vehicle will not have enough energy left in the battery to make it to a destination.



Fig. 9.25 Smartphone application running REView.

Designing mobile web pages instead of apps makes sure they can be used for every smartphone or tablet model. The pages were developed as lightweight web pages using HTML 5 and JavaScript, which communicate periodically with the server for data updates. Each of the charging stations are listed in a page, with their availability indicated by a blue or green icon.

9.9 Results

In this section, we will discuss results from the WA Electric Vehicle Trial, represented in REView graphs. We also present usage analyses and forecasts for the charging infrastructures.

9.9.1 Overall Energy Usage

Fig. 9.26 shows the energy consumed charging EVs by hour of day and location. This information can be used for analysing EV grid impacts and the usage of renewable energy. The locations are defined as followed:

- Home: A residential area.
- A commercial or industrial area.
- An EV charging station.
- An undefined area.

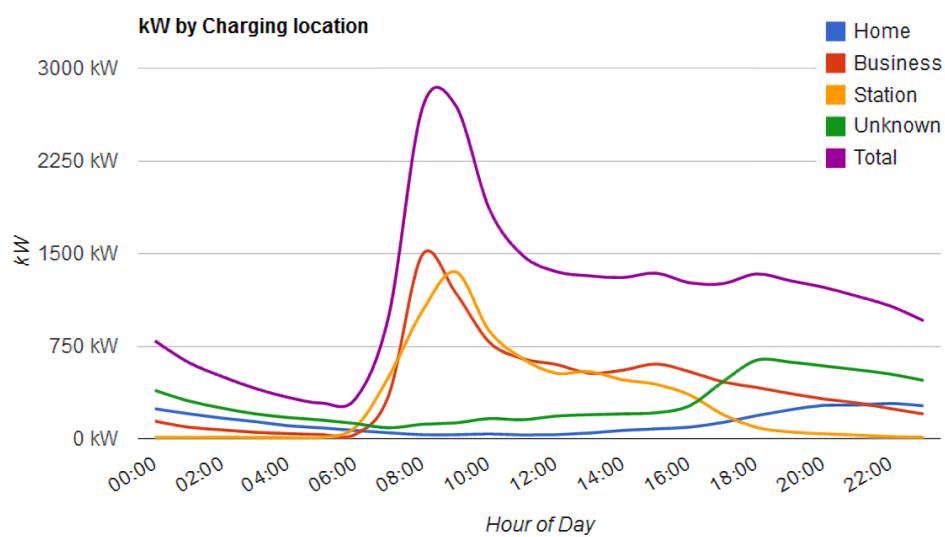


Fig. 9.26 Power drawn by hour of day for EV charging at various location types.

The peak of the energy supplied for charging vehicles (averaged over all locations) is during the morning hours around 9–10 am. This means, EVs are commuting from home to work and use a charging facility at work (most likely free of charge). It is worth noting that the majority of energy supplied is during sunshine hours especially for station charging and business charging. For unknown and home charging, there is a much smaller peak at around 6 pm, as vehicles are returning home and charging there. This also suggest that the majority of unknown locations are unlabelled home locations.

From this information and solar information gathered (see Fig. 9.28) we can show that typical charging scenarios can be offset almost ideally by solar technology. Most of the charging occurs during the day, which differs fundamentally from the scenario propagated by some energy suppliers, which shows all EVs charging around 6pm when they return to home.

9.9.2 Charging Infrastructure Usage

The statistics in this subsection are taken from REView’s Stations page showing the summary of all charging stations as a part of the WA Charging Station Network, from the beginning in June 2012 through March 2019.

In Figs. 9.27 and 9.28 we discuss the difference between charging and maintaining charge. It is common that an EV charger will draw a large amount of power until the battery pack is full, at which point the charger will continue to draw power at a significantly lower rate. When drawing power at the lower rate the EV can be doing several things including maintaining the charge of the battery pack, pre-conditioning the interior of the vehicle with heating or cooling or maintaining the temperature of the battery pack to improve driving efficiency. To distinguish between charging at maintaining charge, we define a vehicle to be charging if it is drawing more than 1 kW of power; otherwise we define it as “maintaining”.

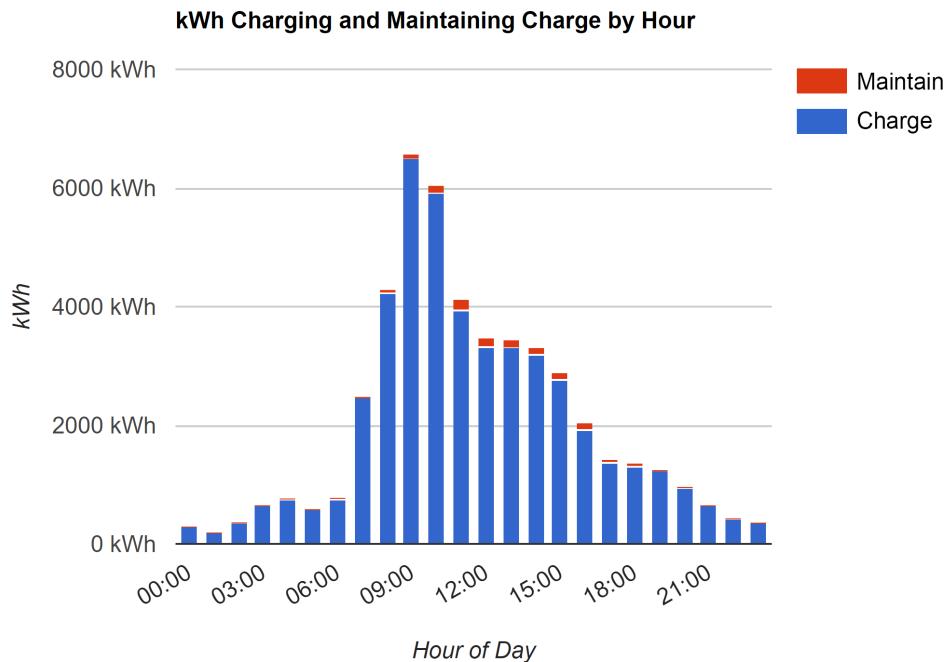


Fig. 9.27 Energy drawn (in kWh) by hour of day stacked with power drawn for charging versus power drawn for maintaining charge.

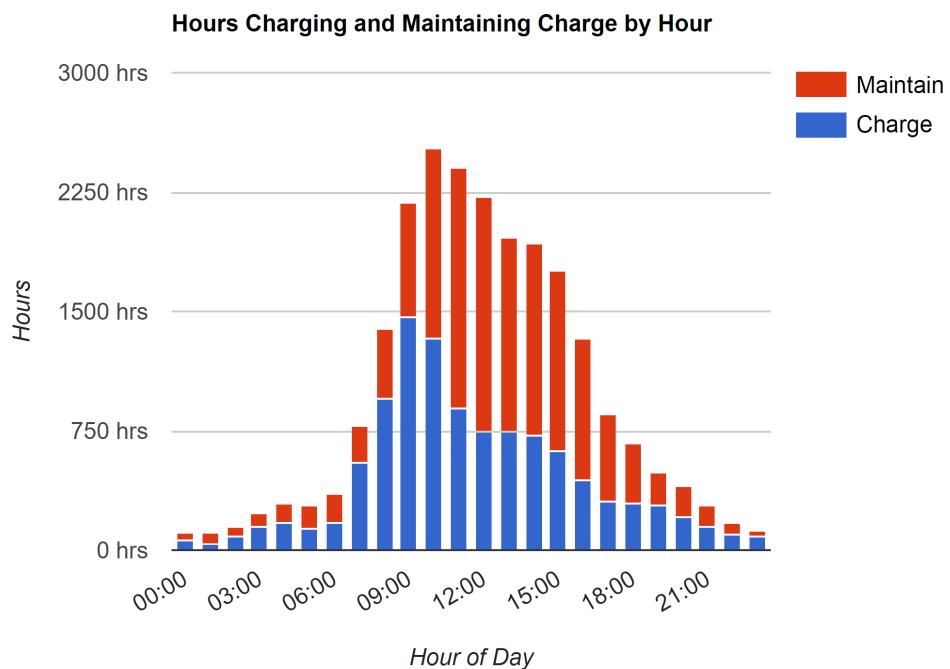


Fig. 9.28 The amount of time spent (in hours) at a charging station with stacked charging time and maintaining charge time.

From Fig. 9.27 it is clear that throughout the day the majority of energy consumed from the station is done during a charging cycle. The energy for charging varies heavily depending on the time of day with the majority of energy being used throughout the day, reducing steadily into the evening and bottoming out around midnight. However, the maintaining charge energy consumption is similar in every hour throughout the day and night. This is because the electric vehicles are sometimes parked at the charging station overnight, and possibly over days when the EV is not being used. The maintaining charge consumption remains steady throughout the time the vehicle is idle.

In Fig. 9.28 we show the amount of time spent for charging and maintaining charge. From the discrepancy between the time required for charging and the time actually spent plugged in at the charging station, it can be seen that the charging stations in many cases are being misused as free parking locations for EVs.

Table 9.3 Charging station statistics June 2012 – March 2019 (81 months)

Total kWh	48788.343 kWh
Estimated Cost (21.87¢ per kWh)	\$10670.01
Estimated Cost (peak/shoulder/off)	\$13891.92
Number of Transactions	6917
Plugged in Time	957 days, 23:29:36
Charging Time	444 days, 2:46:05 (46.36%)
Maintaining Charge Time	513 days, 20:43:31 (53.64%)
Avg Transaction Time	3:19:26
Avg Charging Transaction Time	1:32:27 (46.36%)
Avg Maintaining Time	1:46:58 (53.64%)
Percentage time in use	3.21%
Power used in peak	21785.22 kWh (44.65%, \$9182.47)
Power used in shoulder	21965.21 kWh (45.02%, \$4709.34)
Power used in off-peak	5037.91 kWh (10.33%, \$570.29)

Table 9.3 shows the summary of the charging station usage. From the information collected and automatically analysed, we can draw several conclusions. The flat rate plan of buying electricity is cheaper than the peak-shoulder-off peak plan. Only 46% of the time spent at a station is used actually charging, while for the remaining 54% of the time, the vehicle sits idle and blocks a charging station. This could allow for vehicle-to-grid technologies, however, as shown in [402], V2G applications are not cost effective with current battery technology, as the addition wear and tear from extra charge cycles by far

out-weighs the marginal energy cost. The stations themselves were only in use 3.2% of the time logged, leaving a large proportion of the outlets idle.

9.9.3 Solar PV Monitoring

In the graph in Fig. 9.29 we show the average hourly power output of the 20 kWp solar system at UWA for a typical summer day. The solar system begins generating energy at 6 am and shuts down at 6 pm, with the energy output peaking at 12 pm. The PV system generates approximately 80 kWh per day of operation. So, this system is generating around 30 MWh per year. In comparison, the 23 AC charging stations are using only 5.7 MWh per year on average (see Table 9.3). This shows that one large solar PV installation can effectively power a number of EV charging stations.

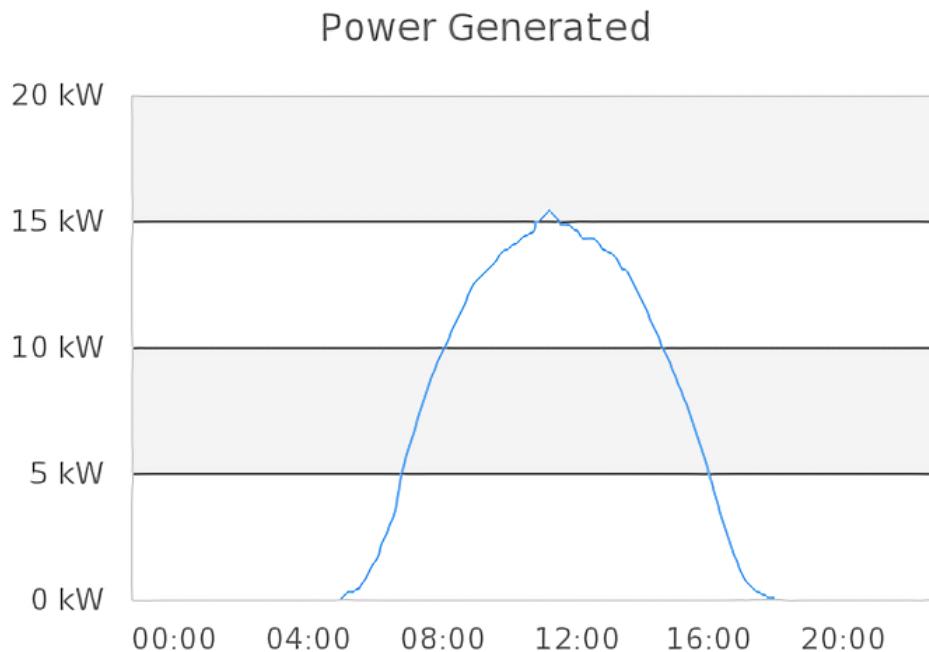


Fig. 9.29 Average power output for 20 kW of UWA's solar PV system on a given day in February.

9.9.4 Heat Maps for EV Tracking

With reference to Fig. 9.18, we generated a heat map of the charging locations for tracked EVs from 2010 to July 2014. By looking at the charge events that took place during the day between 7 am and 6 pm, we can identify possible public locations in the Perth Metro area. The heat map shows several heavily utilised areas, including residential and business locations. One hot spot in Landsdale WA is the location of an EV conversion company that

services most of the tracked EVs. From the heat map, we can determine that this is a place where a charging station would be highly frequented. The heat map also shows hot spots around most existing stations such as at The University of Western Australia.

9.9.5 Charging Infrastructure Usage Forecast

The historical data of the charging stations enables us to forecast the usage on these stations, and consequently predict the EV uptake rate in Western Australia. To achieve this, we have sampled data pertaining to charge frequencies C and energy consumption E for the AC charging network and the DC charging station. Data points were sampled from 1 December 2014 to 1 March 2019 (51 months) for DC charging, and from 1 June 2012 to 1 February 2019 (80 months) for AC charging.

For both data sets, we conducted an augmented Dickey-Fuller test (ADF), which resulted in the rejection of the null hypothesis. This prompted us to fit the forecast over a logarithmic linear model using a log-level regression. We use a natural logarithm as the coefficients on its scale can be interpreted directly as approximate proportional differences, which we describe in our model interpretations.

By modelling the usage of the DC charging station, we present our forecasts of its charging frequency and energy consumption, illustrated as graphs in Figs. 9.30 and 9.31. Note that these models do not account for any usage saturation for the stations, where given the average charge duration of three hours on an AC station and 24 minutes on the DC station, charging frequencies will saturate at 1800 charges per month for the DC station, and 240 charges per month for an AC station, assuming a back-to-back use case scenario, which is beyond the scope of the forecasts.

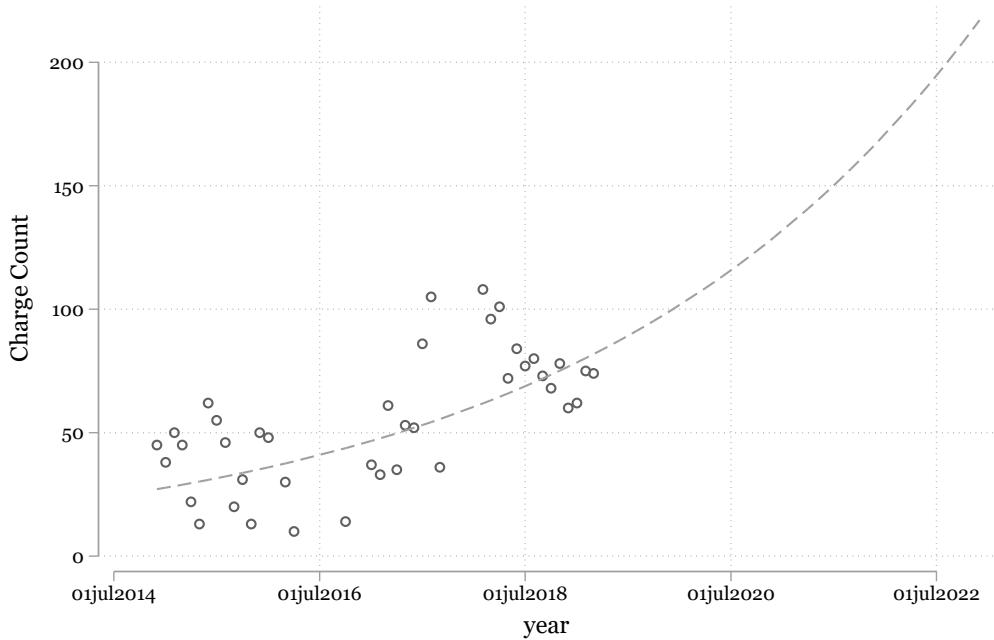


Fig. 9.30 A regression model forecasting the per-month charge frequency on the DC charging station

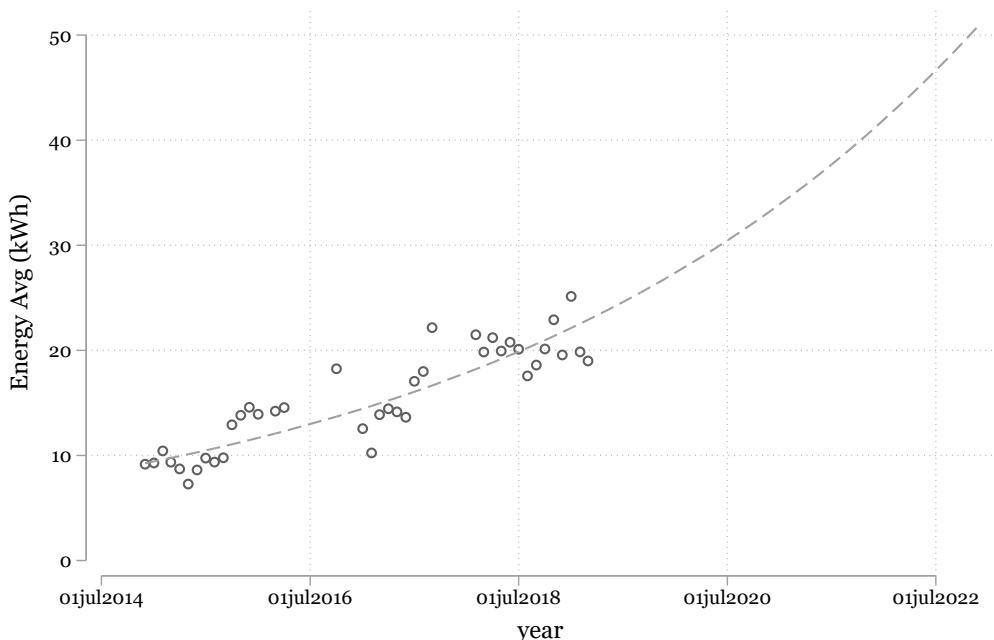


Fig. 9.31 A regression model forecasting the per-month charging energy consumption per charge on the DC charging station.

The coefficients for the models are given as (9.1) for charge frequencies, and (9.2) for energy consumption, with time t measured per day:

$$\ln(C_{DC}) = 0.0007116t - 10.9732 \quad (9.1)$$

$$\ln(E_{DC}) = 0.0005839t - 9.4866 \quad (9.2)$$

We can therefore infer from the coefficients that the charging frequency will increase by 0.07% per day; the average energy consumption will increase by 0.05% per day. This results in a 26.0% annual increase in charge frequency, and a 21.3% annual increase in energy consumption per charge.

Similarly, we model our AC charging station network's usage across the whole network as it compensates for the lower charging frequency on AC charging stations due to its longer charging duration. Here we present models for charging frequency and charging energy consumption as Figs. 9.32 and 9.33.

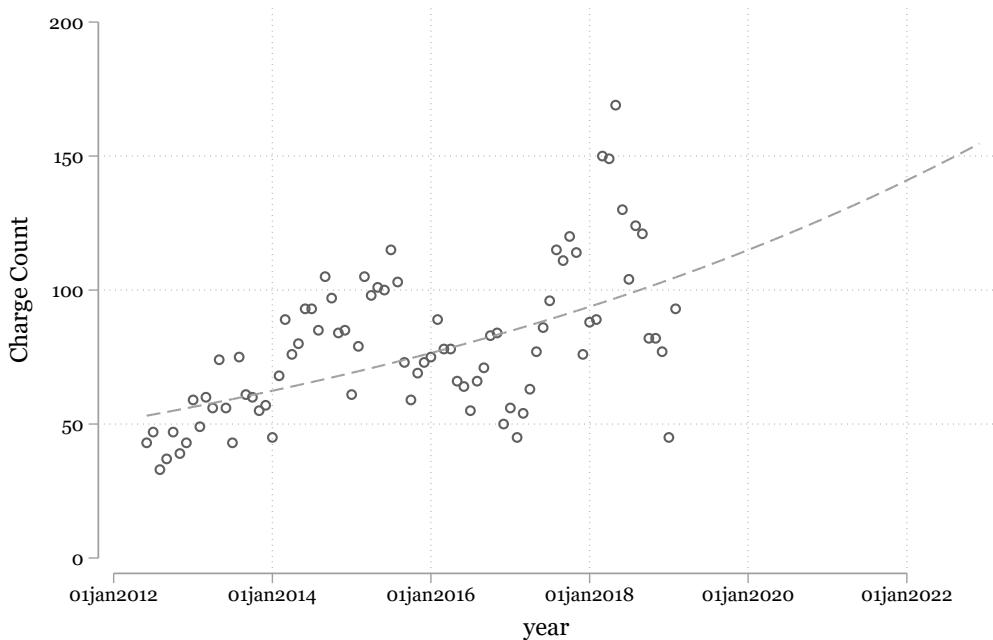


Fig. 9.32 A regression model forecasting the per-month charge frequency on the AC charging network.

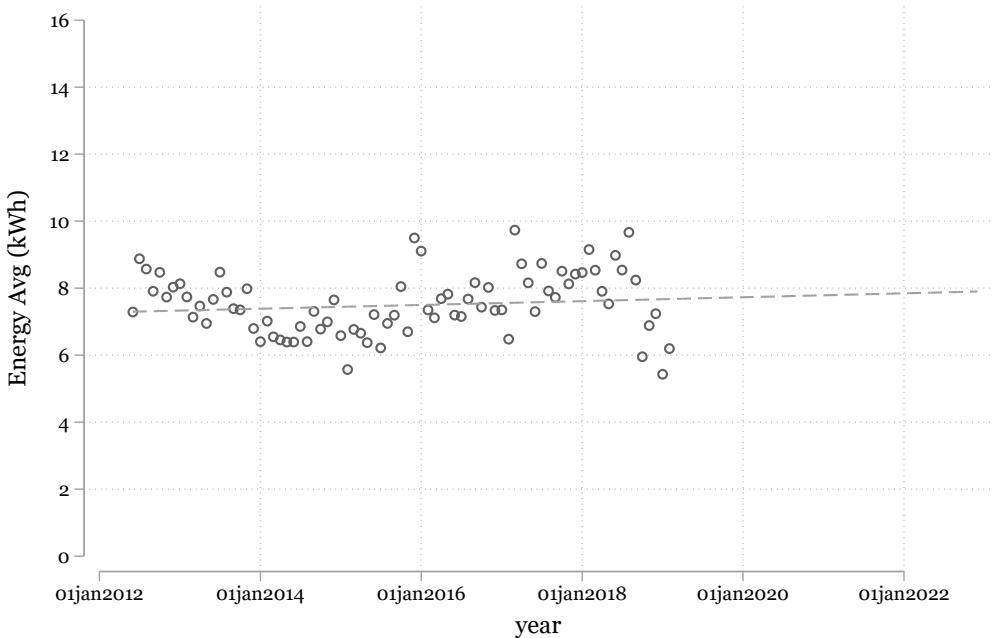


Fig. 9.33 A regression model forecasting the per-month charging energy consumption per charge on the AC charging network.

Likewise, the coefficients for the AC charging network's models are given as (9.3) for charge frequencies, and (9.4) for energy consumption.

$$\ln(C_{AC}) = 0.0002787t - 1.362995 \quad (9.3)$$

$$\ln(E_{AC}) = 0.0000206t - 1.592417 \quad (9.4)$$

From (9.3), we interpret the coefficients whereby the charging frequency for the AC network will increase by 0.03% per day, or 10.2% per annum; the average energy consumption per charge at an AC station increases by 0.002% per day, or 0.75% per annum.

We can thus deduce though these models that DC charging is the more preferred charging method for local EVs, and that the increasing energy consumption per charge could suggest the increasing battery capacity and range for newer EVs. As the number of EV uptakes in WA increases, so does the charging frequency at the DC station. Note that we have observed different charging behaviours across AC and DC charging stations whereby our DC charging station registers more unique users, and many of the charging events registered large energy consumptions, as shown in Fig. 9.34. This likely implies that most users are charging their vehicle from a low charge state. On the contrary, most AC charging users are routine users, whereby the charging bays are occupied for extended periods while the driver is at work,

for example. This explains the low increase in energy consumption across AC charges, as charging events are often top-up charges that recharge the vehicle after a daily commute, and where the vehicle is unlikely to be at a low charge state. However, the increase in charge frequency on the AC network also implies the increasing EV uptake in WA.

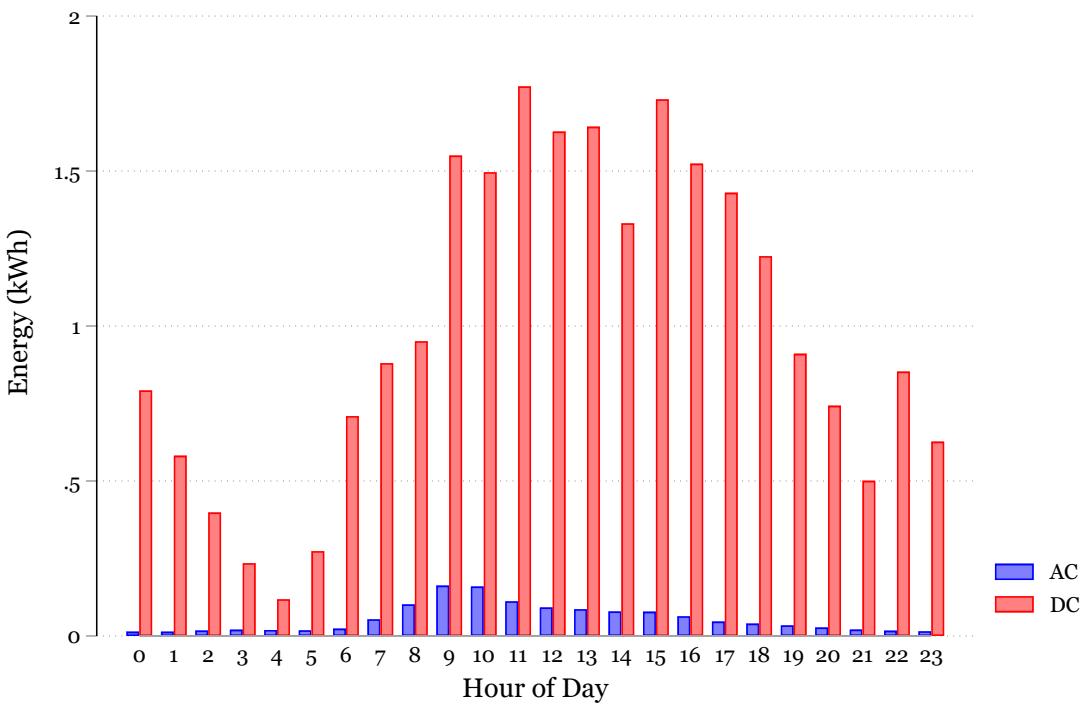


Fig. 9.34 Per-hour energy usage comparison between AC and DC charging stations.

9.10 Conclusion

We have presented in this paper our telematics platform for connected electric vehicles and its infrastructures, REView. This system comprises live data information portals for customers as well as for fleet operators and charging network operators. It provides statistical information on time and location of charge events and includes a time-of-use billing system. It interfaces with charging stations, vehicle-based data loggers and solar PV systems. This required configuration and testing for each of the different devices in parallel with server and database development. The software was written for server-based and client-based processing and data display. Each of the different levels of this project (server, data processing, and interface) was developed in tandem to ensure integration. The software was designed in a modular way with separate scripts for individual features, making unit testing easier, reducing integration

problems and isolating failures. All of the programming languages used in the system are interpreted, which means that design changes could be made very quickly. The main contribution of REView is that it consolidates incoming data from connected vehicle fleets, charging stations and power generators into a unified platform to improve the efficiency of information presentation. These are presented entirely as a web-based solution for increased accessibility and are supplemented by additional features such as billing which supports the monetisation of the system. From the data collected and analysed, we can deduce that solar technology is an effective way for offsetting energy required for charging EVs at public charging stations and place-of-work. For home charging, energy is mostly required outside of solar generation hours and would need to be provided by a domestic energy storage system. A 20 kW solar PV system was more than enough to offset the energy used by EV charging at 23 public charging stations. The results produced from this system has, throughout the years, enabled us to perform various analyses on the EV landscape within Western Australia.

Moving forward, with the increase of data volume and infrastructures, future plans for REView will involve its eventual compliance with the arrival of the Internet of Vehicles (IoV) standards. With this proposal, we are planning to integrate REView as a cloud application while utilising the Platform as a Service (PaaS) environment. This will enable us to streamline further developments on REView and to increase its modularity through more efficient workflows while improving scalability.

Chapter 10

A Comparative Study of AC and DC Electric Vehicle Charging Station Usage

Fast-DC charging stations can charge an Electric Vehicle (EV) several times faster than Level-2 AC charging stations. Using a network of DC charging stations, it becomes possible to use EVs for long distance, cross-country driving with only short recharging stops. This paper examines and compares typical customer usage patterns at DC fast-charging stations (50 kW) versus Level-2 AC stations (7 kW). It includes data collected from the University of Western Australia's AC and DC charging network in the Perth metropolitan area, as well as from stations along the highway connecting Perth to Augusta in the rural South West of Western Australia (over 300 km apart). A cost model is also drawn up to calculate the operating cost and break-even requirement across several different styles of charging stations. User behaviour and adoption of certain charging infrastructure is crucial for the take up of electric vehicles in general. EV charging standards and infrastructure availability have, therefore, a fundamental influence on the electrification of transport.

10.1 Introduction

Electric vehicles (EVs) are an environmentally friendly alternative to traditional internal combustion engine vehicles (ICE), which are a major contributor of carbon emissions [362]. EVs are emission free if charged from renewable energy sources and they improve urban air quality as well as fuel security [403]. Additionally, they are becoming more and more common on the roads today, with an increase on the roads worldwide from 100,000 vehicles in 2012 to over 1 million in 2016 [404]. This paper discusses the data collected from three different sources — the Western Australian Electric Vehicle Trial [347], The University of

Western Australia's fast-charging station [405] and the RAC-funded Electric Highway in Western Australia [406]. Comparing these trials allows the assessment of different charging infrastructure types, different locations and different usage patterns between paying and non-paying customers (e.g. free stations). The current state of EV charging technology, specifically international standards and their adoption in different countries, is also examined by using publicly available information [24]. Electric vehicle adoption has a direct link to the availability of fast-charging infrastructure [407] (though not without contention [408]). The infrastructure installation and maintenance of these charging stations is an expensive process, so having greater clarity on usage patterns can assist organisations in their decision making.

This paper's aim is to give an overview of all charging infrastructure developed to date and the overall necessity of an electric vehicle charging station network. The University of Western Australia's Renewable Energy Vehicle Project (REV) installed Western Australia's first EV charging infrastructure in 2010 as a series of 23 Level-2 ("medium fast") AC charging stations (7.7 kW), funded through the WA Electric Vehicle Trial in combination with an ARC Linkage grant [347]. REV later installed Australia's first commercial CCS fast-DC charging station (50 kW) in 2014.

Although the EV Trial and REV/UWA had proposed an Electric Highway through Western Australia with several partners, it took over two years until RAC WA eventually funded this network. Funds were given to nine rural communities to install a pair of AC and DC charging stations at each location, plus a tenth at the RAC headquarters in West Perth. The rural locations are Mandurah, Harvey, Bunbury, Busselton, Dunsborough, Margaret River, Augusta, Donnybrook and Nannup. While power is provided free of charge at all UWA stations, users of the Electric Highway have to pay \$0.50 per kWh. This is twice the amount of the domestic energy rate, which makes these stations unattractive to local EV owners.

The remainder of this paper is organised as follows. Section 10.2 presents the various types of EV charging infrastructure from a global to local standpoint. Section 10.3 explores different EV charging methods and the preferred methods of adoption. Section 10.4 analyses and compares data collected from the UWA AC and DC charging stations, and the local Electric Highway network. In Section 10.5, a cost model then drawn using this data from the UWA stations. The data analysis is validated in Section 10.6 using a similar study before a summary and concluding remarks are drawn in Section 10.7.

10.2 AC and DC Charging Infrastructure

Countries around the world have adopted different charging standards, and in some cases more than one. The United States and Canada have passed legislation to adopt the IEC 62196 Type-1 standard (single-phase AC), while the European Union has adopted the IEC 62196 Type-2 charging standard (three-phase AC). For DC, these countries use the compatible Combined Charging System (CCS) standard, again as Type-1 (USA, Canada) and Type-2 (Europe), which allows vehicle manufacturers to use a single combined vehicle inlet for either AC or DC charging. France and Italy initially adopted Type-3 (Scame) connectors and are currently in transition towards Type-2 connectors.

Japan uses almost exclusively its CHAdeMO standard for DC charging, while China uses its GB/T standard. Some countries, like Australia, have failed to adopt any national standard and then had to suffer the consequences. A mix of Type-1 and Type-2 charging stations were installed in different states in Australia initially when mostly Type-1 vehicles were imported into the country (no EVs were ever produced in Australia). This changed in late 2017, when leading vehicle manufacturers decided to change over to Type-2 for newly imported vehicles, and other manufacturers can be presumed to follow. This leads to presumptions that the whole country should adopt Type-2 stations as a standard which would cause major problems for both charging station operators, as they could not serve all cars (unless they installed Type-2 stations, which have exchangeable power cables), and vehicle owners, who would not be able to charge their cars on CCS stations of the wrong type. Using Type-2 chargers, however, makes sense for Australia, as the country does have a three-phase power grid.

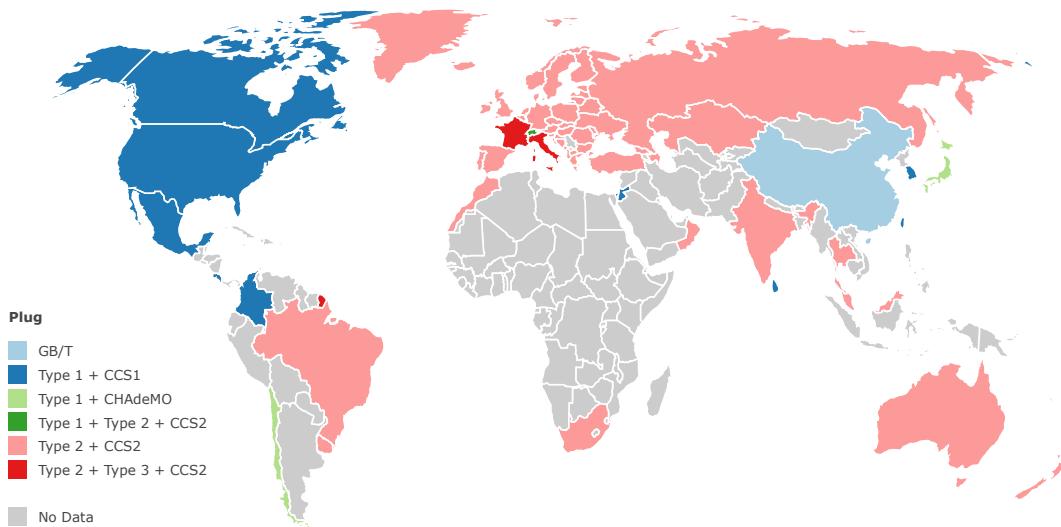


Fig. 10.1 Global EV charging inlet adoption [24].

Fig. 10.1 shows each country's predominant AC charging standard in combination with the adopted DC standard. The information used to generate this chart was extracted from the publicly available PlugShare website [24], which claims to be the most accurate source of charging stations worldwide, with approximately 112,000 locations and more than 170,000 outlets. Countries that have insufficient or no charging station data are not labelled.

There are several charging standards omitted from this graph, perhaps most importantly the Tesla charging stations, which provide brand-specific chargers in all countries where they distribute their vehicles. In Australia, China and Pakistan, Tesla DC charging stations outnumber all other DC stations, as shown later in Fig. 10.4. When only considering the Type-1, 2 and 3 connectors, Tesla stations outnumber all others in Serbia and Hong Kong.

Charging stations in Western Australia are progressing towards Type-2 chargers. This is inherently visible in recent installations of charging stations, as well as the local charging station networks as follows:

The REV/UWA fast-DC station supports:

- DC CCS Combo Type-2
- DC CHAdeMO

while, the RAC stations provide:

- DC CCS Combo Type-1
- DC CHAdeMO
- AC Type-2 (Mennekes) [409]

This variety of outlets allows the stations to support the different EV standards currently in use. All RAC DC-stations have a Level-2 AC station next to them, allowing vehicles without fast-charging support to charge using an SAE J1772 (Type-1) connector. The power and voltage outputs for charging stations that are commonly found around southwest WA is tabulated as Table 10.1.

Table 10.1 Outputs of various charging stations in southwest WA

Type	Phase	Output	Value
DC	n/a	Max output current	120 A
		Max output power	50 kW
		Output voltage range	50 – 500 V DC
AC	3	Max output current	63 A
		Max output power	543 kW
		Output voltage range	400 V AC
AC	1	Max output current	32 A
		Max output power	7.2 kW
		Output voltage range	230 V AC ($\pm 10\%$)

10.3 Types of EV Charging

There are several different methods of EV charging. When discussing the efficiency of the various methods this paper does not include any transmission losses or power generation. Various power generation methods for electric vehicle charging can be found here [410, 411], with an in-depth comparative study in [412].

Electric vehicles are traditionally charged off AC mains. The AC power needs to be converted into DC power by a rectifier inside the vehicle. Although this makes the charging infrastructure quite simple, each EV must carry an expensive and heavy AC-to-DC converter element. In many cases, first generation EVs are equipped with only a basic AC charger, useful for Level-1 home charging (up to 2.4 kW), but not taking advantage of the higher AC currents available at Level-2 charging stations.

The higher the output power of a charger, the heavier and larger the charger must be. Electric vehicles carry this internal charger as a part of their design, to allow charging off a standard electric power point. But at higher currents this method becomes impractical, as larger and heavier AC-DC converters would have to be carried.

DC stations offer a solution for this. Very little electronics is required in the EV itself, as most of the hardware is included in the charging station. First, EV and station negotiate the correct DC voltage level over a communication link. Then the station provides the correct DC level at a much higher current than is feasible with AC charging. The communication protocol used between the charging station and the vehicle is defined by IEC 61851-1 [413].

Signal data lines are part of all charging stations, whether AC or DC, and are fully defined in IEC 62196 and IEC 61851. They are also part of safe-guarding stations and EVs against failures and potential hazards. The stations used in the UWA EV trials were equipped with internal over-voltage/over-current protection, over-heating control, and protective earth detection. The stations were also installed on separate circuits with dedicated RCDs, following the conventions of AS/NZS 3000 Wiring Rules [414].

10.3.1 Typical Charging Cycle

Electric vehicles go through three or more different states when charging. This can vary from vehicle to vehicle. At a DC charging system, a battery is typically filled up to only 80% capacity, as the charging rate significantly slows down for the remaining 20%, due to the battery's increase in internal resistance [415].

At most AC charging systems, an EV is fully charged to 100%, but even then, it continues to draw a small amount of power to maintain the charge of the battery at the top level. This is to counteract the parasitic draw of various electrical systems in the vehicle, and keep the battery full. Some EVs also condition the battery pack through heating or air conditioning, in order to increase charging efficiency [416, 417] or simply pre-condition the cabin through heating or cooling as a comfort feature for the driver.

Fig. 10.2 shows an EV charged from about 25% to 100% state of charge (SoC) on the DC charging station at UWA. Although this station can provide 50 kW of power to the EV, charging begins at 40 kW, and as the battery level rises the output power is further reduced. For this reason, all DC charging stations stop charging at 80% SoC. The remaining 20% of charging can take longer than the initial 80% and would preclude other customers from using the charging station.

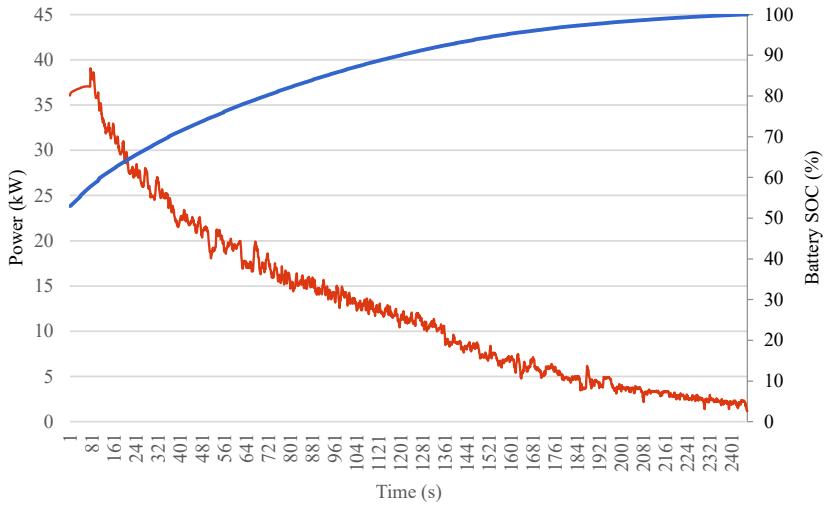


Fig. 10.2 Battery charge rate (kW) in red and State of Charge (%) in blue over time.

10.3.2 Limitations on Charging Speed

The following factors limit the effective charging speed (or charging power) of a charging station:

- **Temperature of batteries** — Very high, as well as very low temperatures, require lower charging rates.
- **Temperature of tolerable heat dissipation in the power electronics** — Examples include charging in closed environments, such as a domestic garage has to limit heat dissipation in order to reduce any fire hazard.
- **Health of the battery** — Ageing or unhealthy batteries exhibit a larger variation in individual cell voltages and will therefore require more time for balancing during the charging process.

10.3.3 Authentication and Billing

Charging station operators may want to control access by some form of user authentication and bill users for their power usage. Authentication can take place in several different ways, including locally at the station (allowing for the station to control authentication without needing an internet connection), or via a server. The charging stations in the REV/UWA trials use RFID cards that were provided to station users. These can be authenticated against an external server. A local whitelist is useful in the event that the station loses its network connection.

Interfaces to manage these stations are also necessary to collate and display the data to users or operators. The Open Charge Point Protocol (OCPP) was developed in an attempt to foster global development, adoption and compliance of communication protocols [418]. This common protocol means that stations from different manufacturers can be controlled by a single OCPP server.

10.3.4 Driving Efficiency and Battery Size for EV

There is a significant variation in energy efficiency for EVs [419], ranging between:

- BMW i3: 129 Wh/km
- Mitsubishi i-MiEV: 135 Wh/km
- Nissan Leaf: 173 Wh/km
- Tesla Model S: 186 Wh/km

Also, each of these vehicles has a different battery capacity, ranging from the Leaf's 16 kWh battery to the Tesla Model S 100 kWh battery. For the sake of comparing the different charging stations, two typical scenarios are taken, representing both ends of the spectrum:

- Case 1: 16 kWh, 135 Wh/km
- Case 2: 100 kWh, 200 Wh/km

10.3.5 Inductive Charging

Inductive charging allows wireless charging of an EV via an electromagnetic field. There is a coil in the vehicle and one located below the vehicle, usually embedded in a mat. Of the various charging methods, this is the least efficient but the most convenient, as it does not require the driver to plug the vehicle or even to carry a cable. A major issue that manufacturers need to address is that the efficiency is reduced if the coils are not aligned correctly when parked. Only 5% of the surveyed EVs parked within the tolerance level of the coils, so this requires either a movable coil or a self-parking vehicle to reduce this issue [420]. The power transfer efficiency varies depending on the manufacturer, air gap and power rating. In seven different studies between 2011 and 2014 these values were found to be between 83% and 92% [421].

10.3.6 Level-1 Charging (IEC 62196-3 Mode 2)

Level-1 is limited by the rating of a standard power outlet in the respective country. In Australia, the maximum power to be drawn at Level-1 is 240 V at 10 A (2.4 kW). Electric vehicles are mostly fitted with these chargers internally, as they are comparatively lightweight.

10.3.7 Level-2 Charging (IEC 61851-3 Mode 3)

Level-2 charging allows the vehicle to draw a higher current up to 32 A at 240 V (7.7 kW for single phase or 23 kW for three phase). Like Level-1 charging, this relies on the internal charger of the vehicle.

10.3.8 DC-Fast Charging (IEC 61851-3 Mode 4)

DC-fast charging ranges from 50—900 V DC and has a range of varying current outputs. Unlike other stations, the charger is not inside the vehicle, but within the station itself. The station's charger is controlled by the vehicle via data lines. The stations in WA support up to 125 A (50 kW), while Tesla's Supercharger already charges at 120 kW [25]. Recent CCS 2.0 stations are supplying up to 350 kW per station [422], while future CCS DC chargers will deliver up to 450 kW per station [423, 424].

10.3.9 Alternate Methods

Another potential method of converting AC power into DC for charging the vehicle is through the use of integrated motor drives where the vehicles' motors are used to do the conversion [425].

10.3.10 Charging Speed Comparison

Table 10.2 compares the various charging techniques for different battery types and charging levels.

Table 10.2 Charging style configuration and time for small and large battery packs

Charging Type	Charge level	Charging time	
		16 kWh	100 kWh
Level-1	100%	5 hours	33 hours
Level-2 (1-phase)	100%	2 hours	11 hours
Level-2 (3-phase)	100%	40 minutes	3.7 hours
DC 50 kW	80%	15 minutes	1.5 hours
DC 150 kW	80%	5 minutes	32 minutes
DC 450 kW	80%	1.7 minutes	10.7 minutes

10.3.11 Australian Charging Standard Preference

Fig. 10.3 presents a chart of the number of charging stations installed in Australia. In total 416 stations have been registered at online platform PlugShare.

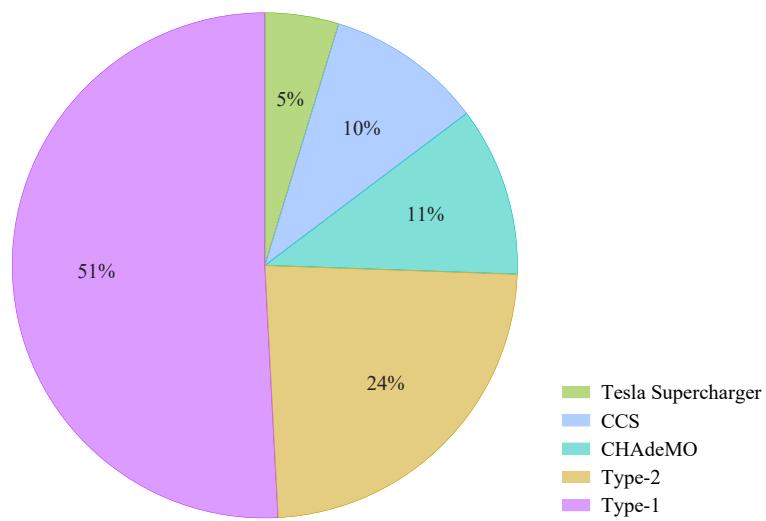


Fig. 10.3 Australian charging inlet adoption.

It was observed that there are slightly more installations for CHAdeMO than CCS in Australia, but CCS is expected to take over within two years, as there is a shift to more CCS inlets from major car manufacturers. BMW as one of the market leaders, has decided to swap over from Type-1 to Type-2 EV inlets for the Australian market and it is expected that will trigger other OEMs to follow suit. Standards Australia has so far failed to recommend

any charging standard although the topic has been debated for over ten years. Out of the 416 stations registered, there are:

- 20 Tesla Superchargers,
- 41 CCS,
- 45 CHAdeMO,
- 98 Type-2, and
- 212 Type-1 stations.

10.3.12 International EV Plug Adoption

The global adoption of DC charging inlets from about 147,911 charging stations worldwide was also analysed, as illustrated in Fig. 10.4.

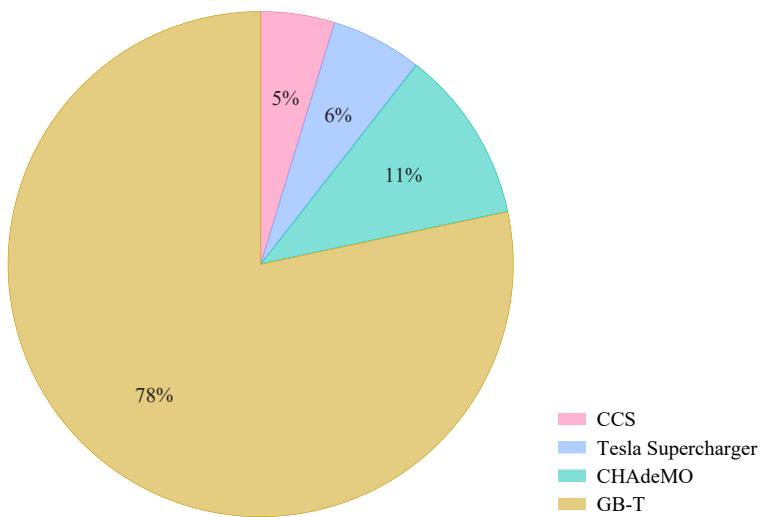


Fig. 10.4 Global DC charging inlet adoption.

The Chinese GB/T standard has the highest share of all worldwide charging installations, but only exists in China, due to the Chinese government's New Energy Vehicle (NEV) initiatives in 2009, which catalysed the installations of charging stations around the country [426]. CHAdeMO, originating in Japan, was introduced prior to CCS and has many installations in Japan and North America, leading to its higher market share. Of the charging stations in Figure 4, there are:

- 115,776 GB/T DC chargers, of which 66,059 are combined AC/DC stations [365]

- 16,639 CHAdeMO stations,
- 8,496 Tesla Superchargers, and
- 7,000 CCS stations [427].

10.4 Analysis of Charging Station Usage

Usage patterns of the UWA/REV charging station network were analysed, comprising 20 7 kW AC chargers and a 50 kW DC-fast charger. Data was obtained during the period of 1 June 2012 through 31 January 2018 for the AC stations, and from 12 November 2014 through 13 October 2017 for the DC station, unless stated otherwise. Short dates are presented in the format “dd/mm/yyyy”.

10.4.1 AC Charging and Maintaining Charge

UWA/REV stations are Level-2 stations which typically require a few hours to fully charge a vehicle and therefore many users leave their vehicles charging while they are at work. Many vehicles are hence idly plugged into the charging station even when charging has been completed. Of course, this is mostly because no fees are being collected for charging or for parking at these stations. In this section the charging patterns of the UWA AC stations were analysed across the data summary tabulated in Table 10.3. To ensure that only real charging events are logged, events that are less than five minutes long are filtered.

Table 10.3 Total statistics for the AC stations across the sample period

Number of events	4,444
Total energy delivered	29,206 kWh
Total plugged in time	672 days

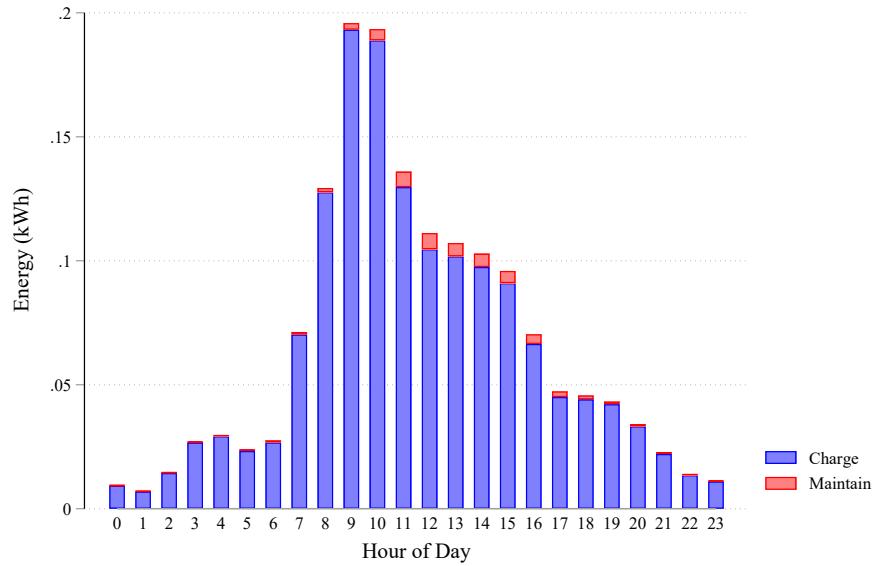


Fig. 10.5 The energy delivered during charging and maintaining charge on average for an AC station at each hour of day.

Fig. 10.5 illustrates the average energy delivery of an AC charging station at each hour of day. Energy delivery increases and peaks at 9 am because it is then when many users arrive at work to charge their vehicle. The energy used to maintain charge increases and peaks at 12 noon, when most of the vehicles have been fully charged. That said, the average energy used to maintain charge on a vehicle averages at only 2.19 Wh, which is significantly below the average charging energy of 63.3 Wh.

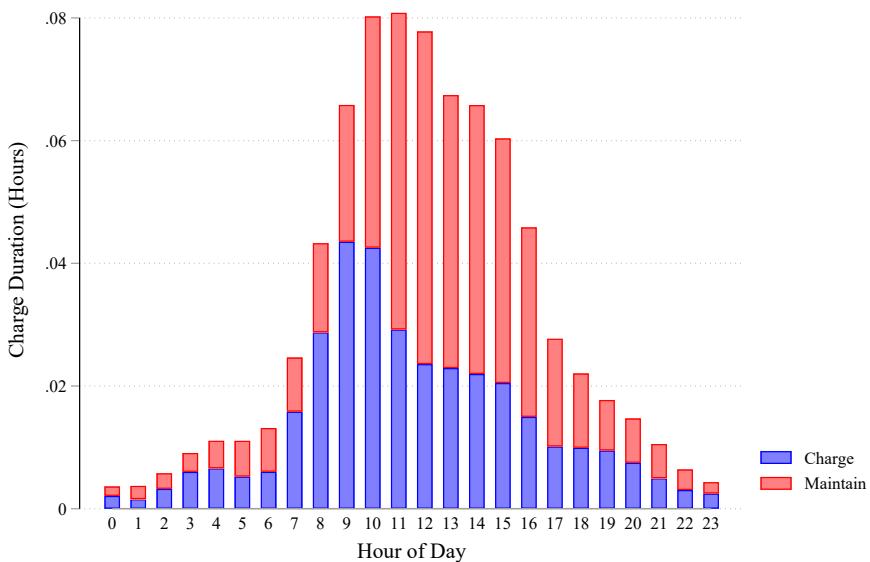


Fig. 10.6 Durations of charging and maintaining AC charge by station time on a vehicle per station at each hour of day.

Fig. 10.6 shows the average time spent for an AC charging station to be in charging or maintaining state over the time of day. As most charging events commence around 9 am to 10 am, more time is spent charging at the station, and as the vehicles get charged, the “charge bar” in the graph eventually transitions into the “maintain bar” for the rest of the vehicle’s plug-in time. The charging stations free up in the evenings, before demand increases again in the next morning. In total, the UWA/REV AC stations have spent 312 days charging and 405 days maintaining charge over the data collection time frame, which averages to 0.342 hours charging and 0.431 hours maintaining per day per station. The average charge event at an AC station takes 3.91 hours and uses 6.66 kWh of energy.

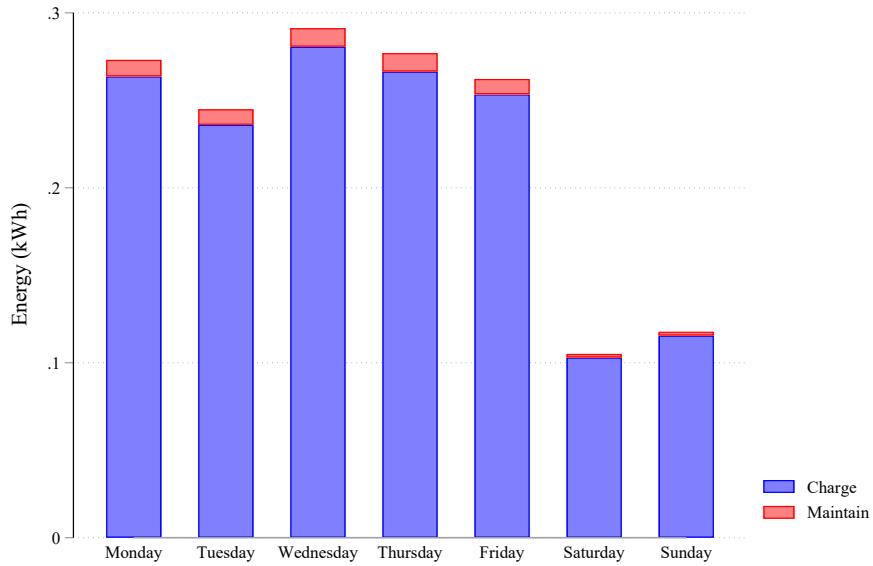


Fig. 10.7 The energy delivered during charging and maintaining charge on average for an AC station for each day of week.

By analysing the charging patterns across a week, Fig. 10.7 indicates that more energy is used during the weekdays for charging, at an average of 0.27 kWh per day. Charger usage drops significantly on weekends to less than half at 0.11 kWh per day.

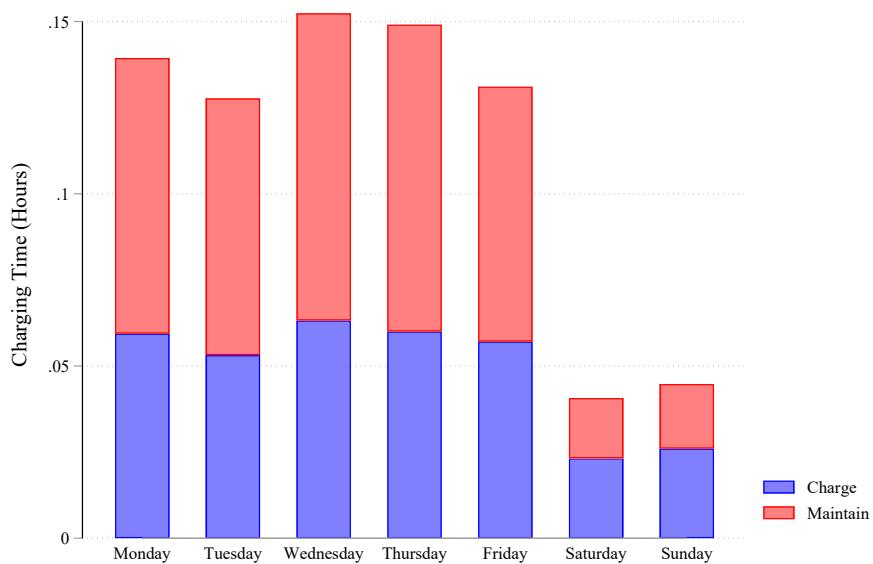


Fig. 10.8 The time taken to charge or maintaining AC charge on a vehicle for each day of week. (CS vs DC).

When comparing charge times across the days of the week, Fig. 10.8 shows that charging duration decreases during the weekends by 53% on average, each station spends 0.14 hours charging and maintaining on weekdays, and 0.043 hours on weekends. This is consistent with the results from Fig. 10.7.

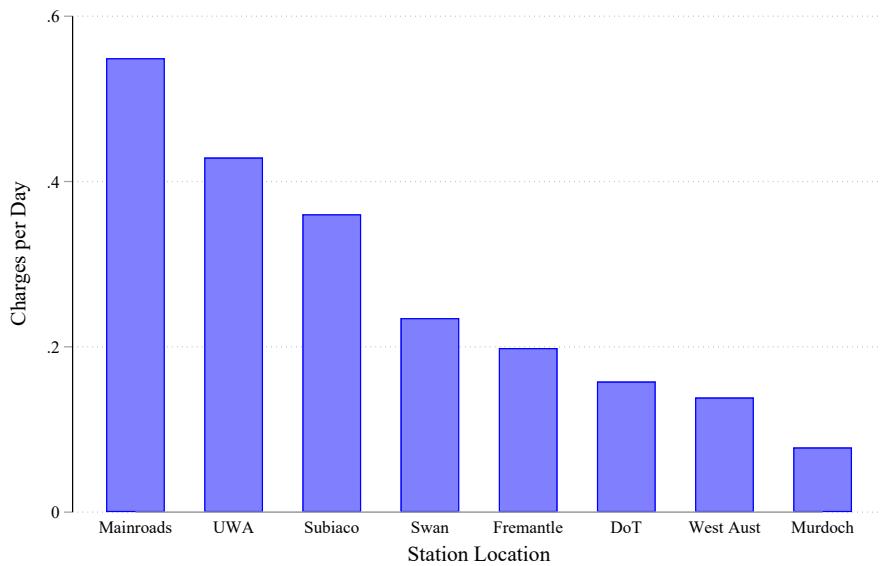


Fig. 10.9 Comparison for the number of chargers per day between each of the AC stations.

A comparison of the average daily number of charge events for each UWA/REV AC charging station is shown in Fig. 10.9. The low number of charges per day is mostly due to slower charging on AC and the fact that cars are not collected when charging is finished, so charging bays are not freed up for new customers. The charger locations near offices and work locations enable their staff to charge on a more consistent basis, but it leaves the stations vacant on weekends. This is evident in the UWA Computer Science and Main Roads stations, where staff charge their vehicles daily on weekdays. The stations in the suburbs of Subiaco and Fremantle are in general parking areas and are more accessible to the public. However, the low EV penetration rate combined with the long charging times contributes to lower charging numbers for these stations. Overall, UWA/REV AC stations have on average 0.27 charges per day, ranging from 0.08 to 0.55 charges per day.

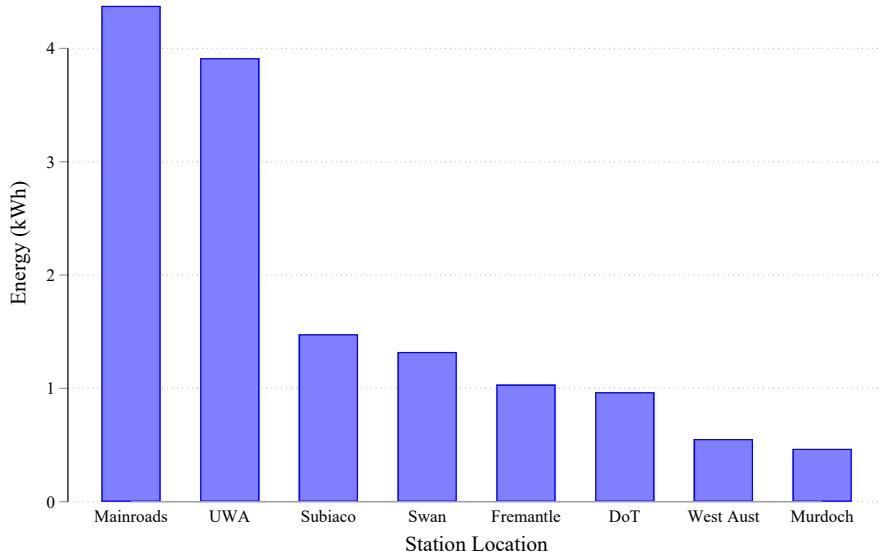


Fig. 10.10 Comparison for the energy delivered at each station per day across each of the AC stations.

By comparing the energy delivery per day for each AC station, Fig. 10.10 shows a similar trend to Fig. 10.9, whereby a higher charge per day will contribute to a higher energy usage for each station. Each station delivers on average 1.76 kWh per day, with the Main Roads station delivering the most energy at 4.38 kWh per day.

10.4.2 AC versus DC Station Comparison (CS vs DC)

A comparison of the UWA/REV fast-DC station against the AC station network at the UWA Computer Science (CS) car park is shown in Fig. 10.11. As expected, the DC station delivers much higher energy amounts in a shorter time than the AC station.

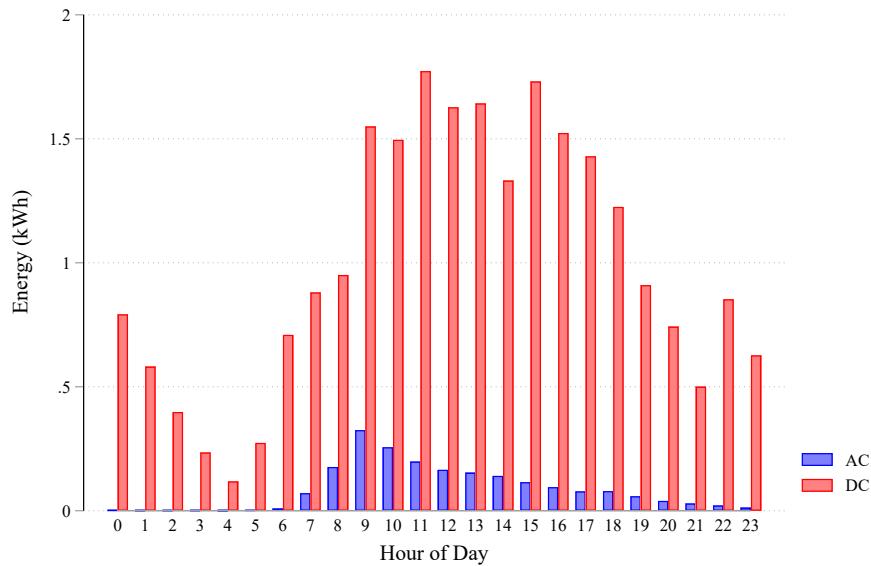


Fig. 10.11 The differences in energy delivered by an AC station versus a DC station at each hour of day (CS vs DC).

Fig. 10.11 compares the energy usage between the DC station and the AC station across each hour of day based on its charge events. The energy used for the AC station is the sum of its energy delivery during charging and maintaining phases. The DC station uses 7.78 times more energy per hour than the AC station. On average, the AC station delivers 0.09 kWh per hour, while the DC station delivers 1.0 kWh per hour. Also, while the energy delivery at the AC station peaks at 9 am, charging events at the DC station usually peak later in the morning and continue into the afternoon and evening. The quick charging capability of the DC stations means that users can often charge their vehicle en route to their destination.

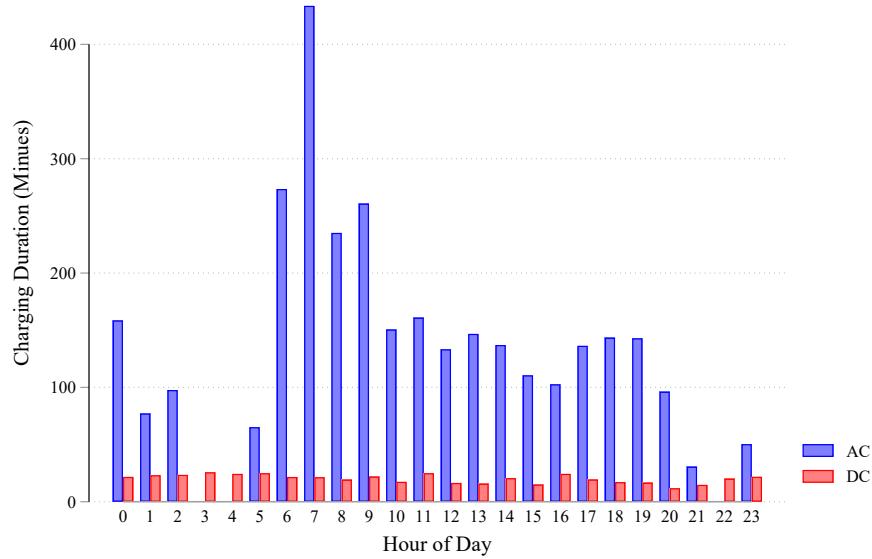


Fig. 10.12 The difference in charging time on an AC station versus a DC station at each hour of day.

Fig. 10.12 compares the charging duration between the UWA DC station and the UWA AC station per hour of day. Charging durations for the AC station is a sum of its charging and maintaining phases. On average, vehicles are tethered to an AC station 6.5 times longer than at a DC station. Even so, there is only a 13.3% difference in the energy delivered between the DC and AC charge events.

It is noted that while charging durations on the AC station are longest for morning arrivals, there is no such noticeable trend for DC charging durations.

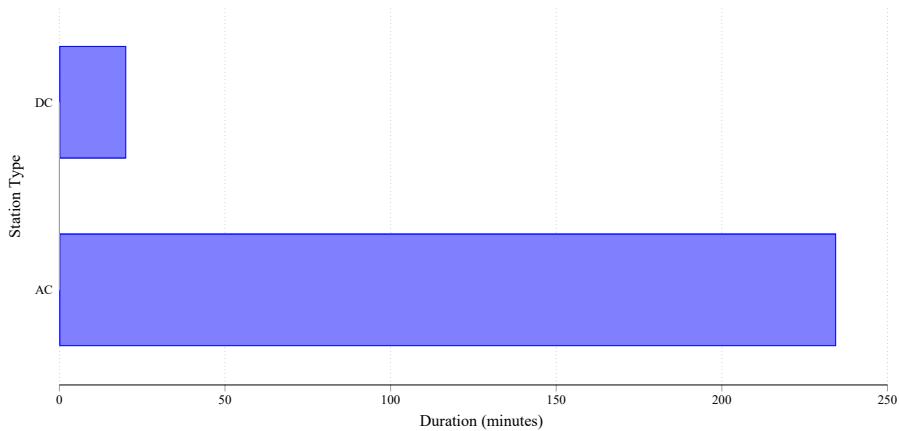


Fig. 10.13 The average charging duration for a DC and AC charge event.

Fig. 10.13 compares the average charging duration for each charge event on the REV/UWA DC and AC stations. The data for AC charging is averaged across all charging events on all AC stations. The average AC charging time across all metropolitan stations is 235 minutes (3h55min) for 6.65 kWh, while the average DC charging takes 20.2 minutes for 7.80 kWh.

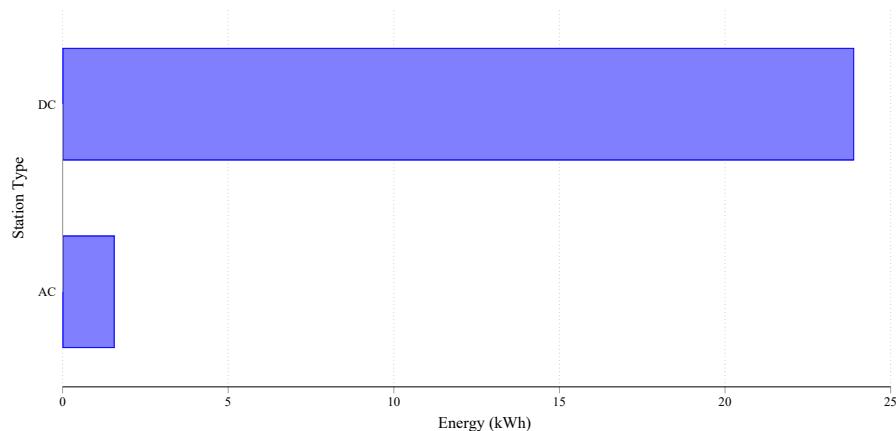


Fig. 10.14 The daily energy delivery for a DC and AC station.

When comparing the daily energy delivery between the AC and DC charging stations, Fig. 10.14 illustrates that the DC station typically delivers 23.9 kWh per day, and 1.57 kWh per day for an AC station.

10.4.3 DC Station Comparison

Comparing data from the UWA DC station with the Electric Highway DC stations in the WA South-West, the number of charge events, charging duration and the energy delivered is considered.

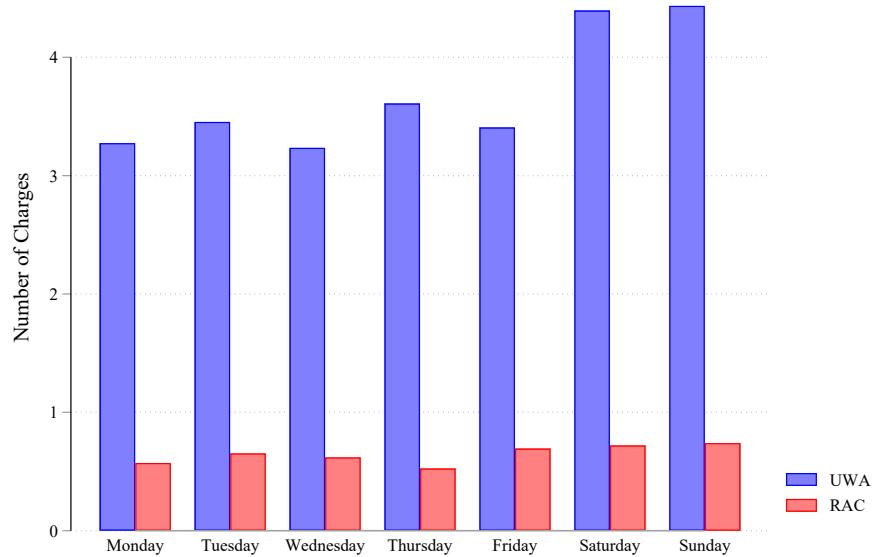


Fig. 10.15 Number of DC charge events per station per day of week between the UWA (12/11/2014 to 13/10/2017) and the Electric Highway (RAC) (02/03/2016 to 20/09/2016).

The number of charges per day of week in Fig. 10.15 compares the average charges at UWA with the RAC stations. The charging data from the RAC stations is compared with the UWA/REV data across 2,370 recorded charging instances beginning from 12 November 2014 to 13 October 2017. The average number of DC charge events is 3.35 per day at UWA, but only 0.65 per day for the average Electric Highway station.

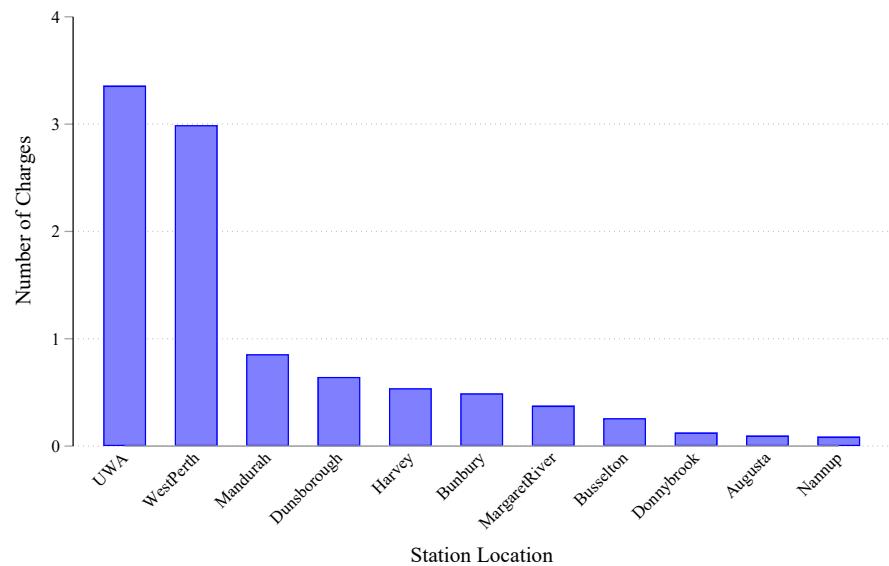


Fig. 10.16 The number of charges per day for each station from the UWA (12/11/2014 to 13/10/2017) and the RAC (02/03/2016 to 20/09/2016).

By comparing the number of charges per day for each station, Fig. 10.16 shows that the stations closer to the Perth CBD are used more often than those in regional areas. The RAC West Perth station has 3.0 charges per day, whereas the UWA station has 3.35 charges per day. The regional stations have significantly fewer than 1.0 charge per day, with Mandurah at 0.86 charges per day, and the lowest being Nannup at 0.087 charge events per day. This puts the average number of charge events of an Electric Highway station to 0.65 charges per day.

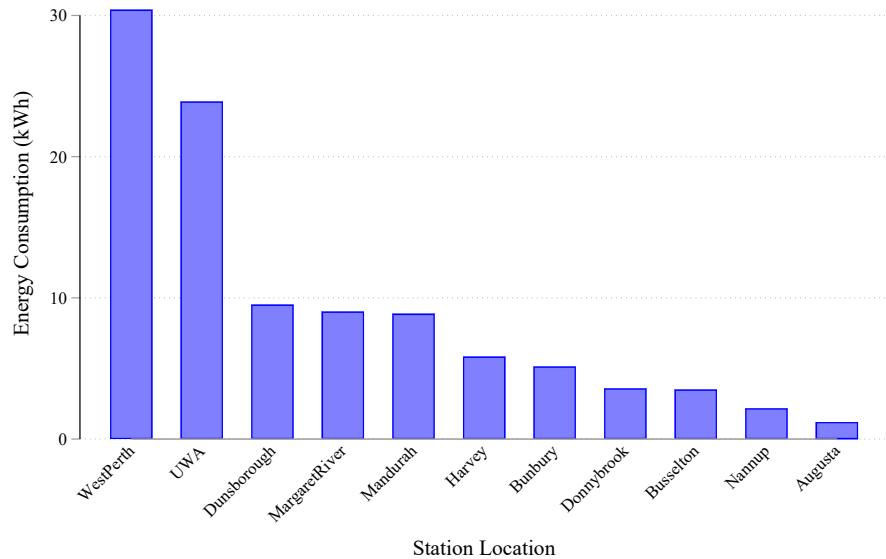


Fig. 10.17 The amount of energy in kWh delivered per day for each DC station from the UWA (12/11/2014 to 13/10/2017) and the RAC (02/03/2016 to 20/09/2016).

Energy delivery across all stations per day is in line with their number of charge events in Fig. 10.16, whereby stations in the city deliver more energy per day. However, despite their lower charging frequency, regional stations deliver more energy per charge as illustrated in Fig. 10.17. The West Perth station delivers the most energy at 30.4 kWh per day, followed by the UWA station at 23.9 kWh. The Augusta station delivers the least amount of energy at 1.2 kWh per day. The average energy delivered by the Electric Highway stations comes to 7.92 kWh per day.

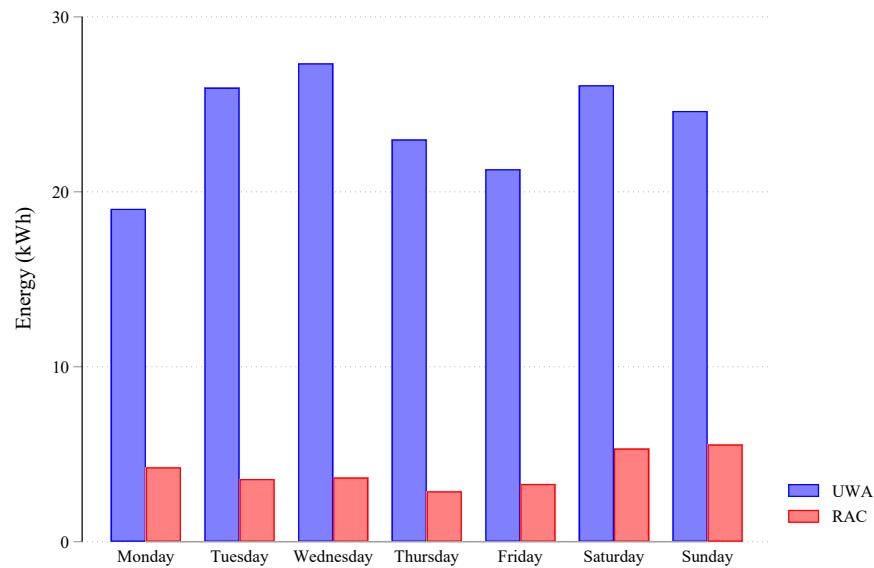


Fig. 10.18 The energy delivered per station per day of the week between the UWA (12/11/2014 to 13/10/2017) and the Electric Highway (RAC) (02/03/2016 to 20/09/2016) DC stations.

Fig. 10.18 compares the energy usage between the UWA station and the average Electric Highway station across each day of the week. The Highway stations are more popular during weekends, as more traffic commutes to regional destinations. On average the Highway stations consume 5.55 kWh on a Sunday as compared to 2.88 kWh on a Thursday. The UWA charging station delivers the most energy on Wednesday with 27.3 kWh, and the least on Monday with 19 kWh.

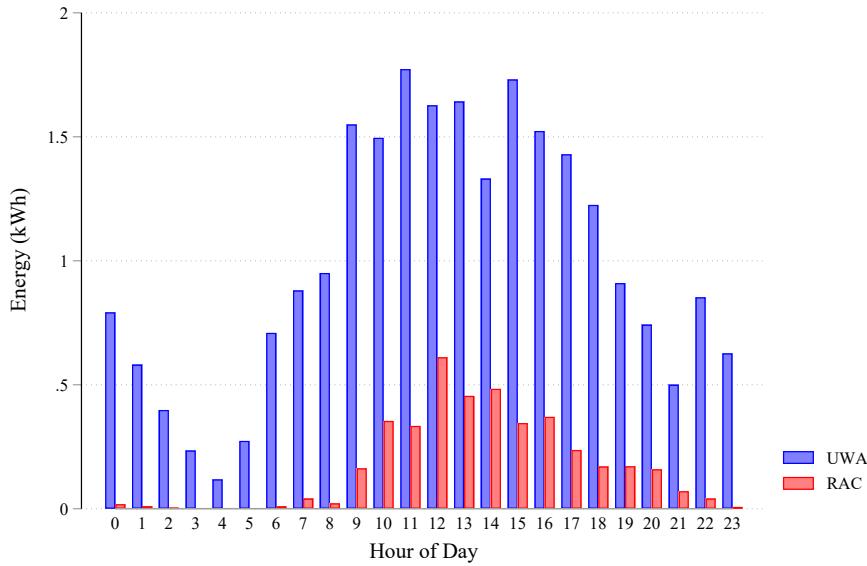


Fig. 10.19 The energy delivered per station per hour of day between the UWA (12/11/2014 to 13/10/2017) and the RAC (02/03/2016 to 20/09/2016) DC stations.

Fig. 10.19 compares the energy consumption per time of day between the UWA station and the average of the RAC charging stations. This data was averaged through all the historical charges on the UWA station, which was then classified to its instantaneous energy consumption at each hourly duration per day. This data is then compared with the data that was obtained from the RAC stations. On average, the UWA station delivers 23.9 kWh per day, while the average Highway station delivers 4.08 kWh per day.

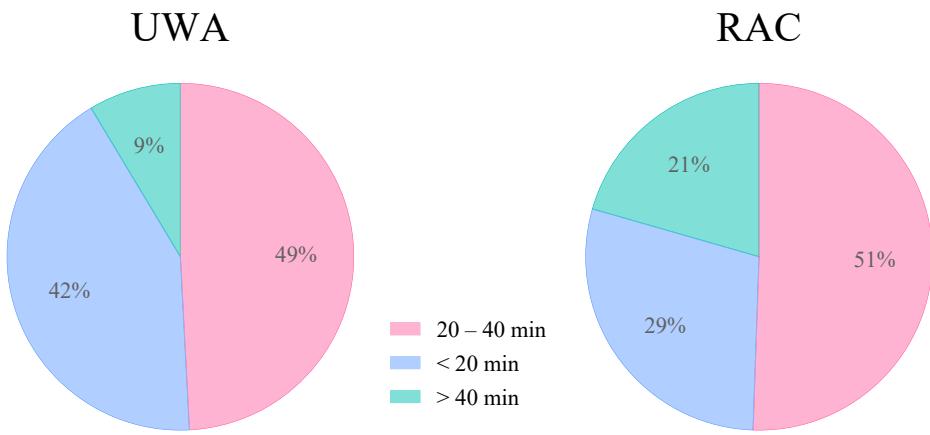


Fig. 10.20 The average charging durations on the UWA (12/11/2014 to 13/10/2017) and the Electric Highway (02/03/2016 to 20/09/2016) DC stations.

Charging durations at the UWA stations, as shown in Fig. 10.20, are predominantly under 40 minutes, which makes up 89% of all charges. The average charging time for the UWA DC station is 22.45 minutes. Half of the charges at the Electric Highway stations take between 20 to 40 minutes, with 29% taking less than 20 minutes. The average charging time for the Electric Highway DC stations is 30.68 minutes.

Table 10.4 Comparison of average charging duration and energy consumption for AC and DC stations (02/03/2016 to 20/09/2016).

Type	Owner	Duration (hh:mm)	Energy (kWh)
DC	UWA	00:21	7.128
	Highway	00:31	12.26
AC	UWA (7 kW)	05:11	9.811
	Highway (7 kW)	02:01	4.313
	Highway (43kW)	01:19	16.69

Table 10.4 summarises the average charging duration and energy consumption per charge on AC or DC charging stations of UWA and RAC. Comparing the DC charge times, users of an RAC DC station charge 10 minutes longer on average and delivered 4.6 kWh more energy than they do at the UWA station. This is mostly contributed by the West Perth station, which is more frequented by drivers due to its close proximity to the city centre, which implies that drivers can visit the nearby shopping centre and cafes while their vehicle is charging. Conversely, charging durations are longer at the UWA/REV AC stations (of which half are installed near workplaces) when compared to the RAC 7 kW AC stations, which average to about 1.5 hours longer and 2.83 kWh more energy delivered. The 43 kW fast-AC chargers average at 1.3 hours charge time, delivering 16.69 kWh of energy. The average charging time per vehicle on the UWA DC station is 21 minutes to take, on average, 7.1 kWh of energy. For the Highway stations, the average charging time is 31 minutes for 12.26 kWh of energy.

10.4.4 DC Charging Connectors Used

Fig. 10.21 compares the types of connectors used at the UWA DC station. CHAdeMO (88%) is in higher demand than CCS (12%) which is because popular EV models from Mitsubishi and Nissan use CHAdeMO, and Tesla provides a CHAdeMO adapter for their vehicles. This trend is set to change with the introduction of more EVs with CCS connectors in Australia from the 2018 model year onwards.

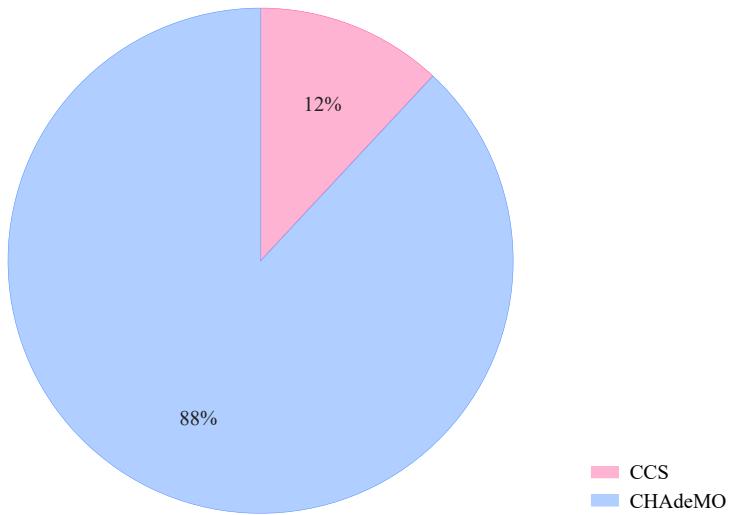


Fig. 10.21 Percentage of connector types used at the UWA DC station (12/11/2014 to 13/10/2017).

10.5 Cost Modelling

Table 10.5 introduces a cost model that includes the usage analysis as summarised in Section 10.4. This is presented as a probabilistic case study for running and maintaining various types of charging stations, namely 7 kW AC (AC-7), 50 kW DC (DC-50), 150 kW DC (DC-150) and 350 kW DC (DC-350).

The stations' running costs are calculated per day based on the costs associated to their estimated purchasing and installation costs, while assuming a financing option and depreciation of 5% and 8% per annum respectively over its lifespan. Energy tariffs are based on ongoing rates from Synergy, which is the sole residential energy provider in metropolitan WA. Based on observations, new stations are expected to be provisioned for ten years before needing replacements or large-scale maintenance. The total running cost includes estimated ongoing maintenance cost, and the option of parking bay rental. Calculations of the sales required to break even include scenarios where bay rental is needed or otherwise. *Actual energy* and charging time values are based on data collection from the UWA/REV stations. The *estimated* use subject illustrates conservative estimates for utilisation of more powerful DC stations under a higher EV adoption rate.

A station running cost C_r is calculated as the sum of its finance interest, depreciation and its operating/maintenance cost per day, adding its energy supply cost and if applicable, its

bay lease.

$$C_r = \frac{i + D + C_m}{365.25} + \frac{C_{sup}}{S} [+C_s] \quad (10.1)$$

Using estimates for i , D , and C_m in Table 10.5, along C_{sup} provided by Synergy, the running cost for the 7 kW AC, 50 kW DC, 150 kW DC and 350 kW DC stations was calculated to be \$2.11, \$13.81, \$28.99 and \$57.62 respectively, excluding an estimated bay lease of \$10 per day. These figures scale exponentially with the charging station's power output, as more powerful stations are more expensive and require more energy to operate. This is, however, compensated with faster charging durations, allowing a higher charge frequency.

To calculate the required break-even energy sales R for each charging station to break even, scenarios with profit margins M at 50% and 100%, with or without the bay lease of \$10/day (B/\bar{B}) were considered. The energy tariff T_E is referenced to Synergy, which at time of writing stands at \$0.28327/kWh.

$$R = \frac{C_r}{M \cdot T_E} \quad (10.2)$$

The calculated sales requirements R to break-even for these four scenarios across the four charging station types is then plotted as illustrated in Fig. 10.22.

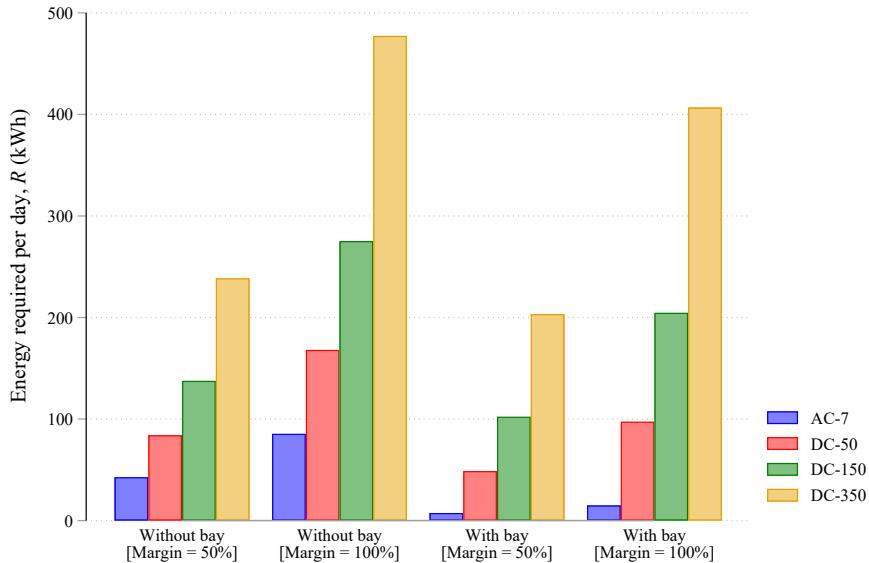


Fig. 10.22 Break-even points for the AC and DC stations' energy delivery in kWh required under scenarios representing with or without bay rentals C_B (Bay/No bay), with sales margins set at 50% ($M = 0.5$) and 100% ($M = 1$).

From Fig. 10.22, it is clear that any fee for the charging bay rental C_B increases the required break-even energy sales requirement R , but it has a lower relative effect on the higher-output DC stations, which are expected to sell more energy per day accordingly. For instance, the presence of the bay rental fee C_B across both margins increases the break-even point R by 573% on the 7 kW AC station, which means this station will never be profitable in this scenario.

For 50 kW, 150 kW and 350 kW DC stations, break-even point R increases to 172%, 134% and 117%, respectively. This results in less impact for faster stations. Increasing the sales margins from 50% to 100% halves the break-even point R across all stations and C_B scenarios.

The collected data in Sections 10.4.1 and 10.4.3 was subsequently utilised to measure the actual usage of the 7 kW AC and 50 kW DC stations, the energy delivery E_d is defined as the product of the number of users N and the average energy use per charge E_C .

$$E_d = N \cdot E_C \quad (10.3)$$

The energy cost C_E at that station is thus determined by the energy tariff T_E .

$$C_E = T_E \cdot E_d \quad (10.4)$$

By drawing a conservative estimate that anticipates a higher EV penetration density, a three to four-fold increase in users per day is expected across the 7 kW AC and 50 kW DC station, and more daily users for 150 kW and 350 kW DC stations once they are available.

Table 10.5 Cost model of the AC and DC stations according to their power throughout. The 350 kW DC station requires a dedicated transformer and substation, which is reflected in its installation cost. Running costs are estimated based on UWAs own 7 kW AC and 50 kW DC stations costs, and supplier quotes for the 150 kW and 350 kW DC stations.

Subject	Category	Unit	AC-7	DC-50	DC-150	DC-350
Running cost	Station cost, C_S	\$	3,000	30,000	70,000	127,000
	Installation cost, C_I	\$	1,000	6,000	8,000	30,000
	Expected lifespan, t_L	Years	10	10	10	10
	Interest at 5% (average), i	\$ / year	109.11	982.03	2,127.73	4,282.74
	Depreciation (constant), D	\$ / year	400	3,600	7,800	15,700
	Operating cost / maintenance, C_m	\$ / year	200	400	600	1,000
	Energy supply charge, C_{sup}	\$ / day	1.02	1.02	1.02	1.02
	Stations per site, S	Stations	6	6	6	6
	Supply charge per station, $C_{sup/S}$	\$ / day	0.17	0.17	0.17	0.17
Cost per day						
Total		\$ / day	2.11	13.81	28.99	57.62
Bay lease per day, C_B		\$ / day	10.00	10.00	10.00	10.00
Cost per day with bay		Total	\$ / day	12.11	23.81	38.99
Energy		Energy tariff, T_E	\$ / kWh	0.28327	0.28327	0.28327
Sales required to break even, R	Without bay [Margin = 50%]	kWh / day	14.90	97.50	204.70	406.80
	Without bay [Margin = 100%]	kWh / day	7.45	48.75	102.35	203.40
	With bay [Margin = 50%]	kWh / day	85.51	168.10	275.30	477.40
	With bay [Margin = 100%]	kWh / day	42.75	84.05	137.65	238.70

continues on next page

Table 10.5 *continued from previous page*

Subject	Category	Unit	AC-7	DC-50	DC-150	DC-350
Actual use	Actual user count, N	Users / day	0.43	3.35		
	Actual amount of energy per charge, E_C	kWh	9.12	7.13		
	Actual energy delivery at UWA, E_d	kWh / day	3.91	23.90		
	Actual Energy cost, C_E	\$ / day	1.11	6.77		
Estimated use for higher EV density (conservative estimate)	User count, N	Users / day	2	10	20	40
	Amount of energy per charge, E_C	kWh	7	15	20	30
	Energy delivery at UWA, E_d	kWh / day	14.00	150.00	400.00	1200.00
	Energy cost, C_E	\$ / day	3.97	42.49	113.31	339.93

10.6 Validation

A similar study was performed in Ireland, finishing in 2016 [428]. This study first investigates the EV charging landscape in Ireland, while drawing comparisons to other European countries. The authors noticed that the numerous EV adoption strategies and incentives undertaken by these countries are contributing to the large growth of EV sales, which introduces a demand for charging stations. The authors then analysed the usage of 711 charging stations, including 83 DC fast-chargers in Ireland and Northern Ireland through their recorded charge events. Comparisons were performed on aggregated standard and fast-DC charge point datasets, use cases for standard charge points, and use cases for fast-charge points. From these comparisons, the authors then deduced that slow AC chargers have more usage throughout the day, compared to fast chargers that see more usage through the evening and night, which is consistent with the findings presented in this paper. Additionally, the average charge duration for fast chargers is 36 minutes versus three hours for standard chargers, which is also comparable to this paper's findings. To the best of the authors' knowledge this work presents the only other analysis of charging station usages in a geographic location.

10.7 Conclusion

While it makes a significant difference, whether charging energy is provided free of charge or for a nominal fee, the location of the stations is also a fundamental factor. While originally proposed as an Electric Highway by UWA, the RAC in cooperation with the local councils decided to place charging stations in the local town centres instead of in proximity to the bypassing highway. The idea was probably that with the low number of EVs at this stage, the local communities should also benefit from this charging infrastructure. However, introducing power charges at about twice the rate of domestic fees made sure that locals will not use these chargers. Why would they use a charging station if they can charge for half the cost at their nearby home (or practically free if they have solar PV)?

As battery technology continues to evolve, EVs with larger batteries are coming onto the market. This means that public Level-1 and Level-2 AC charging infrastructure will become obsolete. The market is expected to shift such that AC charging is being used exclusively for home charging, while all public infrastructure will be DC charging.

The costs of the infrastructure, coupled with the consistently changing technology makes such an investment quite risky, considering the lifecycle and return on investment. Only where massive government incentives or investor capital are available do these projects become feasible. Even then, the infrastructure will only be utilised when the vehicle itself

does not have access to home charging. So, if one tries a comparison with the existing petrol station network, only about 10% of all charges are expected to need public infrastructure. Of course, this number highly depends on the local housing environment. The higher the percentage of people who live in houses with garages (as is the case in Western Australia), as opposed to apartments without any EV charging options, the lower the infrastructure requirement will be.

The major factors in EV adoption remain the initial purchase price (which is closely tied to \$/kWh battery prices) followed by the availability—or possibly just the perception of availability—of EV charging infrastructure. For modern EVs, range and charging times are almost on par with ICE vehicles, so these points should no longer play a role in purchase decisions.

Chapter 11

Conclusions

This chapter summarises the contributions of this thesis and presents possible outlines for future research directions. While this thesis explores problems across an interdisciplinary field related to intelligent vehicles, certain prevailing conclusions can be drawn in this respect.

11.1 Overall Findings

This thesis has resulted in the following contributions to the research in applied autonomous driving and electric vehicles.

In Chapter 4, a visual approach to multi-robot navigation was conceptualised. This method utilised an existing multi-robot system that was initially developed for distributed cooperative SLAM to solve localisation problems relating to wheel slip and obstacle detection, leading to the incorporation of visual odometry and semantic segmentation into the system. Evaluations have verified the feasibility of these algorithms in tangible outdoor environments, thereby motivating their implementations on frameworks for autonomous cars.

Chapters 5 through 8 have described subsequent implementations of these visual navigation approaches with an emphasis on autonomous driving. This began with semantic segmentation validations on Perth roads in Chapter 5, which resulted in robust classification accuracies and frame rates that are adequate for practical autonomous drives, which is further enhanced with LiDAR measurements. This was then integrated first as a module in a C++-based autonomous driving framework in Chapter 6, and then as part of an improved ROS-based framework in Chapter 7. In both instances, the visual navigation algorithms were supplemented with additional sensors such as LiDARs and IMUs to establish a holistic driving system. Semantic segmentation results were successfully used for the detection of road regions and lane markers, enabling the vehicle to achieve lane keeping and scene understanding. Visual odometry was implemented as described Chapter 7, which saw the

optimisation of an existing method to exploit the architecture of embedded computers, resulting in real-time accurate localisations. In addition to road lanes, visual object detection and recognition was utilised to construct path delimiters for drive tests. In this case, traffic cones were classified through an SVM and their positions are accurately ascertained by fusing LiDAR measurements, thereby producing an open path with the cones placed at either side. This method was verified first on the real system in Chapter 7, and then on a simulation platform in Chapter 8. For both applications, the cone detection algorithms were able to detect and position cones in real-time, enabling the vehicle to autonomously navigate the path.

Chapter 9 introduced the research on electric vehicles in this thesis through the presentation of a cloud-based telemetry platform, REView for data collection and analyses. This platform was programmed as a hybrid V2C/I2C solution for connected EVs, charging infrastructures and energy sources that are capable of aggregating data and interpreting it in a cohesive and meaningful way. Processed data are then visualised as a series of tables and charts with gamification features to critically inform EV users and station operators, in addition to providing automated billings to support network monetisation. Each feature on REView was modularly programmed to ensure scalability and improve interoperability for upcoming infrastructures and technologies. Usage forecasts for charging infrastructures have predicted the increase in popularity of fast chargers as EVs are shipped with larger battery capacities, whereas AC charging may be phasing into obsolescence with its lower charging frequencies. Results originating from REView have since facilitated significant headways into the investigation of Western Australia's EV landscape. Chapter 10 is an example of this contribution; data from REView was used to compare infrastructure usages between AC and DC charging. The study began with a comprehensive overview of the current EV charging outlook and proceeds with a consecution of time series analyses across AC and DC charging, with comparisons drawn against data from the RAC Electric Highway. Analysis results have demonstrated that charging behaviours differ across different charging types and that the location and usage cost of charging infrastructures directly affects its popularity. Following this, a cost model is also presented to illustrate the cost of charging infrastructure ownership and concluded that faster chargers will better benefit from higher usage traffic which will quickly offset the higher initial cost of investment.

11.2 Future Research Recommendations

Throughout the compilation of this thesis, there had been limitations that were identified, thereby disseminating further research questions. Notable recommendations for future works are described in the following paragraphs at a higher level.

Improved optimisations on visual navigation algorithms. Much of the algorithmic implementations described in this thesis were performed on embedded computers which provide limited computation headroom when compared to a workstation computer. While preliminary optimisations were performed for visual odometry, semantic segmentation was not subjected to this treatment which can yield higher frame rates. Segmentations can be optimised either by segregating ROIs (such as the road region for lane following routines) or by ignoring classes that are less relevant to a routine. Further, these methods can leverage on the rapid enhancements of deep learning libraries such as TensorFlow (including the upcoming TensorFlow 2.0) to easily design platform-optimised architectures. Contributing to this ease are also the availabilities of newer embedded computers that are purpose-built for deep learning applications from Nvidia. Its CUDA parallel computing platform and cuDNN deep learning library are well-supported for their compute hardware, capable of accelerating deep learning performances including TensorFlow and Caffe.

End-to-end multi-sensor driving system. The autonomous driving frameworks that were described in this thesis uses mediated perception that sees the decoupling of environmental perception and decision making. On the contrary, an end-to-end approach applies machine learning methods directly onto the control system with inputs such as steering, braking and acceleration. Using an end-to-end model that incorporates accurate localisation and scene understanding will introduce a cohesive deep learning paradigm for autonomous driving. Preliminary works to extend the software framework in Chapter 7 into an end-to-end solution is documented as a whitepaper in Appendix A. This approach utilises existing sensors and compute hardware along with the simulation system described in Chapter 8 to present a multi-sensor solution for autonomous road drives.

Electromobility penetration forecasts. The plethora of data collected throughout the years on REView presents myriad possible analytics beyond its usage forecasts. Using UWA’s DC charger as an example, the trends observed for charge frequencies, duration and energy consumption can be reinterpreted to predict Perth’s electric vehicle landscape, as it experiences a consistent usage frequency. For instance, models from charging frequencies and the number of EVs in Perth can be heterogeneously fused to improve prediction accuracies

of EV numbers. Likewise, data relating to charging energy delivery can be used to predict battery capacities of future EVs. Predictions like these can therefore be used as critical information such as policy whitepapers and consumer education.

Cloud-based big data analytics Throughout the preparation of works presented in this thesis, an understanding was established that much of the efforts, be it in autonomous driving or electromobility, are homogeneously converging towards a big data problem. Similar to the connected infrastructures, the autonomous driving software generates large amounts of complex real-time data from its interfaces whereby the use of big data analytics can greatly benefit and streamline any recursive learning routines. This can be further enhanced with cloud computing, taking advantage of high-speed, low-latency mobile connectivity to introduce a centralised, large-scale deployment of an edge-based multi-agent autonomous vehicle system over a V2V/V2C model. On the other hand, the nature of REView's current data structure can greatly benefit from a big data framework as data collection and visualisation occur in real-time. REView, when properly redesigned over a PaaS model for big data analytics, will thereby provide high-performance data acquisition with guaranteed scalability as a long-term solution to cater for the increasing number of EVs and charging infrastructures. When cloud-based autonomous driving is deployed alongside REView over a PaaS model, the foundation for a highly intelligent framework for smart cities can be established. This interconnectivity will enable an autonomous EV to access information relating to charging infrastructures such as its location and type, where it can further access information from the smart grid to schedule ideal charging patterns for its driving behaviours. Ultimately, this could thereby enable perpetual autonomous drives without user interference.

11.3 Final Remarks

This thesis has documented the application of visual navigation algorithms and the development of comprehensive software frameworks for autonomous driving and electric vehicles. Visual navigation algorithms were first verified for application feasibility, implemented first on mobile robots as a precursor, and then on purpose-built autonomous driving frameworks for real-world evaluations. Continuing on the trend of software framework developments is the presentation of an intelligent telemetry platform REView for electric vehicles and their infrastructures. Analytics stemming from REView have aided in the study of charging behaviours, usage forecasts and ownership costs for charging infrastructures around Western Australia.

The works described in this thesis originates within The REV Project and are testaments to its mission in developing intelligent and sustainable transportation solutions. We hope that the presentation of these readings is able to encourage future advancements towards this fast-paced research field.

References

- [1] E. D. Dickmanns and A. Zapp, “Autonomous High Speed Road Vehicle Guidance by Computer Vision1,” *IFAC Proceedings Volumes*, 10th Triennial IFAC Congress on Automatic Control - 1987 Volume IV, Munich, Germany, 27-31 July, vol. 20, no. 5, Part 4, pp. 221–226, Jul. 1987, ISSN: 1474-6670. DOI: 10.1016/S1474-6670(17)55320-3.
- [2] M. Buehler, K. Iagnemma, and S. Singh, *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*, 1st. Springer Publishing Company, Incorporated, 2009, ISBN: 3642039901, 9783642039904.
- [3] Grand View Research, “Self-driving Cars And Trucks Market Size | Industry Report, 2020-2030,” Grand View Research, San Francisco, Market Research 978-1-68038-884-8, Jun. 2018, p. 100.
- [4] R. Kumar and Richa, “Autonomous Vehicle Market by Level of Automation (Level 3, Level 4, and Level 5) and Component (Hardware, Software, and Service) and Application (Civil, Robo Taxi, Self-driving Bus, Ride Share, Self-driving Truck, and Ride Hail) - Global Opportunity Analysis and Industry Forecast, 2019-2026,” Allied Market Research, Pune, Market Research AU_184649, May 2018, p. 493.
- [5] HTF Market Intelligence, “Overview of Global Self driving Car Market review now and beyond,” HTF Market Intelligence, Pune, Market Research HTF1304659, Mar. 2019, p. 100.
- [6] Frost & Sullivan, “Global Autonomous Driving Market Outlook, 2018,” Frost & Sullivan, San Antonio, Market Research K24A-01-00-00-00, Mar. 2018.
- [7] S. M. Hubbard, “Synthesis of Automated Vehicle Legislation,” Purdue University, Tech. Rep. FHWA/IN/JTRP-2017/21, Oct. 2017. DOI: 10.5703/1288284316575.
- [8] National Conference of State Legislatures. (Apr. 2019). Autonomous Vehicles State Bill Tracking Database, [Online]. Available: <http://www.ncsl.org/research/transportation/autonomous-vehicles-legislative-database.aspx> (visited on 06/09/2019).
- [9] National Transport Commission. (Apr. 2019). Automated vehicles in Australia, [Online]. Available: <https://www.ntc.gov.au/roads/technology/automated-vehicles-in-australia/> (visited on 06/09/2019).
- [10] Department of Transport. (Dec. 2018). Automated vehicles, [Online]. Available: <https://www.transport.wa.gov.au/projects/automated-vehicles.asp> (visited on 06/09/2019).
- [11] S. Singh, “Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey,” National Center for Statistics and Analysis, Publication DOT HS 812 115, Feb. 2015, p. 2.

- [12] M. Bertoncello and D. Wee. (Jun. 2015). Ten ways autonomous driving could redefine the automotive world, [Online]. Available: <https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/ten-ways-autonomous-driving-could-redefine-the-automotive-world> (visited on 06/09/2019).
- [13] Department of Infrastructure, Transport, Cities and Regional Development. (May 2019). Safety Statistics, [Online]. Available: <https://www.bitre.gov.au/statistics/safety/> (visited on 06/09/2019).
- [14] MIT Technology Review Insights. (Mar. 2019). Autonomous driving: Safety first, [Online]. Available: <https://www.technologyreview.com/s/613087/autonomous-driving-safety-first/> (visited on 06/09/2019).
- [15] On-Road Automated Driving (ORAD) committee, “Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles,” SAE International, Report, Jun. 15, 2018, p. 35. DOI: 10.4271/J3016_201806.
- [16] J. Shuttleworth. (Jan. 2019). SAE J3016 automated-driving graphic, [Online]. Available: <https://www.sae.org/news/2019/01/sae-updates-j3016-automated-driving-graphic> (visited on 06/09/2019).
- [17] M. Burns. (Apr. 2019). ‘Anyone relying on lidar is doomed,’ Elon Musk says, [Online]. Available: <http://social.techcrunch.com/2019/04/22/anyone-relying-on-lidar-is-doomed-elon-musk-says/> (visited on 06/09/2019).
- [18] J. Rogelj, M. den Elzen, N. Höhne, T. Fransen, H. Fekete, H. Winkler, R. Schaeffer, F. Sha, K. Riahi, and M. Meinshausen, “Paris Agreement climate proposals need a boost to keep warming well below 2 °C,” *Nature*, vol. 534, no. 7609, pp. 631–639, Jun. 2016, ISSN: 1476-4687. DOI: 10.1038/nature18307.
- [19] United States Environmental Protection Agency, “Inventory of U.S. Greenhouse Gas Emissions and Sinks: 1990–2017,” United States Environmental Protection Agency, Washington, D.C., Report EPA 430-R-19-001, Apr. 11, 2019, p. 675.
- [20] G. Santos, “Road transport and CO₂ emissions: What are the challenges?” *Transport Policy*, vol. 59, pp. 71–74, Oct. 2017, ISSN: 0967-070X. DOI: 10.1016/j.tranpol.2017.06.007.
- [21] Department of the Environment and Energy, “Australia’s emissions projections 2018,” Australian Government, Canberra, Report CC227.1118, Dec. 2018, p. 46.
- [22] D. A. Kirsch, *The electric vehicle and the burden of history*. New Brunswick, N.J: Rutgers University Press, 2000, ISBN: 978-0-8135-2808-3 978-0-8135-2809-0.
- [23] B. C. Johnson, “Environmental products that drive organizational change: General motor’s electric vehicle (EV1),” *Corporate Environmental Strategy*, vol. 6, no. 2, pp. 140–150, Jan. 1999, ISSN: 1066-7938. DOI: 10.1016/S1066-7938(00)80024-X.
- [24] PlugShare. (2018). PlugShare - EV Charging Station Map, [Online]. Available: <https://www.plugin.com/> (visited on 02/02/2018).
- [25] Tesla. (2018). Supercharger, [Online]. Available: https://www.tesla.com/en_AU/supercharger (visited on 02/02/2018).
- [26] J. Joseph. (Jan. 2016). How much does the electricity cost for an electric car? [Online]. Available: <https://www.finder.com.au/electricity-cost-for-an-electric-car> (visited on 06/09/2019).

- [27] G. H. Broadbent, D. Drozdzewski, and G. Metternicht, “Electric vehicle adoption: An analysis of best practice and pitfalls for policy making from experiences of Europe and the US,” *Geography Compass*, vol. 12, no. 2, pp. 1–15, 2018, ISSN: 1749-8198. DOI: 10.1111/gec3.12358.
- [28] G. Bauer, “The impact of battery electric vehicles on vehicle purchase and driving behavior in Norway,” *Transportation Research Part D: Transport and Environment*, vol. 58, pp. 239–258, Jan. 2018, ISSN: 1361-9209. DOI: 10.1016/j.trd.2017.12.011.
- [29] G. H. Broadbent, G. Metternicht, and D. Drozdzewski, “An Analysis of Consumer Incentives in Support of Electric Vehicle Uptake: An Australian Case Study,” *World Electric Vehicle Journal*, vol. 10, no. 1, p. 11, Mar. 2019. DOI: 10.3390/wevj10010011.
- [30] Senate Select Committee on Electric Vehicles, “Report,” Parliament of Australia, Canberra, Report, Jan. 30, 2019, p. 197.
- [31] N. S. Pearre and H. Ribberink, “Review of research on V2x technologies, strategies, and operations,” *Renewable and Sustainable Energy Reviews*, vol. 105, pp. 61–70, May 2019, ISSN: 1364-0321. DOI: 10.1016/j.rser.2019.01.047.
- [32] A. Papathanassiou and A. Khoryaev, “Cellular V2x as the essential enabler of superior global connected transportation services,” *IEEE 5G Tech Focus*, vol. 1, no. 2, Jun. 2017.
- [33] R. Deng, B. Di, and L. Song, “Cooperative Collision Avoidance for Overtaking Maneuvers in Cellular V2x-Based Autonomous Driving,” *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 4434–4446, May 2019, ISSN: 0018-9545. DOI: 10.1109/TVT.2019.2906509.
- [34] D. A. Hensher, “Tackling road congestion – What might it look like in the future under a collaborative and connected mobility model?” *Transport Policy*, vol. 66, A1–A8, Aug. 2018, ISSN: 0967-070X. DOI: 10.1016/j.tranpol.2018.02.007.
- [35] H. A. Khattak, H. Farman, B. Jan, and I. U. Din, “Toward Integrating Vehicular Clouds with IoT for Smart City Services,” *IEEE Network*, vol. 33, no. 2, pp. 65–71, Mar. 2019, ISSN: 0890-8044. DOI: 10.1109/MNET.2019.1800236.
- [36] K. Weeratunga and A. Somers, “Connected Vehicles: Are we ready?” Main Roads Western Australia, East Perth, Report, Jun. 2015, p. 64.
- [37] Transport for NSW, “Connected and Automated Vehicles Plan,” Transport for NSW, Chippendale, Report, Oct. 2018, p. 56.
- [38] Queensland Government. (Aug. 2017). CAVI: Cooperative and Automated Vehicle Initiative, [Online]. Available: <https://www.qld.gov.au/transport/projects/cavi> (visited on 06/09/2019).
- [39] S. Yenikaya, G. Yenikaya, and E. Düven, “Keeping the vehicle on the road: A survey on on-road lane detection systems,” *ACM Computing Surveys*, vol. 46, no. 1, pp. 1–43, Oct. 2013, ISSN: 0360-0300. DOI: 10.1145/2522968.2522970.
- [40] D. G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004, ISSN: 1573-1405. DOI: 10.1023/b:visi.0000029664.99615.94.

- [41] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, “Speeded-Up Robust Features (SURF),” *Computer Vision and Image Understanding*, vol. 110, pp. 346–359, 2008, ISSN: 1077-3142. DOI: 10.1016/j.cviu.2007.09.014.
- [42] A. Bar Hillel, R. Lerner, D. Levi, and G. Raz, “Recent progress in road and lane detection: A survey,” *Machine Vision and Applications*, vol. 25, pp. 727–745, 2014, ISSN: 1432-1769. DOI: 10.1007/s00138-011-0404-2.
- [43] C.-A. Brust, S. Sickert, M. Simon, E. Rodner, and J. Denzler, “Convolutional Patch Networks with Spatial Prior for Road Detection and Urban Scene Understanding,” *CoRR*, vol. abs/1502.06344, 2015.
- [44] P. Y. Shinzato, V. Grassi, F. S. Osorio, and D. F. Wolf, “Fast visual road recognition and horizon detection using multiple artificial neural networks,” in *2012 IEEE Intelligent Vehicles Symposium*, IEEE, Jun. 2012, pp. 1090–1095. DOI: 10.1109/IVS.2012.6232175.
- [45] N. Abbas and V. Mahdi, “A novel neural network based voting approach for road detection via image entropy and color filtering,” *Indian Journal of Science and Technology*, vol. 9, no. 7, 2016, ISSN: 0974 -5645.
- [46] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The KITTI dataset,” *The International Journal of Robotics Research*, vol. 32, pp. 1231–1237, Sep. 2013. DOI: 10.1177/0278364913491297.
- [47] T. Scharwächter, M. Enzweiler, U. Franke, and S. Roth, “Efficient Multi-cue Scene Segmentation,” in *Pattern Recognition: 35th German Conference, GCPR 2013, Saarbrücken, Germany, September 3-6, 2013. Proceedings*, J. Weickert, M. Hein, and B. Schiele, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 435–445, ISBN: 978-3-642-40602-7.
- [48] F. Janda, S. Pangerl, E. Lang, and E. Fuchs, “Road boundary detection for run-off road prevention based on the fusion of video and radar,” in *2013 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, Jun. 2013, pp. 1173–1178, ISBN: 1931-0587. DOI: 10.1109/IVS.2013.6629625.
- [49] P. D. Cristóforis, M. A. Nitsche, T. Krajník, and M. Mejail, “Real-time monocular image-based path detection,” *Journal of Real-Time Image Processing*, vol. 11, pp. 335–348, 2016, ISSN: 1861-8219. DOI: 10.1007/s11554-013-0356-z.
- [50] NVIDIA Corporation. (2017). NVIDIA DRIVE auto-pilot and cockpit computers, [Online]. Available: <http://www.nvidia.com/object/drive-automotive-technology.html>.
- [51] Mobileye. (May 2018). Mobileye, [Online]. Available: <https://www.mobileye.com/>.
- [52] T. Ahmad, G. Bebis, M. Nicolescu, A. Nefian, and T. Fong, “Fusion of edge-less and edge-based approaches for horizon line detection,” in *2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA)*, IEEE, Jul. 2015, pp. 1–6. DOI: 10.1109/IISA.2015.7387988.
- [53] W.-N. Lie, T. C. I. Lin, T.-C. Lin, and K.-S. Hung, “A robust dynamic programming algorithm to extract skyline in images for navigation,” *Pattern Recognition Letters*, vol. 26, pp. 221–230, 2005, ISSN: 0167-8655. DOI: 10.1016/j.patrec.2004.08.021.

- [54] T. Ahmad, G. Bebis, M. Nicolescu, A. Nefian, and T. Fong, “An edge-less approach to horizon line detection,” in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, IEEE, Dec. 2015, pp. 1095–1102. DOI: 10.1109/ICMLA.2015.67.
- [55] R. Verbickas and A. Whitehead, “Sky and Ground Detection Using Convolutional Neural Networks,” in *International Conference on Machine Vision and Machine Learning (MVML)*, 2014.
- [56] C. Rother, “A new approach to vanishing point detection in architectural environments,” *Image and Vision Computing*, vol. 20, pp. 647–655, 2002, ISSN: 0262-8856. DOI: 10.1016/S0262-8856(02)00054-9.
- [57] J.-C. Bazin and M. Pollefeys, “3-line RANSAC for orthogonal vanishing point detection,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, Oct. 2012, pp. 4282–4287, ISBN: 2153-0858. DOI: 10.1109/IROS.2012.6385802.
- [58] H. Kong, J. Y. Audibert, and J. Ponce, “General Road Detection From a Single Image,” *IEEE Transactions on Image Processing*, vol. 19, pp. 2211–2220, 2010, ISSN: 1057-7149. DOI: 10.1109/TIP.2010.2045715.
- [59] M. Zhu, Y. Liu, Y. Zhuang, and H. Hu, “Visual Campus Road Detection for an UGV using Fast Scene Segmentation and Rapid Vanishing Point Estimation,” *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 11 898–11 903, 2014, ISSN: 1474-6670. DOI: 10.3182/20140824-6-ZA-1003.00635.
- [60] Z. Wu, W. Fu, R. Xue, and W. Wang, “A Novel Line Space Voting Method for Vanishing-Point Detection of General Road Images,” *Sensors (Basel)*, vol. 16, no. 7, 2016, ISSN: 1424-8220 (Electronic) 1424-8220 (Linking). DOI: 10.3390/s16070948.
- [61] J. M. Álvarez, A. M. López, T. Gevers, and F. Lumbreras, “Combining Priors, Appearance, and Context for Road Detection,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 3, pp. 1168–1178, 2014, ISSN: 1524-9050. DOI: 10.1109/TITS.2013.2295427.
- [62] B. Ricaud, B. Stanciulescu, and A. Breheret, “General road detection algorithm,” in *Proceedings of the 3rd International Conference on Pattern Recognition Applications and Methods*, ser. ICPRAM 2014, ESEO, Angers, Loire Valley, France: SCITEPRESS - Science and Technology Publications, Lda, 2014, pp. 825–830, ISBN: 978-989-758-018-5. DOI: 10.5220/0004935208250830.
- [63] T. H. Bui and E. Nobuyama, “A local soft voting method for texture-based vanishing point detection from unstructured road images,” in *2012 Proceedings of SICE Annual Conference (SICE)*, Aug. 2012, pp. 396–401.
- [64] P. Moghadam, J. A. Starzyk, and W. S. Wijesoma, “Fast Vanishing-Point Detection in Unstructured Environments,” *IEEE Transactions on Image Processing*, vol. 21, pp. 425–430, 2012, ISSN: 1057-7149. DOI: 10.1109/TIP.2011.2162422.
- [65] K. Ghazali, R. Xiao, and J. Ma, “Road Lane Detection Using H-Maxima and Improved Hough Transform,” in *2012 Fourth International Conference on Computational Intelligence, Modelling and Simulation*, IEEE, Sep. 2012, pp. 205–208, ISBN: 2166-8523. DOI: 10.1109/CIMSim.2012.31.

- [66] T.-Y. Chen, C.-H. Chen, G.-M. Luo, W.-C. Hu, and J.-C. Chern, “Vehicle Detection in Nighttime Environment by Locating Road Lane and Taillights,” in *2015 International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP)*, IEEE, Sep. 2015, pp. 60–63. DOI: 10.1109/IIH-MSP.2015.82.
- [67] V. S. Bottazzi, P. V. K. Borges, B. Stantic, and J. Jo, “Adaptive Regions of Interest Based on HSV Histograms for Lane Marks Detection,” in *Robot Intelligence Technology and Applications 2: Results from the 2nd International Conference on Robot Intelligence Technology and Applications*, J.-H. Kim, E. T. Matson, H. Myung, P. Xu, and F. Karray, Eds., Cham: Springer International Publishing, 2014, pp. 677–687, ISBN: 978-3-319-05582-4.
- [68] D. Ding, C. Lee, and K.-Y. Lee, “An adaptive road ROI determination algorithm for lane detection,” in *2013 IEEE International Conference of IEEE Region 10 (TENCON 2013)*, IEEE, Oct. 2013, pp. 1–4, ISBN: 2159-3442. DOI: 10.1109/TENCON.2013.6718807.
- [69] J. C. McCall and M. M. Trivedi, “Video-based lane estimation and tracking for driver assistance: Survey, system, and evaluation,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, pp. 20–37, 2006, ISSN: 1524-9050. DOI: 10.1109/TITS.2006.869595.
- [70] S. Fernando, L. Udawatta, B. Horan, and P. Pathir, “Real-time Lane Detection on Suburban Streets using Visual Cue Integration,” *International Journal of Advanced Robotic Systems*, p. 1, 2014, ISSN: 1729-8806. DOI: 10.5772/58248.
- [71] J. Huang, H. Liang, Z. Wang, T. Mei, and Y. Song, “Robust lane marking detection under different road conditions,” in *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, IEEE, Dec. 2013, pp. 1753–1758. DOI: 10.1109/ROBIO.2013.6739721.
- [72] F. Oniga and S. Nedevschi, “Curb detection for driving assistance systems: A cubic spline-based approach,” in *2011 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, Jun. 2011, pp. 945–950, ISBN: 1931-0587. DOI: 10.1109/IVS.2011.5940580.
- [73] T. Scharwächter, M. Schuler, and U. Franke, “Visual guard rail detection for advanced highway assistance systems,” in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, IEEE, Jun. 2014, pp. 900–905, ISBN: 1931-0587. DOI: 10.1109/IVS.2014.6856573.
- [74] J. Dai, Y. Fang, T. Wu, D. Zhao, and H. He, “Night-Time Road Boundary Detection with Infrared Channel Features Classifier,” in *2014 IEEE International Conference on Computer and Information Technology*, IEEE, Sep. 2014, pp. 751–755. DOI: 10.1109/CIT.2014.35.
- [75] K. Shibata, K. Takeuch, S. Kawai, and Y. Horita, “Detection of road surface conditions in winter using road surveillance cameras at daytime, night-time and twilight,” *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 14, p. 21, 2014, ISSN: 1738-7906.
- [76] T. Hu and T. Wu, “Roadside curb detection based on fusing stereo vision and mono vision,” in *Fourth International Conference on Machine Vision (ICMV 2011): Computer Vision and Image Analysis; Pattern Recognition and Basic Technologies*, International Society for Optics and Photonics, vol. 8350, 2012, 83501H.

- [77] C. Fernández, D. F. Llorca, C. Stiller, and M. A. Sotelo, “Curvature-based curb detection method in urban environments using stereo and laser,” in *2015 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, Jun. 2015, pp. 579–584, ISBN: 978-1-4799-8587. DOI: 10.1109/IVS.2015.7225747.
- [78] P. Paalanen, J.-K. Kamarainen, J. Ilonen, and H. Kälviäinen, “Feature representation and discrimination based on Gaussian mixture model probability densities—Practices and algorithms,” *Pattern Recognition*, vol. 39, pp. 1346–1358, Jul. 2006, ISSN: 0031-3203. DOI: 10.1016/j.patcog.2006.01.005.
- [79] M. Aly, “Real time detection of lane markers in urban streets,” in *2008 IEEE Intelligent Vehicles Symposium*, IEEE, Jun. 2008, pp. 7–12, ISBN: 978-1-4799-0587. DOI: 10.1109/IVS.2008.4621152.
- [80] Q. Wang, J. Fang, and Y. Yuan, “Adaptive road detection via context-aware label transfer,” *Neurocomputing*, vol. 158, pp. 174–183, 2015, ISSN: 0925-2312. DOI: 10.1016/j.neucom.2015.01.054.
- [81] Y. Alkhorshid, K. Aryafar, S. Bauer, and G. Wanielik, “Road Detection through Supervised Classification,” *CoRR*, vol. abs/1605.03150, 2016.
- [82] Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997, ISSN: 0022-0000. DOI: 10.1006/jcss.1997.1504.
- [83] L. Xiao, B. Dai, D. Liu, D. Zhao, and T. Wu, “Monocular road detection using structured random forest,” *International Journal of Advanced Robotic Systems*, vol. 13, no. 3, p. 101, 2016. DOI: 10.5772/63561. eprint: <https://doi.org/10.5772/63561>.
- [84] S. Strygulec, D. Müller, M. Meuter, C. Nunn, S. Ghosh, and C. Wöhler, “Road boundary detection and tracking using monochrome camera images,” in *Proceedings of the 16th International Conference on Information Fusion*, Jul. 2013, pp. 864–870.
- [85] M. Wang, L. Jiang, W. Lu, and A. Fang, “Component-model based detection and recognition of road vehicles,” in *2015 IEEE International Conference on Progress in Informatics and Computing (PIC)*, IEEE, Dec. 2015, pp. 449–453. DOI: 10.1109/PIC.2015.7489887.
- [86] C. Siagian, C.-K. Chang, and L. Itti, “Mobile robot navigation system in outdoor pedestrian environment using vision-based road recognition,” in *2013 IEEE International Conference on Robotics and Automation*, IEEE, May 2013, pp. 564–571, ISBN: 978-1-4799-0472-9. DOI: 10.1109/ICRA.2013.6630630.
- [87] R. Chen, H. Xiao, X. Dou, and W. Hou, “Research on Recognition Methods of Bus Front Road Condition Based on Video,” in *2013 Seventh International Conference on Image and Graphics*, IEEE, Jul. 2013, pp. 439–442. DOI: 10.1109/ICIG.2013.94.
- [88] Y. Wang, D. Shen, and E. K. Teoh, “Lane detection using spline model,” *Pattern Recognition Letters*, vol. 21, pp. 677–689, 2000, ISSN: 0167-8655. DOI: 10.1016/S0167-8655(00)00021-0.
- [89] Z. Tao, P. Bonnifait, V. Frémont, and J. Ibañez-Guzman, “Lane marking aided vehicle localization,” in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, IEEE, Oct. 2013, pp. 1509–1515, ISBN: 978-1-4799-2153-0. DOI: 10.1109/ITSC.2013.6728444.

- [90] M. Revilloud, D. Gruyer, and E. Pollard, “An improved approach for robust road marking detection and tracking applied to multi-lane estimation,” in *2013 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, Jun. 2013, pp. 783–790, ISBN: 1931-0587. DOI: 10.1109/IVS.2013.6629562.
- [91] R. O. Duda and P. E. Hart, “Use of the Hough transformation to detect lines and curves in pictures,” *Communications of the ACM*, vol. 15, pp. 11–15, 1972, ISSN: 0001-0782. DOI: 10.1145/361237.361242.
- [92] A. Muhammad, *OpenCV Android Programming By Example*. Packt Publishing Ltd, 2015, ISBN: 1-78528-293-X.
- [93] S. Zhou and K. Iagnemma, “Self-supervised learning method for unstructured road detection using Fuzzy Support Vector Machines,” in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, Oct. 2010, pp. 1183–1189, ISBN: 2153-0858. DOI: 10.1109/IROS.2010.5650300.
- [94] J. Wang, Z. Ji, and Y.-T. Su, “Unstructured road detection using hybrid features,” in *2009 International Conference on Machine Learning and Cybernetics*, vol. 1, IEEE, Jul. 2009, pp. 482–486, ISBN: 2160-133X. DOI: 10.1109/ICMLC.2009.5212506.
- [95] M. Foedisch and A. Takeuchi, “Adaptive real-time road detection using neural networks,” in *Proceedings. The 7th International IEEE Conference on Intelligent Transportation Systems (IEEE Cat. No.04TH8749)*, IEEE, Oct. 2004, pp. 167–172. DOI: 10.1109/ITSC.2004.1398891.
- [96] A. Cord and S. Chambon, “Automatic Road Defect Detection by Textural Pattern Recognition Based on AdaBoost,” *Computer-Aided Civil and Infrastructure Engineering*, vol. 27, pp. 244–259, 2012, ISSN: 1467-8667. DOI: 10.1111/j.1467-8667.2011.00736.x.
- [97] P. Conrad and M. Foedisch, “Performance evaluation of color based road detection using neural nets and support vector machines,” in *32nd Applied Imagery Pattern Recognition Workshop, 2003. Proceedings.*, IEEE, Oct. 2003, pp. 157–160. DOI: 10.1109/AIPR.2003.1284265.
- [98] J. Yao, S. Ramalingam, Y. Taguchi, Y. Miki, and R. Urtasun, “Estimating Drivable Collision-Free Space from Monocular Video,” in *2015 IEEE Winter Conference on Applications of Computer Vision*, IEEE, Jan. 2015, pp. 420–427, ISBN: 1550-5790. DOI: 10.1109/WACV.2015.62.
- [99] J. Fritsch, T. Kühnl, and A. Geiger, “A new performance measure and evaluation benchmark for road detection algorithms,” in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, IEEE, Oct. 2013, pp. 1693–1700, ISBN: 2153-0009. DOI: 10.1109/ITSC.2013.6728473.
- [100] D. A. Chakra and J. Zelek, “Road Segmentation in Street View Images Using Texture Information,” in *2016 13th Conference on Computer and Robot Vision (CRV)*, IEEE, Jun. 2016, pp. 424–431. DOI: 10.1109/CRV.2016.47.
- [101] G. L. Oliveira, W. Burgard, and T. Brox, “Efficient deep models for monocular road segmentation,” in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Oct. 2016, pp. 4885–4891. DOI: 10.1109/IROS.2016.7759717.

- [102] C. C. T. Mendes, V. Frémont, and D. F. Wolf, “Exploiting fully convolutional neural networks for fast road detection,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 3174–3179. DOI: 10.1109/ICRA.2016.7487486.
- [103] L. Caltagirone, S. Scheidegger, L. Svensson, and M. Wahde, “Fast lidar-based road detection using fully convolutional neural networks,” in *2017 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, Jun. 2017, pp. 1019–1024. DOI: 10.1109/IVS.2017.7995848.
- [104] D. Kochanov, A. Osep, J. Stuckler, and B. Leibe, “Scene flow propagation for semantic mapping and object discovery in dynamic street scenes,” in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Oct. 2016, pp. 1785–1792. DOI: 10.1109/IROS.2016.7759285.
- [105] M. Teichmann, M. Weber, M. Zollner, R. Cipolla, and R. Urtasun, “Multinet: Real-time joint semantic reasoning for autonomous driving,” in *2018 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, Jun. 2018, pp. 1013–1020. DOI: 10.1109/IVS.2018.8500504.
- [106] V. Badrinarayanan, A. Kendall, and R. Cipolla, “SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2016.2644615.
- [107] M. Thoma, “A survey of semantic segmentation,” *arXiv preprint arXiv:1602.06541*, 2016.
- [108] A. Garcia-Garcia, S. Orts-Escalano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez, “A Review on Deep Learning Techniques Applied to Semantic Segmentation,” *arXiv preprint arXiv:1704.06857*, 2017.
- [109] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? The KITTI vision benchmark suite,” in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2012, pp. 3354–3361. DOI: 10.1109/CVPR.2012.6248074.
- [110] G. J. Brostow, J. Fauqueur, and R. Cipolla, “Semantic object classes in video: A high-definition ground truth database,” *Pattern Recognition Letters*, vol. 30, no. 2, pp. 88–97, 2009, ISSN: 0167-8655. DOI: 10.1016/j.patrec.2008.04.005.
- [111] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The cityscapes dataset for semantic urban scene understanding,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.
- [112] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, “Enet: A deep neural network architecture for real-time semantic segmentation,” *arXiv preprint arXiv:1606.02147*, 2016.
- [113] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, “Caffe: Convolutional architecture for fast feature embedding,” in *Proceedings of the 22nd ACM International Conference on Multimedia*, ser. MM ’14, Orlando, Florida, USA: ACM, 2014, pp. 675–678, ISBN: 978-1-4503-3063-3. DOI: 10.1145/2647868.2654889.
- [114] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

- [115] M. D. Zeiler and R. Fergus, “Visualizing and Understanding Convolutional Networks,” in *Computer Vision – ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part I*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., Cham: Springer International Publishing, 2014, pp. 818–833, ISBN: 978-3-319-10590-1. DOI: 10.1007/978-3-319-10590-1_53.
- [116] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.
- [117] NVIDIA Corporation. (2017). Embedded Systems, [Online]. Available: <http://www.nvidia.com/object/embedded-systems-dev-kits-modules.html>.
- [118] S. Song, S. P. Lichtenberg, and J. Xiao, “SUN RGB-D: A RGB-D scene understanding benchmark suite,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2015, pp. 567–576, ISBN: 1063-6919. DOI: 10.1109/CVPR.2015.7298655.
- [119] M. Treml, J. Arjona-Medina, T. Unterthiner, R. Durgesh, F. Friedmann, P. Schuberth, A. Mayr, M. Heusel, M. Hofmarcher, M. Widrich, B. Nessler, and S. Hochreiter, “Speeding up semantic segmentation for autonomous driving,” in *MLITS, NIPS Workshop*, Dec. 2016.
- [120] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and\leq0.5 MB model size,” *arXiv preprint arXiv:1602.07360*, 2016.
- [121] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, Apr. 2018, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2017.2699184.
- [122] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, “Tensorflow: A system for large-scale machine learning,” in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, Savannah, GA: USENIX Association, 2016, pp. 265–283, ISBN: 978-1-931971-33-1.
- [123] A. Palazzi. (2017). GitHub - ndrplz/dilation-tensorflow, [Online]. Available: <https://github.com/ndrplz/dilation-tensorflow>.
- [124] Velodyne LiDAR. (2017). Velodyne LiDAR, [Online]. Available: <http://velodynelidar.com/>.
- [125] FLIR® Systems. (2017). FLIR Systems - The World’s Sixth Sense, [Online]. Available: <http://www.flir.com>.
- [126] Mobileye. (2017). The Evolution of EyeQ, [Online]. Available: <https://www.mobileye.com/our-technology/evolution-eyeq-chip/>.
- [127] SAE International. (2016). U.S. Department of Transportation’s New Policy on Automated Vehicles Adopts SAE International’s Levels of Automation for Defining Driving Automation in On-Road Motor Vehicles, [Online]. Available: <https://www.sae.org/news/3544/>.

- [128] D. Shapiro. (Sep. 28, 2016). Introducing Xavier, the NVIDIA AI Supercomputer for the Future of Autonomous Transportation, [Online]. Available: <https://blogs.nvidia.com/blog/2016/09/28/xavier/>.
- [129] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, and J. Zhang, “End to end learning for self-driving cars,” *arXiv preprint arXiv:1604.07316*, 2016.
- [130] Waymo. (2017). Waymo, [Online]. Available: <https://waymo.com/>.
- [131] D. Etherington. (Dec. 14, 2016). Uber’s self-driving cars start picking up passengers in San Francisco, [Online]. Available: <https://techcrunch.com/2016/12/14/ubers-self-driving-cars-start-picking-up-passengers-in-san-francisco/>.
- [132] G. Nica. (Aug. 2, 2016). BMW CEO Wants Autonomous Driving Cars within Five Years, [Online]. Available: <https://www.bmwblog.com/2016/08/02/bmw-ceo-wants-autonomous-driving-cars-within-five-years/>.
- [133] Volvo Car Corporation. (2017). Autonomous Driving | Intellisafe | Volvo Cars, [Online]. Available: <http://www.volvocars.com/intl/about/our-innovation-brands/intellisafe/autonomous-driving>.
- [134] Daimler AG. (2017). Autonomous Driving - Mobility of the future, [Online]. Available: <https://www.daimler.com/innovation/autonomous-driving/>.
- [135] Udacity. (2017). Self-Driving Car Engineer Nanodegree, [Online]. Available: <https://www.udacity.com/course/self-driving-car-engineer-nanodegree--nd013>.
- [136] T. Anthony. (2017). GitHub - thomasantony/CarND-P04-Advanced-Lane-Lines, [Online]. Available: <https://github.com/thomasantony/CarND-P04-Advanced-Lane-Lines>.
- [137] comma.ai. (2017). Comma.ai, [Online]. Available: <https://comma.ai/>.
- [138] ——, (2017). Openpilot, [Online]. Available: <https://openpilot.comma.ai>.
- [139] ——, (2016). Self coloring books, [Online]. Available: <https://commaai.blogspot.com/2016/07/self-coloring-books.html>.
- [140] F. Visin, A. Romero, K. Cho, M. Matteucci, M. Ciccone, K. Kastner, Y. Bengio, and A. Courville, “ReSeg: A Recurrent Neural Network-Based Model for Semantic Segmentation,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2016, pp. 426–433. DOI: 10.1109/CVPRW.2016.60.
- [141] E. Santana and G. Hotz, “Learning a driving simulator,” *arXiv preprint arXiv:1608.01230*, 2016.
- [142] K. Öfjäll, M. Felsberg, and A. Robinson, “Visual autonomous road following by symbiotic online learning,” in *2016 IEEE Intelligent Vehicles Symposium (IV)*, 2016, pp. 136–143. DOI: 10.1109/IVS.2016.7535377.
- [143] K. Öfjäll and M. Felsberg, “Biologically inspired online learning of visual autonomous driving,” in *Proceedings of the British Machine Vision Conference 2014*, BMVA Press, 2014. DOI: 10.5244/C.28.94.
- [144] T. Krajiník, J. Blažíček, and J. M. Santos, “Visual road following using intrinsic images,” in *2015 European Conference on Mobile Robots (ECMR)*, IEEE, Sep. 2015, pp. 1–6. DOI: 10.1109/ECMR.2015.7324212.

- [145] C. E. Shannon, “A mathematical theory of communication,” *The Bell System Technical Journal*, vol. 27, pp. 379–423, 1948, ISSN: 0005-8580. DOI: 10.1002/j.1538-7305.1948.tb01338.x.
- [146] H. P. Moravec, “Obstacle avoidance and navigation in the real world by a seeing robot rover,” PhD thesis, Stanford University - Computer Science Department, 1980.
- [147] D. Nistér, O. Naroditsky, and J. Bergen, “Visual odometry for ground vehicle applications,” *Journal of Field Robotics*, vol. 23, no. 1, pp. 3–20, Jan. 2006, ISSN: 1556-4959, 1556-4967. DOI: 10.1002/rob.20103.
- [148] D. Scaramuzza and F. Fraundorfer, “Visual Odometry [Tutorial],” *IEEE Robotics & Automation Magazine*, vol. 18, no. 4, pp. 80–92, 2011, ISSN: 1070-9932. DOI: 10.1109/MRA.2011.943233.
- [149] K. Yousif, A. Bab-Hadiashar, and R. Hoseinnezhad, “An Overview to Visual Odometry and Visual SLAM: Applications to Mobile Robotics,” *Intelligent Industrial Systems*, vol. 1, no. 4, pp. 289–311, 2015, ISSN: 2199-854X. DOI: 10.1007/s40903-015-0032-7.
- [150] R. I. Hartley and P. Sturm, “Triangulation,” *Computer Vision and Image Understanding*, vol. 68, no. 2, pp. 146–157, Nov. 1997, ISSN: 1077-3142. DOI: 10.1006/cviu.1997.0547.
- [151] G. Klein and D. Murray, “Parallel Tracking and Mapping for Small AR Workspaces,” in *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, 1514363: IEEE Computer Society, Nov. 2007, pp. 1–10. DOI: 10.1109/ismar.2007.4538852.
- [152] D. Valiente García, L. Fernández Rojo, A. Gil Aparicio, L. Payá Castelló, and O. Reinoso García, “Visual Odometry through Appearance- and Feature-Based Method with Omnidirectional Images,” *Journal of Robotics*, pp. 1–13, 2012. DOI: 10.1155/2012/797063.
- [153] E. Rosten and T. Drummond, “Machine Learning for High-Speed Corner Detection,” in *Computer Vision – ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I*, A. Leonardis, H. Bischof, and A. Pinz, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 430–443, ISBN: 978-3-540-33833-8.
- [154] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, “BRIEF: Binary Robust Independent Elementary Features,” in *Computer Vision – ECCV 2010*, K. Daniilidis, P. Maragos, and N. Paragios, Eds., ser. Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2010, pp. 778–792, ISBN: 978-3-642-15561-1.
- [155] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “ORB: An efficient alternative to SIFT or SURF,” in *2011 International Conference on Computer Vision*, IEEE, Nov. 2011, pp. 2564–2571, ISBN: 1550-5499. DOI: 10.1109/ICCV.2011.6126544.
- [156] M. A. Fischler and R. C. Bolles, “Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, vol. 24, pp. 381–395, 1981, ISSN: 0001-0782. DOI: 10.1145/358669.358692.

- [157] M. Buczko and V. Willert, “Monocular Outlier Detection for Visual Odometry,” in *2017 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2017, pp. 739–745. DOI: 10.1109/IVS.2017.7995805.
- [158] ——, “How to distinguish inliers from outliers in visual odometry for high-speed automotive applications,” in *2016 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2016, pp. 478–483. DOI: 10.1109/IVS.2016.7535429.
- [159] B. D. Lucas and T. Kanade, “An iterative image registration technique with an application to stereo vision,” in *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, ser. IJCAI’81, Vancouver, BC, Canada: Morgan Kaufmann Publishers Inc., 1981, pp. 674–679.
- [160] B. K. Horn and B. G. Schunck, “Determining optical flow,” *Artificial Intelligence*, vol. 17, no. 1, pp. 185–203, Aug. 1981, ISSN: 0004-3702. DOI: 10.1016/0004-3702(81)90024-2.
- [161] G. Farnebäck, “Two-Frame Motion Estimation Based on Polynomial Expansion,” in *Image Analysis: 13th Scandinavian Conference, SCIA 2003 Halmstad, Sweden, June 29 – July 2, 2003 Proceedings*, J. Bigun and T. Gustavsson, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 363–370, ISBN: 978-3-540-45103-7.
- [162] M. Tao, J. Bai, P. Kohli, and S. Paris, “SimpleFlow: A Non-iterative, Sublinear Optical Flow Algorithm,” *Computer Graphics Forum*, vol. 31, pp. 345–353, 2012, ISSN: 0167-7055. DOI: 10.1111/j.1467-8659.2012.03013.x.
- [163] H. Wang and C. Schmid, “Action recognition with improved trajectories,” in *The IEEE International Conference on Computer Vision (ICCV)*, Dec. 2013.
- [164] H. Liu, C. Wang, J. Lu, Z. Tang, and J. Yang, “Maximum Likelihood Estimation of Monocular Optical Flow Field for Mobile Robot Ego-motion,” *International Journal of Advanced Robotic Systems*, p. 1, 2016, ISSN: 1729-8806. DOI: 10.5772/62157.
- [165] T. Kroeger, R. Timofte, D. Dai, and L. J. V. Gool, “Fast Optical Flow using Dense Inverse Search,” *CoRR*, vol. abs/1603.03590, 2016.
- [166] S. Baker and I. Matthews, “Equivalence and efficiency of image alignment algorithms,” in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, IEEE Comput. Soc, 2001, I–1090–I–1097 vol.1, ISBN: 1063-6919. DOI: 10.1109/cvpr.2001.990652.
- [167] N. Yang, R. Wang, X. Gao, and D. Cremers, “Challenges in Monocular Visual Odometry: Photometric Calibration, Motion Bias and Rolling Shutter Effect,” *arXiv:1705.04300 [cs]*, May 2017, arXiv: 1705.04300.
- [168] R. Mur-Artal and J. D. Tardós, “ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras,” *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017, ISSN: 1552-3098. DOI: 10.1109/TRO.2017.2705103.
- [169] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, “SVO: Semidirect Visual Odometry for Monocular and Multicamera Systems,” *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 249–265, Apr. 2017, ISSN: 1552-3098. DOI: 10.1109/TRO.2016.2623335.

- [170] J. Engel, V. Koltun, and D. Cremers, “Direct Sparse Odometry,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 611–625, Mar. 2018, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2017.2658577.
- [171] H. J. Chien, C. C. Chuang, C. Y. Chen, and R. Klette, “When to use what feature? SIFT, SURF, ORB, or A-KAZE features for monocular visual odometry,” in *2016 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, Nov. 2016, pp. 1–6. DOI: 10.1109/IVCNZ.2016.7804434.
- [172] P. Alcantarilla, J. Nuevo, and A. Bartoli, “Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces,” in *Proceedings of the British Machine Vision Conference 2013*, British Machine Vision Association, 2013, pp. 131–1311, ISBN: 978-1-901725-49-0. DOI: 10.5244/C.27.13.
- [173] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, “DTAM: Dense tracking and mapping in real-time,” in *2011 International Conference on Computer Vision*, Nov. 2011, pp. 2320–2327. DOI: 10.1109/ICCV.2011.6126513.
- [174] M. Meilland, A. Comport, and P. Rives, “Real-time Dense Visual Tracking under Large Lighting Variations,” in *Proceedings of the British Machine Vision Conference 2011*, British Machine Vision Association, 2011, pp. 451–4511, ISBN: 978-1-901725-43-8. DOI: 10.5244/C.25.45.
- [175] S. Daftry, D. Dey, H. Sandhawalia, S. Zeng, J. A. Bagnell, and M. Hebert, “Semi-Dense Visual Odometry for Monocular Navigation in Cluttered Environment,” *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2015.
- [176] J. Engel, T. Schöps, and D. Cremers, “LSD-SLAM: Large-Scale Direct Monocular SLAM,” in *Computer Vision – ECCV 2014*, ser. Lecture Notes in Computer Science, Springer, Cham, Sep. 2014, pp. 834–849, ISBN: 978-3-319-10604-5 978-3-319-10605-2. DOI: 10.1007/978-3-319-10605-2_54.
- [177] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, “iSAM2: Incremental smoothing and mapping using the Bayes tree,” *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 216–235, Feb. 2012, ISSN: 0278-3649. DOI: 10.1177/0278364911430419.
- [178] J. Engel, V. Usenko, and D. Cremers, “A Photometrically Calibrated Benchmark For Monocular Visual Odometry,” *arXiv:1607.02555 [cs]*, Jul. 2016, arXiv: 1607.02555.
- [179] A. D. Sappa, C. A. Aguilera, J. A. Carvajal Ayala, M. Oliveira, D. Romero, B. X. Vintimilla, and R. Toledo, “Monocular visual odometry: A cross-spectral image fusion based approach,” *Robotics and Autonomous Systems*, vol. 85, pp. 26–36, Nov. 2016, ISSN: 0921-8890. DOI: 10.1016/j.robot.2016.08.005.
- [180] B. Kitt, A. Geiger, and H. Lategahn, “Visual odometry based on stereo image sequences with RANSAC-based outlier rejection scheme,” in *2010 IEEE Intelligent Vehicles Symposium*, Jun. 2010, pp. 486–492. DOI: 10.1109/IVS.2010.5548123.
- [181] B. Lee, K. Daniilidis, and D. D. Lee, “Online self-supervised monocular visual odometry for ground vehicles,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 5232–5238. DOI: 10.1109/ICRA.2015.7139928.

- [182] G. Ros, S. Ramos, M. Granados, A. Bakhtiary, D. Vazquez, and A. M. Lopez, “Vision-Based Offline-Online Perception Paradigm for Autonomous Driving,” in *2015 IEEE Winter Conference on Applications of Computer Vision*, Jan. 2015, pp. 231–238. DOI: 10.1109/WACV.2015.38.
- [183] L. An, X. Zhang, H. Gao, and Y. Liu, “Semantic segmentation–aided visual odometry for urban autonomous driving,” *International Journal of Advanced Robotic Systems*, vol. 14, no. 5, Sep. 2017, ISSN: 1729-8814. DOI: 10.1177/1729881417735667.
- [184] A. Geiger, J. Ziegler, and C. Stiller, “StereoScan: Dense 3d reconstruction in real-time,” in *2011 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2011, pp. 963–968. DOI: 10.1109/IVS.2011.5940405.
- [185] Y. Choi, N. Kim, S. Hwang, K. Park, J. S. Yoon, K. An, and I. S. Kweon, “KAIST Multi-Spectral Day/Night Data Set for Autonomous and Assisted Driving,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 3, pp. 934–948, Mar. 2018, ISSN: 1524-9050. DOI: 10.1109/TITS.2018.2791533.
- [186] J. Poujol, C. A. Aguilera, E. Danos, B. X. Vintimilla, R. Toledo, and A. D. Sappa, “A Visible-Thermal Fusion Based Monocular Visual Odometry,” in *Robot 2015: Second Iberian Robotics Conference*, ser. Advances in Intelligent Systems and Computing, Springer, Cham, 2016, pp. 517–528, ISBN: 978-3-319-27145-3 978-3-319-27146-0. DOI: 10.1007/978-3-319-27146-0_40.
- [187] M. Persson, T. Piccini, M. Felsberg, and R. Mester, “Robust stereo visual odometry from monocular techniques,” in *2015 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2015, pp. 686–691. DOI: 10.1109/IVS.2015.7225764.
- [188] R. Wang, M. Schworer, and D. Cremers, “Stereo dso: Large-scale direct sparse visual odometry with stereo cameras,” in *The IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017.
- [189] M. Wu, S. K. Lam, and T. Srikanthan, “A Framework for Fast and Robust Visual Odometry,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 12, pp. 3433–3448, Dec. 2017, ISSN: 1524-9050. DOI: 10.1109/TITS.2017.2685433.
- [190] P. F. Proen  a and Y. Gao, “Probabilistic RGB-D odometry based on points, lines and planes under depth uncertainty,” *Robotics and Autonomous Systems*, vol. 104, pp. 25–39, Jun. 2018, ISSN: 0921-8890. DOI: 10.1016/j.robot.2018.02.018.
- [191] T. Holzmann, F. Fraundorfer, and H. Bischof, “A Detailed Description of Direct Stereo Visual Odometry Based on Lines,” in *Computer Vision, Imaging and Computer Graphics Theory and Applications*, ser. Communications in Computer and Information Science, Springer, Cham, Feb. 2016, pp. 353–373, ISBN: 978-3-319-64869-9 978-3-319-64870-5. DOI: 10.1007/978-3-319-64870-5_17.
- [192] S. Ceriani, G. Fontana, A. Giusti, D. Marzorati, M. Matteucci, D. Migliore, D. Rizzi, D. G. Sorrenti, and P. Taddei, “Rawseeds ground truth collection systems for indoor self-localization and mapping,” *Autonomous Robots*, vol. 27, no. 4, p. 353, Nov. 2009, ISSN: 0929-5593, 1573-7527. DOI: 10.1007/s10514-009-9156-5.
- [193] M. Jaimez, C. Kerl, J. Gonzalez-Jimenez, and D. Cremers, “Fast odometry and scene flow from RGB-D cameras based on geometric clustering,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 3992–3999. DOI: 10.1109/ICRA.2017.7989459.

- [194] Y. Liu, Y. Gu, J. Li, and X. Zhang, “Robust Stereo Visual Odometry Using Improved RANSAC-Based Methods for Mobile Robot Localization,” *Sensors (Basel, Switzerland)*, vol. 17, no. 10, Oct. 2017, ISSN: 1424-8220. DOI: 10.3390/s17102339.
- [195] Y. Kunii, G. Kovacs, and N. Hoshi, “Mobile robot navigation in natural environments using robust object tracking,” in *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*, Jun. 2017, pp. 1747–1752. DOI: 10.1109/ISIE.2017.8001512.
- [196] C.-H. Sun, Y.-J. Chen, Y.-T. Wang, and S.-K. Huang, “Sequentially switched fuzzy-model-based control for wheeled mobile robot with visual odometry,” *Applied Mathematical Modelling*, vol. 47, pp. 765–776, Jul. 2017, ISSN: 0307-904X. DOI: 10.1016/j.apm.2016.11.001.
- [197] D. H. Kim and J. H. Kim, “Effective Background Model-Based RGB-D Dense Visual Odometry in a Dynamic Environment,” *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1565–1573, Dec. 2016, ISSN: 1552-3098. DOI: 10.1109/TRO.2016.2609395.
- [198] S. H. Liu, C. C. Hsu, W. Y. Wang, M. Y. Chen, and Y. T. Wang, “Improved visual odometry system based on kinect RGB-D sensor,” in *2017 IEEE 7th International Conference on Consumer Electronics - Berlin (ICCE-Berlin)*, Sep. 2017, pp. 29–30. DOI: 10.1109/ICCE-Berlin.2017.8210581.
- [199] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, “Visual Odometry and Mapping for Autonomous Flight Using an RGB-D Camera,” in *Robotics Research*, ser. Springer Tracts in Advanced Robotics, Springer, Cham, 2017, pp. 235–252, ISBN: 978-3-319-29362-2 978-3-319-29363-9. DOI: 10.1007/978-3-319-29363-9_14.
- [200] M. Stefan, D. David, H. Dirk, F. Stefan, M. Ezio, N. Andreas, and H. Joachim, “Three-dimensional mapping with time-of-flight cameras,” *Journal of Field Robotics*, vol. 26, no. 11-12, pp. 934–965, Sep. 2009, ISSN: 1556-4959. DOI: 10.1002/rob.20321.
- [201] Z. Fang and Y. Zhang, “Experimental Evaluation of RGB-D Visual Odometry Methods,” *International Journal of Advanced Robotic Systems*, vol. 12, no. 3, p. 26, Mar. 2015, ISSN: 1729-8814. DOI: 10.5772/59991.
- [202] M. Maimone, Y. Cheng, and L. Matthies, “Two years of Visual Odometry on the Mars Exploration Rovers,” *Journal of Field Robotics*, vol. 24, pp. 169–186, 2007, ISSN: 15564959 15564967. DOI: 10.1002/rob.20184.
- [203] R. Strydom, S. Thurrowgood, and M. V. Srinivasan, “Visual odometry: Autonomous UAV navigation using optic flow and stereo,” in *Proceedings of Australasian conference on robotics and automation*, Australian Robotics and Automation Association, Jan. 2014.
- [204] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, “A benchmark for the evaluation of RGB-D SLAM systems,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2012, pp. 573–580. DOI: 10.1109/IROS.2012.6385773.
- [205] M. Jaimez and J. Gonzalez-Jimenez, “Fast Visual Odometry for 3-D Range Sensors,” *IEEE Transactions on Robotics*, vol. 31, no. 4, pp. 809–822, Aug. 2015, ISSN: 1552-3098. DOI: 10.1109/TRO.2015.2428512.

- [206] OpenMP. (2018). OpenMP, [Online]. Available: <http://www.openmp.org/> (visited on 04/19/2018).
- [207] NVIDIA Corporation. (2016). Parallel Programming and Computing Platform | CUDA, [Online]. Available: <https://developer.nvidia.com/cuda-zone>.
- [208] Y. Furukawa and C. Hernández, “Multi-View Stereo: A Tutorial,” *Foundations and Trends® in Computer Graphics and Vision*, vol. 9, no. 1-2, pp. 1–148, Jun. 2015, ISSN: 1572-2740, 1572-2759. DOI: 10.1561/0600000052.
- [209] C. Tomasi and T. Kanade, “Detection and Tracking of Point Features,” Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-CS-91-132, Apr. 1991, p. 22.
- [210] R. G. v. Gioi, J. Jakubowicz, J. M. Morel, and G. Randall, “LSD: A Fast Line Segment Detector with a False Detection Control,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 4, pp. 722–732, Apr. 2010, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2008.300.
- [211] A. Handa, T. Whelan, J. McDonald, and A. J. Davison, “A benchmark for RGB-D visual odometry, 3d reconstruction and SLAM,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 1524–1531. DOI: 10.1109/ICRA.2014.6907054.
- [212] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, “KinectFusion: Real-time dense surface mapping and tracking,” in *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, Oct. 2011, pp. 127–136. DOI: 10.1109/ISMAR.2011.6092378.
- [213] D. Gutierrez-Gomez, W. Mayol-Cuevas, and J. J. Guerrero, “Dense RGB-D visual odometry using inverse depth,” *Robotics and Autonomous Systems*, vol. 75, pp. 571–583, Jan. 2016, ISSN: 0921-8890. DOI: 10.1016/j.robot.2015.09.026.
- [214] J. Quiroga, T. Brox, F. Devernay, and J. Crowley, “Dense Semi-rigid Scene Flow Estimation from RGBD Images,” in *Computer Vision – ECCV 2014*, ser. Lecture Notes in Computer Science, Springer, Cham, Sep. 2014, pp. 567–582, ISBN: 978-3-319-10583-3 978-3-319-10584-0. DOI: 10.1007/978-3-319-10584-0_37.
- [215] S. Martull, M. Peris, and K. Fukui, “Realistic cg stereo image dataset with ground truth disparity maps,” *IEICE technical report. Speech*, vol. 111, no. 431, pp. 117–118, Feb. 2012.
- [216] O. Chum and J. Matas, “Matching with PROSAC - progressive sample consensus,” in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, vol. 1, Jun. 2005, 220–226 vol. 1. DOI: 10.1109/CVPR.2005.221.
- [217] M. Aladem, “Robust Real-Time Visual Odometry for Autonomous Ground Vehicles,” Master’s thesis, University of Michigan-Dearborn, Dearborn, MI, USA, Apr. 2017.
- [218] A. Alahi, R. Ortiz, and P. Vandergheynst, “FREAK: Fast Retina Keypoint,” in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2012, pp. 510–517. DOI: 10.1109/CVPR.2012.6247715.
- [219] M. Agrawal, K. Konolige, and M. R. Blas, “CenSurE: Center Surround Extremas for Realtime Feature Detection and Matching,” in *Computer Vision – ECCV 2008*, ser. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, Oct. 2008, pp. 102–115, ISBN: 978-3-540-88692-1 978-3-540-88693-8. DOI: 10.1007/978-3-540-88693-8_8.

- [220] T. Takagi and M. Sugeno, “Fuzzy identification of systems and its applications to modeling and control,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-15, no. 1, pp. 116–132, Jan. 1985, ISSN: 0018-9472. DOI: 10.1109/TSMC.1985.6313399.
- [221] A. Elgammal, D. Harwood, and L. Davis, “Non-parametric Model for Background Subtraction,” in *Computer Vision — ECCV 2000*, ser. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, Jun. 2000, pp. 751–767, ISBN: 978-3-540-67686-7 978-3-540-45053-5. DOI: 10.1007/3-540-45053-X_48.
- [222] F. Steinbrücker, J. Sturm, and D. Cremers, “Real-time visual odometry from dense RGB-D images,” in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, Nov. 2011, pp. 719–722. DOI: 10.1109/ICCVW.2011.6130321.
- [223] C. Kerl, J. Sturm, and D. Cremers, “Robust odometry estimation for RGB-D cameras,” in *2013 IEEE International Conference on Robotics and Automation*, May 2013, pp. 3748–3754. DOI: 10.1109/ICRA.2013.6631104.
- [224] J. Gui, D. Gu, S. Wang, and H. Hu, “A review of visual inertial odometry from filtering and optimisation perspectives,” *Advanced Robotics*, vol. 29, no. 20, pp. 1289–1301, Oct. 2015, ISSN: 0169-1864, 1568-5535. DOI: 10.1080/01691864.2015.1057616.
- [225] K. Konolige, M. Agrawal, and J. Solà, “Large-Scale Visual Odometry for Rough Terrain,” in *Robotics Research*, ser. Springer Tracts in Advanced Robotics, Springer, Berlin, Heidelberg, 2010, pp. 201–212, ISBN: 978-3-642-14742-5 978-3-642-14743-2. DOI: 10.1007/978-3-642-14743-2_18.
- [226] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, “Keyframe-based visual–inertial odometry using nonlinear optimization,” *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, Mar. 2015, ISSN: 0278-3649. DOI: 10.1177/0278364914554813.
- [227] A. I. Mourikis and S. I. Roumeliotis, “A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation,” in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, Apr. 2007, pp. 3565–3572. DOI: 10.1109/ROBOT.2007.364024.
- [228] T. Qin, P. Li, and S. Shen, “Vins-mono: A robust and versatile monocular visual–inertial state estimator,” *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, Aug. 2018, ISSN: 1552-3098. DOI: 10.1109/TRO.2018.2853729.
- [229] M. Faessler, F. Fontana, C. Forster, E. Mueggler, M. Pizzoli, and D. Scaramuzza, “Autonomous, Vision-based Flight and Live Dense 3d Mapping with a Quadrotor Micro Aerial Vehicle: Autonomous, Vision-based Flight and Live Dense 3d Mapping,” *Journal of Field Robotics*, vol. 33, no. 4, pp. 431–450, Jun. 2016, ISSN: 1556-4959. DOI: 10.1002/rob.21581.
- [230] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, “On-Manifold Preintegration for Real-Time Visual–Inertial Odometry,” *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, Feb. 2017, ISSN: 1552-3098. DOI: 10.1109/TRO.2016.2597321.

- [231] J. Delmerico and D. Scaramuzza, “A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, May 2018, pp. 2502–2509. DOI: 10.1109/ICRA.2018.8460664.
- [232] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, “The EuRoC micro aerial vehicle datasets,” *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1157–1163, Sep. 2016, ISSN: 0278-3649. DOI: 10.1177/0278364915620033.
- [233] K. Sun, K. Mohta, B. Pfrommer, M. Watterson, S. Liu, Y. Mulgaonkar, C. J. Taylor, and V. Kumar, “Robust Stereo Visual Inertial Odometry for Fast Autonomous Flight,” *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 965–972, Apr. 2018. DOI: 10.1109/LRA.2018.2793349.
- [234] Y. Liu, R. Xiong, Y. Wang, H. Huang, X. Xie, X. Liu, and G. Zhang, “Stereo Visual-Inertial Odometry With Multiple Kalman Filters Ensemble,” *IEEE Transactions on Industrial Electronics*, vol. 63, no. 10, pp. 6205–6216, Oct. 2016, ISSN: 0278-0046. DOI: 10.1109/TIE.2016.2573765.
- [235] A. R. Vidal, H. Rebecq, T. Horstschaefer, and D. Scaramuzza, “Ultimate SLAM? Combining Events, Images, and IMU for Robust Visual SLAM in HDR and High-Speed Scenarios,” *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 994–1001, Apr. 2018. DOI: 10.1109/LRA.2018.2793357.
- [236] S. Agarwal, K. Mierle, *et al.* (2019). Ceres solver.
- [237] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza, “The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM,” *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 142–149, Feb. 2017, ISSN: 0278-3649. DOI: 10.1177/0278364917691115.
- [238] S. Lovegrove, A. J. Davison, and J. Ibañez-Guzmán, “Accurate visual odometry from a rear parking camera,” in *2011 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2011, pp. 788–793. DOI: 10.1109/IVS.2011.5940546.
- [239] NVIDIA Corporation. (2018). Autonomous Car Development Platform from NVIDIA DRIVE PX2, [Online]. Available: <https://www.nvidia.com/en-us/self-driving-cars/drive-platform/> (visited on 04/20/2018).
- [240] A. Boeing, M. Boulton, T. Bräunl, B. Frisch, S. Lopes, A. Morgan, F. Ophelders, S. Pangeni, R. Reid, K. Vinsen, N. Garel, C. S. Lee, M. Masek, A. Attwood, M. Fazio, and A. Gandossi, “WAMbot: Team MAGICian’s entry to the Multi Autonomous Ground-robotic International Challenge 2010,” *Journal of Field Robotics*, vol. 29, no. 5, pp. 707–728, Jul. 2012, ISSN: 1556-4959. DOI: 10.1002/rob.21434.
- [241] C. L. Ortiz, R. Vincent, and B. Morisset, “Task Inference and Distributed Task Management in the Centibots Robotic System,” in *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS ’05, New York, NY, USA: ACM, 2005, pp. 860–867, ISBN: 978-1-59593-093-4. DOI: 10.1145/1082473.1082604.
- [242] R. Reid, “Large-Scale Simultaneous Localization and Mapping for Teams of Mobile Robots,” PhD Thesis, The University of Western Australia, Perth, Australia, Jul. 2016.

- [243] A. Boeing, S. Pangeni, T. Bräunl, and C. S. Lee, “Real-time tactical motion planning and obstacle avoidance for multi-robot cooperative reconnaissance,” in *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct. 2012, pp. 3117–3122, ISBN: 1062-922X. DOI: 10.1109/ICSMC.2012.6378270.
- [244] A. Boeing, T. Bräunl, R. Reid, A. Morgan, and K. Vinsen, “Cooperative multi-robot navigation and mapping of unknown terrain,” in *2011 IEEE 5th International Conference on Robotics, Automation and Mechatronics (RAM)*, Sep. 2011, pp. 234–238, ISBN: 2158-219X. DOI: 10.1109/RAMECH.2011.6070488.
- [245] L. Luft, T. Schubert, S. I. Roumeliotis, and W. Burgard, “Recursive decentralized localization for multi-robot systems with asynchronous pairwise communication,” *The International Journal of Robotics Research*, p. 0278 364 918 760 698, Mar. 2018, ISSN: 0278-3649. DOI: 10.1177/0278364918760698.
- [246] M. Garzón, J. Valente, J. J. Roldán, L. Cancar, A. Barrientos, and J. D. Cerro, “A Multirobot System for Distributed Area Coverage and Signal Searching in Large Outdoor Scenarios*,” *Journal of Field Robotics*, vol. 33, no. 8, pp. 1087–1106, Dec. 2016, ISSN: 1556-4967. DOI: 10.1002/rob.21636.
- [247] G. Best, O. M. Cliff, T. Patten, R. R. Mettu, and R. Fitch, “Dec-mcts: Decentralized planning for multi-robot active perception,” *The International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 316–337, Mar. 2019. DOI: 10.1177/0278364918755924. eprint: <https://doi.org/10.1177/0278364918755924>.
- [248] S. Saeedi, M. Trentini, M. Seto, and H. Li, “Multiple-Robot Simultaneous Localization and Mapping: A Review,” *Journal of Field Robotics*, vol. 33, no. 1, pp. 3–46, Jan. 2016, ISSN: 1556-4967. DOI: 10.1002/rob.21620.
- [249] M. A. Abdulgalil, M. M. Nasr, M. H. Elalfy, A. Khamis, and F. Karray, “Multi-robot SLAM: An Overview and Quantitative Evaluation of MRGS ROS Framework for MR-SLAM,” in *Robot Intelligence Technology and Applications 5*, J.-H. Kim, H. Myung, J. Kim, W. Xu, E. T. Matson, J.-W. Jung, and H.-L. Choi, Eds., ser. Advances in Intelligent Systems and Computing, Springer International Publishing, 2019, pp. 165–183, ISBN: 978-3-319-78452-6.
- [250] J. Jung, S. Yoon, S. Ju, J. Heo, J. Jung, S. Yoon, S. Ju, and J. Heo, “Development of Kinematic 3d Laser Scanning System for Indoor Mapping and As-Built BIM Using Constrained SLAM,” *Sensors*, vol. 15, no. 10, pp. 26 430–26 456, Oct. 2015. DOI: 10.3390/s151026430.
- [251] P. Koch, S. May, M. Schmidpeter, M. Kühn, C. Pfitzner, C. Merkl, R. Koch, M. Fees, J. Martin, D. Ammon, and A. Nüchter, “Multi-Robot Localization and Mapping Based on Signed Distance Functions,” *Journal of Intelligent & Robotic Systems*, vol. 83, no. 3, pp. 409–428, Sep. 2016, ISSN: 1573-0409. DOI: 10.1007/s10846-016-0375-7.
- [252] G. Zhou, B. Bescos, M. Dymczyk, M. Pfeiffer, J. Neira, and R. Siegwart, “Dynamic Objects Segmentation for Visual Localization in Urban Environments,” *arXiv:1807.02996 [cs]*, Jul. 2018, arXiv: 1807.02996.
- [253] Adept Technology, Inc. (2018). Pioneer 3-AT, [Online]. Available: <http://www.mobilerobots.com/Libraries/Downloads/Pioneer3AT-P3AT-RevA.sflb.ashx> (visited on 10/05/2018).

- [254] AutonomouStuff. (2018). Ibeo Standard Four Layer Multi-Echo LUX Sensor | LiDAR | Product, [Online]. Available: <https://autonomoustuff.com/product/ibeo-lux-standard/> (visited on 09/10/2017).
- [255] SICK AG. (2018). LMS111-10100, [Online]. Available: <https://www.sick.com/au/en/detection-and-ranging-solutions/2d-lidar-sensors/lms1xx/lms111-10100/p/p109842> (visited on 09/05/2018).
- [256] Hokuyo Automatic Co., Ltd. (2018). Scanning Rangefinder Distance Data Output/URG-04lx-UG01, [Online]. Available: <https://www.hokuyo-aut.jp/search/single.php?serial=166> (visited on 10/05/2018).
- [257] Xsens. (2018). MTi (legacy product) - Products, [Online]. Available: <https://www.xsens.com/products/mti/> (visited on 09/10/2017).
- [258] QStarz International Co., Ltd. (2018). BT-Q818xt, [Online]. Available: <http://www.qstarz.com/Products/GPS%20Products/BT-Q818XT-F.htm> (visited on 10/05/2018).
- [259] Logitech. (2018). QuickCam® Orbit AF, [Online]. Available: https://support.logitech.com/en_us/product/quickcam-sphere-af/specs (visited on 10/05/2018).
- [260] Ubiquiti Networks. (2018). PicoStation2hp Datasheet, [Online]. Available: https://dl.ubnt.com/pico2hp_ds.pdf (visited on 10/05/2018).
- [261] STI Engineering Pty Ltd. (2018). RFInnovations RFI-9256 900mhz High Speed Data Radio, [Online]. Available: [http://www.rfinnovations.com.au/Uploads/Images/900MHz%20Data%20Radio%20Modem\(2\).pdf](http://www.rfinnovations.com.au/Uploads/Images/900MHz%20Data%20Radio%20Modem(2).pdf) (visited on 10/05/2018).
- [262] R. Reid and T. Bräunl, “Large-scale multi-robot mapping in MAGIC 2010,” in *2011 IEEE 5th International Conference on Robotics, Automation and Mechatronics (RAM)*, Sep. 2011, pp. 239–244, ISBN: 2158-219X. DOI: 10.1109/RAMECH.2011.6070489.
- [263] R. Reid, A. Cann, C. Meiklejohn, L. Poli, A. Boeing, and T. Bräunl, “Cooperative multi-robot navigation, exploration, mapping and object detection with ROS,” in *2013 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2013, pp. 1083–1088, ISBN: 1931-0587. DOI: 10.1109/IVS.2013.6629610.
- [264] J. E. Bresenham, “Algorithm for computer control of a digital plotter,” *IBM Systems Journal*, vol. 4, no. 1, pp. 25–30, 1965, ISSN: 0018-8670. DOI: 10.1147/sj.41.0025.
- [265] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, “A solution to the simultaneous localization and map building (SLAM) problem,” *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 229–241, Jun. 2001, ISSN: 1042-296X. DOI: 10.1109/70.938381.
- [266] A. Censi, “An accurate closed-form estimate of ICP’s covariance,” in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, Apr. 2007, pp. 3167–3172. DOI: 10.1109/ROBOT.2007.363961.
- [267] J. M. Coughlan and A. L. Yuille, “Manhattan World: Orientation and Outlier Detection by Bayesian Inference,” *Neural Computation*, vol. 15, no. 5, pp. 1063–1088, May 2003, ISSN: 0899-7667. DOI: 10.1162/089976603765202668.
- [268] S. Lopes, B. Frisch, A. Boeing, K. Vinsen, and T. Bräunl, “Autonomous exploration of unknown terrain for groups of mobile robots,” in *2011 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2011, pp. 157–162, ISBN: 1931-0587. DOI: 10.1109/IVS.2011.5940455.

- [269] M. O. A. Aqel, M. H. Marhaban, M. I. Saripan, and N. B. Ismail, “Review of visual odometry: Types, approaches, challenges, and applications,” *SpringerPlus*, vol. 5, no. 1, p. 1897, Oct. 2016, ISSN: 2193-1801. DOI: 10.1186/s40064-016-3573-7.
- [270] B. Zhao, T. Hu, and L. Shen, “Visual odometry - a review of approaches,” in *2015 IEEE International Conference on Information and Automation*, IEEE, Aug. 2015, pp. 2569–2573. DOI: 10.1109/ICInfA.2015.7279718.
- [271] T. Drage, J. Kalinowski, and T. Bräunl, “Integration of Drive-by-Wire with Navigation Control for a Driverless Electric Race Car,” *IEEE Intelligent Transportation Systems Magazine*, vol. 6, pp. 23–33, 2014, ISSN: 1939-1390. DOI: 10.1109/MITS.2014.2327160.
- [272] P. Y. Shinzato, D. Gomes, and D. F. Wolf, “Road estimation with sparse 3d points from stereo data,” in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, IEEE, Oct. 2014, pp. 1688–1693, ISBN: 2153-0009. DOI: 10.1109/ITSC.2014.6957936.
- [273] K. Y. Guo, E. G. Hoare, D. Jasteh, X. Q. Sheng, and M. Gashinova, “Road Edge Recognition Using the Stripe Hough Transform From Millimeter-Wave Radar Images,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 825–833, 2015, ISSN: 1524-9050. DOI: 10.1109/TITS.2014.2342875.
- [274] W. Zhang, “LIDAR-based road and road-edge detection,” in *2010 IEEE Intelligent Vehicles Symposium*, 2010, pp. 845–848, ISBN: 1931-0587. DOI: 10.1109/IVS.2010.5548134.
- [275] M. Nikolova and A. Hero, “Segmentation of a road from a vehicle-mounted radar and accuracy of the estimation,” in *Proceedings of the IEEE Intelligent Vehicles Symposium 2000 (Cat. No.00TH8511)*, 2000, pp. 284–289. DOI: 10.1109/IVS.2000.898356.
- [276] G. Zhao and J. Yuan, “Curb detection and tracking using 3d-LIDAR scanner,” in *2012 19th IEEE International Conference on Image Processing*, 2012, pp. 437–440, ISBN: 1522-4880. DOI: 10.1109/ICIP.2012.6466890.
- [277] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006. DOI: 10.1017/CBO9780511546877.
- [278] Y. Liu and Y. Sun, “Mobile robot instant indoor map building and localization using 2d laser scanning data,” in *2012 International Conference on System Science and Engineering (ICSSE)*, 2012, pp. 339–344, ISBN: 2325-0909. DOI: 10.1109/ICSSE.2012.6257203.
- [279] M. Fu, H. Zhu, Y. Yang, M. Wang, and Y. Li, “Multiple map representations for vehicle localization and scene reconstruction,” in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, IEEE, Oct. 2014, pp. 2241–2242, ISBN: 2153-0009. DOI: 10.1109/ITSC.2014.6958036.
- [280] D. M. Cole and P. M. Newman, “Using laser range data for 3d SLAM in outdoor environments,” in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, IEEE, May 2006, pp. 1556–1563, ISBN: 1050-4729. DOI: 10.1109/ROBOT.2006.1641929.

- [281] H. Zhu, M. Fu, Y. Yang, X. Wang, and M. Wang, “A path planning algorithm based on fusing lane and obstacle map,” in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, IEEE, Oct. 2014, pp. 1442–1448, ISBN: 2153-0009. DOI: 10.1109/ITSC.2014.6957889.
- [282] J. Gillula and J. Leibs, “How to teach a van to drive: An undergraduate perspective on the 2005 DARPA grand challenge,” *IEEE Control Systems*, vol. 26, pp. 19–26, 2006, ISSN: 1066-033X. DOI: 10.1109/MCS.2006.1636306.
- [283] F. Zhang, D. Clarke, and A. Knoll, “Vehicle detection based on LiDAR and camera fusion,” in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, IEEE, Oct. 2014, pp. 1620–1625, ISBN: 2153-0009. DOI: 10.1109/ITSC.2014.6957925.
- [284] A. Gudigar, S. Chokkadi, and R. U, “A review on automatic detection and recognition of traffic sign,” *Multimedia Tools and Applications*, vol. 75, pp. 333–364, 2016, ISSN: 1573-7721. DOI: 10.1007/s11042-014-2293-7.
- [285] J. Yang, S. Zhang, G. Wang, and M. Li, “Scene and place recognition using a hierarchical latent topic model,” *Neurocomputing*, vol. 148, pp. 578–586, 2015, ISSN: 0925-2312. DOI: 10.1016/j.neucom.2014.07.005.
- [286] H. Cho, Y.-W. Seo, B. V. Kumar, and R. R. Rajkumar, “A multi-sensor fusion system for moving object detection and tracking in urban driving environments,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, May 2014, pp. 1836–1843, ISBN: 1050-4729. DOI: 10.1109/ICRA.2014.6907100.
- [287] T. Drage, T. Churack, and T. Bräunl, “LIDAR Road Edge Detection by Heuristic Evaluation of Many Linear Regressions,” in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, Sep. 2015, pp. 2465–2470. DOI: 10.1109/ITSC.2015.397.
- [288] Raspberry Pi Foundation. (2017). Raspberry Pi 3 Model B, [Online]. Available: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>.
- [289] SAE International. (2018). Student Events - Events - Collegiate Design Series, [Online]. Available: <https://www.sae.org/attend/student-events/> (visited on 04/29/2018).
- [290] T. H. Drage, “Development of a Navigation Control System for an Autonomous Formula SAE-Electric Race Car,” BE Thesis, The University of Western Australia, Perth, Australia, Nov. 2013.
- [291] Baidu. (2018). Apollo, [Online]. Available: <http://apollo.auto/> (visited on 02/13/2018).
- [292] Autoware. (2018). Autoware, [Online]. Available: <https://autoware.ai/> (visited on 02/13/2018).
- [293] Google Developers. (2018). Protocol Buffers, [Online]. Available: <https://developers.google.com/protocol-buffers/> (visited on 09/10/2017).
- [294] K. Chu, M. Lee, and M. Sunwoo, “Local Path Planning for Off-Road Autonomous Driving With Avoidance of Static Obstacles,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1599–1616, Dec. 2012, ISSN: 1524-9050. DOI: 10.1109/TITS.2012.2198214.
- [295] H. Wang, J. Kearney, and K. Atkinson, “Arc-length parameterized spline curves for real-time simulation,” in *Proc. 5th International Conference on Curves and Surfaces*, 2002.

- [296] R. P. Brent, *Algorithms for minimization without derivatives*. Englewood Cliffs: Courier Corporation, 1973, ISBN: 0-486-14368-6.
- [297] J. Xu, W. Liu, H. Bian, and L. Li, “Accurate and Efficient Algorithm for the Closest Point on a Parametric Curve,” in *2008 International Conference on Computer Science and Software Engineering*, vol. 2, Dec. 2008, pp. 1000–1002. DOI: 10.1109/CSSE.2008.618.
- [298] E. S. Raymond. (2018). GPSd — Put your GPS on the net! [Online]. Available: <http://www.catb.org/gpsd/> (visited on 09/10/2017).
- [299] Arduino. (2018). Arduino Uno Rev3, [Online]. Available: <https://store.arduino.cc/usa/arduino-uno-rev3> (visited on 09/10/2017).
- [300] J. Canny, “A Computational Approach to Edge Detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, Nov. 1986, ISSN: 0162-8828. DOI: 10.1109/TPAMI.1986.4767851.
- [301] P. V. C. Hough, “Method and means for recognizing complex patterns,” US3069654A, Dec. 1962.
- [302] K. L. Lim, T. Drage, R. Podolski, G. Meyer-Lee, S. Evans-Thompson, J. Y.-T. Lin, G. Channon, M. Poole, and T. Bräunl, “A modular software framework for autonomous vehicles,” in *2018 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2018, pp. 1780–1785. DOI: 10.1109/IVS.2018.8500474.
- [303] Open Source Robotics Foundation. (2019). ROS/Introduction - ROS Wiki, [Online]. Available: <http://wiki.ros.org/ROS/Introduction> (visited on 08/29/2018).
- [304] Arduino. (2018). Arduino Nano, [Online]. Available: <https://store.arduino.cc/arduino-nano> (visited on 09/05/2018).
- [305] Xsens. (2019). MTi-G-710, [Online]. Available: <https://www.xsens.com/products/mti-g-710/> (visited on 09/05/2018).
- [306] B. M. Yu, K. V. Shenoy, and M. Sahani, *Derivation of Extended Kalman Filtering and Smoothing Equations*, Oct. 19, 2004.
- [307] D. Morrell, “Extended Kalman Filter Lecture Notes,” *EEE 581-Spring, Arizona State University*, 1997.
- [308] T. Moore and D. Stouch, “A Generalized Extended Kalman Filter Implementation for the Robot Operating System,” in *Intelligent Autonomous Systems 13*, E. Menegatti, N. Michael, K. Berns, and H. Yamaguchi, Eds., ser. Advances in Intelligent Systems and Computing, Springer International Publishing, 2016, pp. 335–348, ISBN: 978-3-319-08338-4.
- [309] FLIR. (2018). Blackfly 1.3 MP Color GigE PoE (Sony ICX445), [Online]. Available: <https://www.ptgrey.com/blackfly-13-mp-color-gige-vision-poe-sony-icx445-camera> (visited on 09/05/2018).
- [310] ——, (2018). Fujinon YV2.8×2.8sa-2, 2.8mm-8mm, 1/3", CS mount Lens, [Online]. Available: <https://www.ptgrey.com/fujinon-yv28x28sa-2-hd-vari-focal-lens-3> (visited on 09/05/2018).
- [311] C. Liang, L. Chang, and H. H. Chen, “Analysis and Compensation of Rolling Shutter Effect,” *IEEE Transactions on Image Processing*, vol. 17, no. 8, pp. 1323–1330, Aug. 2008, ISSN: 1057-7149. DOI: 10.1109/TIP.2008.925384.

- [312] H. R. Kam, S.-H. Lee, T. Park, and C.-H. Kim, “RViz: A Toolkit for Real Domain Data Visualization,” *Telecommun. Syst.*, vol. 60, no. 2, pp. 337–345, Oct. 2015, ISSN: 1018-4864. DOI: 10.1007/s11235-015-0034-5.
- [313] C. De Boor, *A practical guide to splines*. Springer-Verlag New York, 1978, vol. 27.
- [314] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.
- [315] K. L. Lim, T. Drage, and T. Bräunl, “Implementation of semantic segmentation for road and lane detection on an autonomous ground vehicle with LIDAR,” in *2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, IEEE, Nov. 2017, pp. 429–434. DOI: 10.1109/MFI.2017.8170358.
- [316] Y. Chen. (Feb. 2019). Yunchih/ORB-SLAM2-GPU2016-final. original-date: 2016-05-09T22:52:27Z, [Online]. Available: <https://github.com/yunchih/ORB-SLAM2-GPU2016-final> (visited on 02/25/2019).
- [317] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Region-Based Convolutional Networks for Accurate Object Detection and Segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 1, pp. 142–158, Jan. 2016, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2015.2437384.
- [318] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “CARLA: An Open Urban Driving Simulator,” in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.
- [319] Logitech. (2019). Logitech G920 & G29 Driving Force Steering Wheels & Pedals, [Online]. Available: <https://www.logitechg.com/en-au/products/driving/driving-force-racing-wheel.html> (visited on 06/02/2019).
- [320] Formula Student Germany GmbH, “FSG Competition Handbook 2018,” Formula Student Germany GmbH, Wiesbaden, Germany, Tech. Rep. 1.1, Apr. 2018, p. 21.
- [321] Apollo Auto. (May 2019). Collections of Apollo Platform Software. original-date: 2017-06-30T18:53:11Z, [Online]. Available: <https://github.com/ApolloAuto/apollo-platform> (visited on 06/02/2019).
- [322] BBC News, “Self-drive buses enter ‘mass production’,” *BBC NEews*, Jul. 4, 2018.
- [323] Autoware. (Jun. 2019). Open-source software for self-driving vehicles. original-date: 2015-08-24T23:17:57Z, [Online]. Available: <https://github.com/autowarefoundation/autoware> (visited on 06/02/2019).
- [324] O. Gietelink, J. Ploeg, B. D. Schutter, and M. Verhaegen, “Development of advanced driver assistance systems with vehicle hardware-in-the-loop simulations,” *Vehicle System Dynamics*, vol. 44, no. 7, pp. 569–590, Jul. 2006, ISSN: 0042-3114. DOI: 10.1080/00423110600563338.
- [325] F. Bella and R. Russo, “A Collision Warning System for rear-end collision: A driving simulator study,” *Procedia - Social and Behavioral Sciences*, The State of the Art in the European Quantitative Oriented Transportation and Logistics Research – 14th Euro Working Group on Transportation & 26th Mini Euro Conference & 1st European Scientific Conference on Air Transport, vol. 20, pp. 676–686, Jan. 2011, ISSN: 1877-0428. DOI: 10.1016/j.sbspro.2011.08.075.

- [326] B. Hassan, J. Berssenbrügge, I. A. Qaisi, and J. Stöcklein, “Reconfigurable driving simulator for testing and training of advanced driver assistance systems,” in *2013 IEEE International Symposium on Assembly and Manufacturing (ISAM)*, Jul. 2013, pp. 337–339. DOI: 10.1109/ISAM.2013.6643472.
- [327] T. Chapron and J.-P. Colinot, “The New PSA Peugeot-Citroen Advanced Driving Simulator Overall Design and Motion Cue Algorithm,” in *Proceedings of the Driving Simulation Conference, North America 2007 (DSC-NA 2007)*, Iowa City IA, United States, Sep. 2007.
- [328] T. Murano, T. Yonekawa, M. Aga, and S. Nagiri, “Development of High-Performance Driving Simulator,” *SAE International Journal of Passenger Cars - Mechanical Systems*, vol. 2, no. 1, pp. 661–669, Apr. 2009. DOI: 10.4271/2009-01-0450.
- [329] W. Käding and F. Hoffmeyer, “The advanced daimler-benz driving simulator,” in *SAE Technical Paper*, SAE International, Feb. 1995. DOI: 10.4271/950175.
- [330] J. S. Brodsky, “Autonomous Vehicle Regulation: How an Uncertain Legal Landscape May Hit the Brakes on Self-Driving Cars,” *Berkeley Technology Law Journal*, vol. 31, p. 851, 2016.
- [331] cognata. (2019). Cognata, [Online]. Available: <https://www.cognata.com/> (visited on 06/02/2019).
- [332] rFpro. (2019). rFpro, [Online]. Available: <http://www.rfpro.com/> (visited on 06/02/2019).
- [333] NVIDIA Corporation. (2019). NVIDIA DRIVE Constellation., [Online]. Available: <https://www.nvidia.com/en-au/self-driving-cars/drive-constellation/> (visited on 06/02/2019).
- [334] L. Li, W. Huang, Y. Liu, N. Zheng, and F. Wang, “Intelligence Testing for Autonomous Vehicles: A New Approach,” *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 2, pp. 158–166, Jun. 2016, ISSN: 2379-8904. DOI: 10.1109/TIV.2016.2608003.
- [335] S. Bradley, “Automotive Simulation System,” Master’s thesis, The University of Western Australia, Perth, Australia, 2009.
- [336] Epic Games. (2019). Unreal Engine, [Online]. Available: <https://www.unrealengine.com/en-US/what-is-unreal-engine-4> (visited on 06/02/2019).
- [337] Canonical Ltd. (2019). Ubuntu Manpage: Clockdiff - measure clock difference between hosts, [Online]. Available: <http://manpages.ubuntu.com/manpages/bionic/man8/clockdiff.8.html> (visited on 06/02/2019).
- [338] Carla. (2019). Cameras and sensors - CARLA Simulator, [Online]. Available: https://carla.readthedocs.io/en/latest/cameras_and_sensors/ (visited on 06/02/2019).
- [339] ROS. (2019). Sensor_msgs/LaserScan Documentation, [Online]. Available: http://docs.ros.org/melodic/api/sensor_msgs/html/msg/LaserScan.html (visited on 06/02/2019).
- [340] ——, (2019). LMS1xx - ROS Wiki, [Online]. Available: <http://wiki.ros.org/LMS1xx> (visited on 06/02/2019).
- [341] ——, (2019). Pointcloud_to_laserscan - ROS Wiki, [Online]. Available: http://wiki.ros.org/pointcloud_to_laserscan (visited on 06/02/2019).

- [342] S. Lauxtermann, A. Lee, J. Stevens, and A. Joshi, “Comparison of global shutter pixels for CMOS image sensors,” in *Proc, 2007 International Image Sensor Workshop*, Ogunquit Maine, USA, Jun. 2007, pp. 82–85.
- [343] G. Sebastien. (Jun. 2019). Performance monitoring tools for Linux. original-date: 2013-04-25T11:23:10Z, [Online]. Available: <https://github.com/sysstat/sysstat> (visited on 06/02/2019).
- [344] IEEE, “IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems,” *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, pp. 1–300, Jul. 2008. DOI: 10.1109/IEEEESTD.2008.4579760.
- [345] Bureau of Meteorology. (Dec. 2013). Climate Data Online, [Online]. Available: <http://www.bom.gov.au/climate/data/> (visited on 06/02/2019).
- [346] V. C. Magana and M. Munoz-Orgaño, “GAFU: Using a Gamification Tool to Save Fuel,” *IEEE Intelligent Transportation Systems Magazine*, vol. 7, no. 2, pp. 58–70, 2015, ISSN: 1939-1390. DOI: 10.1109/MITST.2015.2408152.
- [347] T. Mader and T. Bräunl, “Western Australian Electric Vehicle Trial,” The University of Western Australia, Perth, Australia, Tech. Rep., 2013, p. 59.
- [348] S. Speidel and T. Bräunl, “Driving and charging patterns of electric vehicles for energy usage,” *Renewable and Sustainable Energy Reviews*, vol. 40, pp. 97–110, Dec. 2014, ISSN: 1364-0321. DOI: 10.1016/j.rser.2014.07.177.
- [349] F. Jabeen, D. Olaru, B. Smith, T. Bräunl, and S. Speidel, “Electric vehicle battery charging behaviour: Findings from a driver survey,” in *Australasian Transport Research Forum (ATRF), 36th, 2013, Brisbane, Queensland, Australia*, Brisbane, 2013.
- [350] S. Speidel, F. Jabeen, D. Olaru, D. Harries, and T. Bräunl, “Analysis of Western Australian electric vehicle and charging station trials,” in *35th 2012 Australasian Transport Research Forum (ATRF)*, Sep. 2012.
- [351] F. Jabeen, D. Olaru, B. Smith, T. Bräunl, and S. Speidel, “Acceptability of electric vehicles: Findings from a driver survey,” in *Australasian Transport Research Forum (ATRF), 35th, 2012, Perth, Western Australia, Australia*, Perth, Australia, Sep. 2012.
- [352] International Energy Agency, “Global EV Outlook 2018,” International Energy Agency, Paris, Report, May 2017, p. 141.
- [353] Á. Rodríguez-Serrano, A. Torralba, E. Rodríguez-Valencia, and J. Tarifa-Galisteo, “A communication system from EV to EV Service Provider based on OCPP over a wireless network,” in *IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*, Nov. 2013, pp. 5434–5438, ISBN: 1553-572X. DOI: 10.1109/IECON.2013.6700020.
- [354] AECOM Australia, “Economic Viability of Electric Vehicles,” AECOM, Sydney, Report 60099409, Sep. 2009, p. 102.
- [355] R. Kinghorn and D. Kua, “Forecast Uptake and Economic Evaluation of Electric Vehicles in Victoria,” AECOM, Melbourne, Report 60149263, May 2011, p. 105.
- [356] N. Roberts, “Media Release: ABMARC releases key findings from their electric & hybrid vehicles report,” Oct. 2012.

- [357] Energy Supply Association of Australia, “Sparking an electric vehicle debate in australia,” *Discussion paper*, 2013, Cover title.
- [358] Clean Energy Council, “Clean Energy Australia Report 2019,” Clean Energy Council Australia, Melbourne, Report, Apr. 4, 2019, p. 82.
- [359] D. Anair and A. Mahmassani, “State of charge,” *Union of Concerned Scientists*, vol. 10, 2012.
- [360] Department of Infrastructure and Regional Development. (2018). Green Vehicle Guide, [Online]. Available: <https://www.greenvehicleguide.gov.au/> (visited on 02/02/2018).
- [361] K. van Namen, A. Tieu, and P. Olden, “Green House Gas Emissions from Households in Western Australia,” SMEC Australia, Perth, Report 3006127, Aug. 2011, p. 47.
- [362] Department of Infrastructure and Regional Development. (Dec. 2016). Vehicle emissions standards for cleaner air, [Online]. Available: https://infrastructure.gov.au/roads/environment/forum/files/Vehicle_Noxious_Emissions_RIS.pdf (visited on 02/02/2018).
- [363] National Transport Commission, “Carbon Dioxide Emissions from New Australian Vehicles 2013,” National Transport Commission, Information paper, May 2014, p. 68.
- [364] ChargePoint Inc. (2019). ChargePoint, [Online]. Available: <https://www.chargepoint.com> (visited on 04/02/2019).
- [365] China Electric Vehicle Charging Infrastructure Promotion Alliance, “Zhongguo diandong qiche chongdian jichu sheshi fazhan niandu baogao 2016-2017 ban [China Electric Vehicle Charging Infrastructure Development Annual Report 2016-2017 Edition],” National Energy Administration, Beijing, China, Tech. Rep., Apr. 2017, p. 76.
- [366] ChargeStar. (2018). ChargeStar, [Online]. Available: <https://www.chargestar.com.au/> (visited on 04/10/2019).
- [367] Go Electric Stations S.r.l.s. (2019). Go Electric Stations, [Online]. Available: <https://goelectricstations.com/> (visited on 04/02/2019).
- [368] GeoTelematic Solutions, Inc. (2019). OpenGTS, [Online]. Available: <http://www.opengts.org/> (visited on 04/02/2019).
- [369] Traccar Ltd. (2019). Traccar, [Online]. Available: <https://www.traccar.org/> (visited on 04/02/2019).
- [370] G. AB. (2019). GpsGate, [Online]. Available: <https://gpsgate.com/> (visited on 04/02/2019).
- [371] NETSTAR. (2019). EZY2c GPS Tracking, [Online]. Available: <https://www.ezy2c.com.au/> (visited on 04/23/2019).
- [372] Verizon. (2019). Fleetmatics, [Online]. Available: <https://www.verizonconnect.com/au/fleetmatics/> (visited on 04/23/2019).
- [373] Linxio. (2019). Linxio, [Online]. Available: <https://linxio.com/> (visited on 04/23/2019).
- [374] SolarEdge Technologies Inc. (2019). SolarEdge, [Online]. Available: <https://www.solaredge.com/aus/> (visited on 04/02/2019).

- [375] Synergy. (2018). Solar & battery, [Online]. Available: <https://www.synergy.net.au:443/Solar-and-battery> (visited on 04/02/2019).
- [376] I. B. Rybak, D. Wood, J. Murray, A. Janes, and M. Pichette, “Systems and methods for extraction and telemetry of vehicle operational data from an internal automotive network,” US9659417B2, May 2017.
- [377] M. Amadeo, C. Campolo, and A. Molinaro, “Information-centric networking for connected vehicles: A survey and future perspectives,” *IEEE Communications Magazine*, vol. 54, no. 2, pp. 98–104, Feb. 2016, ISSN: 0163-6804. DOI: 10.1109/MCOM.2016.7402268.
- [378] J. E. Siegel, D. C. Erb, and S. E. Sarma, “A Survey of the Connected Vehicle Landscape—Architectures, Enabling Technologies, Applications, and Development Areas,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 8, pp. 2391–2406, Aug. 2018, ISSN: 1524-9050. DOI: 10.1109/TITS.2017.2749459.
- [379] M. Amjad, A. Ahmad, M. H. Rehmani, and T. Umer, “A review of EVs charging: From the perspective of energy optimization, optimization approaches, and charging techniques,” *Transportation Research Part D: Transport and Environment*, vol. 62, pp. 386–417, Jul. 2018, ISSN: 1361-9209. DOI: 10.1016/j.trd.2018.03.006.
- [380] N. Shaukat, B. Khan, S. M. Ali, C. A. Mehmood, J. Khan, U. Farid, M. Majid, S. M. Anwar, M. Jawad, and Z. Ullah, “A survey on electric vehicle transportation within smart grid system,” *Renewable and Sustainable Energy Reviews*, vol. 81, pp. 1329–1349, Jan. 2018, ISSN: 1364-0321. DOI: 10.1016/j.rser.2017.05.092.
- [381] Y. Zhou, R. Kumar, and S. Tang, “Incentive-based distributed scheduling of Electric Vehicle charging under uncertainty,” *IEEE Transactions on Power Systems*, vol. 34, no. 1, pp. 3–11, Jan. 2019, ISSN: 0885-8950. DOI: 10.1109/TPWRS.2018.2868501.
- [382] H. Manghani, J. Prasanth Ram, and N. Rajasekar, “An Internet of Things to Maximum Power Point Tracking Approach of Solar PV Array,” in *Advances in Smart Grid and Renewable Energy*, S. SenGupta, A. F. Zobaa, K. S. Sherpa, and A. K. Bhoi, Eds., ser. Lecture Notes in Electrical Engineering, Springer Singapore, 2018, pp. 401–409, ISBN: 978-981-10-4286-7.
- [383] F. Touati, M. A. Al-Hitmi, N. A. Chowdhury, J. A. Hamad, and A. J. R. San Pedro Gonzales, “Investigation of solar PV performance under Doha weather using a customized measurement and monitoring system,” *Renewable Energy*, vol. 89, pp. 564–577, Apr. 2016, ISSN: 0960-1481. DOI: 10.1016/j.renene.2015.12.046.
- [384] G. Hill, P. Blythe, and V. Suresh, “Tracking and managing real world electric vehicle power usage and supply,” in *9th IET Data Fusion & Target Tracking Conference (DF&TT 2012): Algorithms & Applications*, London, UK: IET, 2012, pp. 15–15, ISBN: 978-1-84919-624-6. DOI: 10.1049/cp.2012.0415.
- [385] Tritium. (2019). VEEFIL-RT 50kw DC FAST CHARGER, [Online]. Available: <https://www.tritium.com.au/product/productitem?url=veefil-rt-50kw-dc-fast-charger> (visited on 04/03/2019).
- [386] T. Anegawa, “Development of quick charging system for electric vehicle,” in *Proc. World Energy Congress*, Tokyo Electric Power Company, 2010.
- [387] I. E. Commission, “IEC 62196-3: 2014,” *Plugs, Socket-Outlets, Vehicle Connectors and Vehicle Inlets—Conductive Charging of Electric Vehicles—Part*, vol. 3, 2014.

- [388] ElaadNL. (2019). Elaad NL, [Online]. Available: <https://www.elaad.nl/> (visited on 04/03/2019).
- [389] Google Developers. (2019). Google Visualization API Reference | Charts, [Online]. Available: <https://developers.google.com/chart/interactive/docs/reference> (visited on 04/03/2019).
- [390] Xiamen Four-Faith Communication Technology Co., Ltd. (2018). F2414 UMTS/WCDMA/HSDPA/HSU IP MODEM(DTU), [Online]. Available: <https://en.four-faith.com/f2414-wcdma-ip-modem.html> (visited on 04/03/2019).
- [391] NXP Semiconductors. (2018). MIFARE DESFire EV1, [Online]. Available: <https://www.mifare.net/en/products/chip-card-ics/mifare-desfire/mifare-desfire-ev1/> (visited on 04/03/2019).
- [392] Astra Telematics. (2018). AT110 GPS-GPRS vehicle tracking device, [Online]. Available: <https://gps-telematics.co.uk/products/at110-gps-gprs-fleet-management-applications/> (visited on 04/03/2019).
- [393] ——, (2018). AT240 3g UMTS / GNSS IP67 waterproof vehicle tracking device, [Online]. Available: <https://gps-telematics.co.uk/products/at240-waterproof-vehicle-tracking-device/> (visited on 04/03/2019).
- [394] u-blox AG, “EVA-M8 series,” u-blox, Data sheet UBX-160007405, R02, Jul. 2016, p. 32.
- [395] J. Yiu, “ARMv8-M Architecture Technical Overview,” ARM, White Paper, Nov. 2015, p. 16.
- [396] jQuery. (May 2019). jQuery JavaScript Library. original-date: 2009-04-03T15:20:14Z, [Online]. Available: <https://github.com/jquery/jquery> (visited on 05/01/2019).
- [397] D. Vanderkam. (Apr. 2019). Interactive visualizations of time series using JavaScript and the HTML canvas tag: Danvk/dygraphs. original-date: 2009-11-24T10:26:21Z, [Online]. Available: <https://github.com/dankv/dygraphs> (visited on 05/01/2019).
- [398] J. J. Guy. (Mar. 2019). Python module to create heatmaps. original-date: 2012-09-18T02:02:44Z, [Online]. Available: <https://github.com/jjguy/heatmap> (visited on 04/04/2019).
- [399] S. S. T. AG, “Sunny Tripower 5000tl-12000tl,” SMA Solar Technology AG, Datasheet, May 2017, p. 6.
- [400] ——, “Sunny Webbox,” SMA Solar Technology AG, Datasheet, 2018.
- [401] Asial Corporation. (2018). JpGraph, [Online]. Available: <https://jpgraph.net/> (visited on 04/04/2019).
- [402] J. Mullan, D. Harries, T. Bräunl, and S. Whitely, “The technical, economic and commercial viability of the vehicle-to-grid concept,” *Energy Policy*, vol. 48, pp. 394–406, Sep. 2012, ISSN: 0301-4215. doi: 10.1016/j.enpol.2012.05.042.
- [403] ClimateWorks Australia, “The Path Forward for Electric Vehicles in Australia,” ClimateWorks Australia, Melbourne, Australia, Tech. Rep., Apr. 2016, p. 36.
- [404] Statista. (2018). Worldwide number of battery electric vehicles in use from 2012 to 2016 (in 1,000s), [Online]. Available: <https://www.statista.com/statistics/270603/worldwide-number-of-hybrid-and-electric-vehicles-since-2009/> (visited on 02/02/2018).

- [405] The REV Project. (Dec. 2017). Electric Vehicle DC Fast-Charging Station, [Online]. Available: <http://therevproject.com/trials/dc-charging-trial.php> (visited on 02/02/2018).
- [406] RAC WA. (2018). RAC Electric Highway, [Online]. Available: <http://electrichighway.rac.com.au/> (visited on 02/02/2018).
- [407] F. Gebauer, R. Vilimek, A. Keinath, and C.-C. Carbon, “Changing attitudes towards e-mobility by actively elaborating fast-charging technology,” *Technological Forecasting and Social Change*, vol. 106, pp. 31–36, May 2016, ISSN: 0040-1625. DOI: 10.1016/j.techfore.2016.02.006.
- [408] J. Bailey, A. Miele, and J. Axsen, “Is awareness of public charging associated with consumer interest in plug-in electric vehicles?” *Transportation Research Part D: Transport and Environment*, vol. 36, pp. 1–9, May 2015, ISSN: 1361-9209. DOI: 10.1016/j.trd.2015.02.001.
- [409] IEEE, “IEEE Standard Technical Specifications of a DC Quick Charger for Use with Electric Vehicles,” *IEEE Std 2030.1.1-2015*, pp. 1–97, Feb. 2016. DOI: 10.1109/IEEEESTD.2016.7400449.
- [410] S. Speidel and T. Bräunl, “Leaving the grid—The effect of combining home energy storage with renewable energy generation,” *Renewable and Sustainable Energy Reviews*, vol. 60, pp. 1213–1224, Jul. 2016, ISSN: 1364-0321. DOI: 10.1016/j.rser.2015.12.325.
- [411] R. M. Dell, P. T. Moseley, and D. A. J. Rand, *Towards Sustainable Road Transport*. Elsevier LTD, Oxford, Aug. 2014, 345 pp., ISBN: 0124046169.
- [412] J. Martínez-Lao, F. G. Montoya, M. G. Montoya, and F. Manzano-Agugliaro, “Electric vehicles in Spain: An overview of charging systems,” *Renewable and Sustainable Energy Reviews*, vol. 77, pp. 970–983, Sep. 2017, ISSN: 1364-0321. DOI: 10.1016/j.rser.2016.11.239.
- [413] International Electrotechnical Commission. (Feb. 2017). IEC 61851-1:2017, [Online]. Available: <https://webstore.iec.ch/publication/33644> (visited on 02/02/2018).
- [414] Standards Association of Australia, Joint Technical Committee EL/001, Standards Australia Limited, and Standards New Zealand, “Electrical installations (known as the Australian/New Zealand wiring rules),” Standards Australia, Report, 2018, OCLC: 1052794055.
- [415] Tritium. (2015). Veefil-Electric vehicle fast charger instruction manual, [Online]. Available: <https://fccid.io/2AFHX-TRI935001US/User-Manual/Users-Manual-3059907.pdf> (visited on 02/02/2018).
- [416] Q. Wang, B. Jiang, B. Li, and Y. Yan, “A critical review of thermal management models and solutions of lithium-ion batteries for the development of pure electric vehicles,” *Renewable and Sustainable Energy Reviews*, vol. 64, pp. 106–128, Oct. 2016, ISSN: 1364-0321. DOI: 10.1016/j.rser.2016.05.033.
- [417] K. Bullis. (Dec. 2013). Electric Vehicles Out in the Cold, [Online]. Available: <https://www.technologyreview.com/s/522496/electric-vehicles-out-in-the-cold/> (visited on 02/02/2018).

- [418] Open Charge Alliance. (2018). OCPP 1.6, OCPP, Protocols - Open Charge Alliance, [Online]. Available: <http://www.openchargealliance.org/protocols/ocpp/ocpp-16/> (visited on 02/02/2018).
- [419] Australian Government. (2019). Green Vehicle Guide Home, [Online]. Available: <http://www.greenvehicleguide.gov.au/> (visited on 04/23/2019).
- [420] S. A. Birrell, D. Wilson, C. P. Yang, G. Dhadyalla, and P. Jennings, "How driver behaviour and parking alignment affects inductive charging systems for electric vehicles," *Transportation Research Part C: Emerging Technologies*, Technologies to support green driving, vol. 58, pp. 721–731, Sep. 2015, ISSN: 0968-090X. DOI: 10.1016/j.trc.2015.04.011.
- [421] K. A. Kalwar, M. Aamir, and S. Mekhilef, "Inductively coupled power transfer (ICPT) for electric vehicle charging – A review," *Renewable and Sustainable Energy Reviews*, vol. 47, pp. 462–475, Jul. 2015, ISSN: 1364-0321. DOI: 10.1016/j.rser.2015.03.040.
- [422] Charging Interface Initiative e. V. (Jan. 2018). CCS Specification, [Online]. Available: <http://www.charinev.org/ccs-at-a-glance/ccs-specification/> (visited on 02/02/2018).
- [423] F. Lambert. (Dec. 2017). BMW and Porsche join forces to enable 15-min electric car charging at 450 kW charge rate, [Online]. Available: <https://electrek.co/2017/12/05/bmw-porsche-electric-car-charging-450-kw-charge-rate/> (visited on 02/02/2018).
- [424] M. Kane. (Dec. 2017). FastCharge Now Evaluating 450 kW Charging, [Online]. Available: <https://insideevs.com/fastcharge-now-evaluating-450-kw-charging/> (visited on 02/02/2018).
- [425] J. S. Johansen, "Fast-Charging Electric Vehicles using AC," PhD thesis, Technical University of Denmark, Kongens Lyngby, Denmark, 2013.
- [426] Z. Ji and X. Huang, "Plug-in electric vehicle charging infrastructure deployment of China towards 2020: Policies, methodologies, and challenges," *Renewable and Sustainable Energy Reviews*, vol. 90, pp. 710–727, Jul. 2018, ISSN: 1364-0321. DOI: 10.1016/j.rser.2018.04.011.
- [427] C. Steitz, "Plug wars: The battle for electric car supremacy," *Reuters*, Jan. 2018.
- [428] P. Morrissey, P. Weldon, and M. O'Mahony, "Future standard and fast charging infrastructure planning: An analysis of electric vehicle charging behaviour," *Energy Policy*, vol. 89, pp. 257–270, Feb. 2016, ISSN: 0301-4215. DOI: 10.1016/j.enpol.2015.12.001.
- [429] K. 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. 01, Jul. 2015, pp. 191–198.
- [430] K. Kaur and G. Rampersad, "Trust in driverless cars: Investigating key factors influencing the adoption of driverless cars," *Journal of Engineering and Technology Management*, vol. 48, pp. 87–96, Apr. 2018, ISSN: 0923-4748. DOI: 10.1016/j.jengtecman.2018.04.006.
- [431] X. Han, H. Wang, J. Lu, and C. Zhao, "Road detection based on the fusion of Lidar and image data," *International Journal of Advanced Robotic Systems*, vol. 14, no. 6, p. 10, Nov. 2017, ISSN: 1729-8814. DOI: 10.1177/1729881417738102.

Appendix A

Robust Deep Learning System for Multi-Sensor Road Detection

Applications of machine learning algorithms have become commonplace in automotive research, particularly the use of techniques such as deep neural networks for complex image processing tasks including object identification and tracking. In this article we describe a hardware accelerated, end-to-end deep learning solution for the most fundamental driving safety task; road keeping. We use an Autonomous Formula-SAE vehicle equipped with multiple LiDARs, high resolution camera and NVIDIA Jetson TX1 compute platform. We examine the issues surrounding reliability of such an algorithm and how this is mitigated by using an array of sensors to provide redundancy of raw information, as well as presenting a training and testing methodology

A.1 Introduction

The Renewable Energy Vehicle Project (REV) operates a compact Formula SAE race car as a platform for hardware-in-the-loop solution development (see Fig. A.1). The system features a modular software structure utilising Robot Operating System (ROS) to maintain functions of perception, path planning, mapping and control and is practicable for driving in structured and unstructured driving environments.

Recently, autonomous driving is a topic of much interest in the public domain, with commercial vehicles accessible to the public expected at the beginning of the next decade [429]. The technological advances in communications, controls and embedded systems have created an environment conducive with the implementation of fully autonomous solutions, however, one of the main barriers to the adoption of autonomous cars on the roads is public

trust [430]. Research into driverless technology should address the factors of performance expectancy, reliability and security as they are now the key factors influencing driverless car adoption [430].

The priority of road detection is to ensure safe driving and has been commonly implemented using LiDAR or cameras. A key outcome in recent applications is the shift toward the use of sensor fusion to obtain a larger dataset. LiDAR is effective at detecting objects and generating an accurate hypothesis for their position, while vision is used as a classifier responsible for validation or final classification of objects. The result of combining LiDAR and camera data prior to processing has provided improvements in the false positive rate [431]. An end-to-end or sensor-to-control implementation utilising machine learning will introduce efficiencies in the processing of data via a Deep Neural Network [129].



Fig. A.1 UWA-REV Autonomous SAE-Electric car.

A.2 Problem Description

Road detection is a vital pillar in autonomous driving and care must be taken to ensure a robust approach for all autonomous applications in order to maintain public expectations. The single sensor based approaches previously adopted does not utilise the data to its full extent and is inferior to an approach where sensor fusion is applied [431]. LiDAR is not strongly affected by environmental conditions while vision based methods suffer greatly from variations in light intensity and limited fields of view, but can provide greater performance through identification of a larger feature space. The multisensory approach is capable of utilising the strengths of each sensor, improving the overall robustness.

A.3 Sensor Configuration

The Formula-SAE car features an array of sensors, including two cameras, three LiDARs, a high-performance inertial navigation system as well as odometry and GPS. For road keeping, we focus on using environmental sensing to provide redundancy in case of inaccurate localisation.

A.3.1 Computer Vision

A pair of FLIR Blackfly GigE cameras are mounted on the vehicle's frame to perform tasks related to visual navigation, such as semantic segmentation and visual odometry. These cameras are fitted with wide-aperture varifocal lenses and use 1.3MP 1/3" global shutter CCD image sensors. The cameras are connected via Gigabit Ethernet to the Jetson TX1, interfacing them through Blackfly's ROS node where it is configured to record at 10 Hz per channel.

A.3.2 LiDAR

The vehicle utilises an array of Light Distance and Ranging (LiDAR) systems. This consists of a SICK LMS111-1010, an ibeo LUX 4, and a Velodyne VLP-16 connected via Ethernet to the NVIDIA Jetson TX1. The data is published by each of the LiDARs' ROS drivers and made available for processing within ROS.

The LMS111 is mounted forward-facing on the front of the vehicle and scans a single layer at 50 Hz. The LUX scans four layers at 10 Hz and features in-built object detection and tracking. It is utilised to achieve road edge and object detection at a distance. The VLP-16 scans 16 layers at 20 Hz, and is mounted in the horizontal plane at the top of the vehicle, resulting in scans of the ground at distances from 5 to 45 metres ahead of the vehicle and can be used for LiDAR-based mapping tasks.

A.4 Proposed Pipeline and Algorithms

Our approach uses a hybrid of three components, combining traditional engineering approaches with deep learning methods, to achieve a more efficient and robust system.

A.4.1 LiDAR Road Edge Detection

Road edge-detection with LIDAR can be accomplished by means of either locating the road-body itself, or by location of an edge feature. Currently, both approaches have been tested using novel algorithms such as [271]. The nature of the data lends itself well to processing by CNN which, with sufficient training, will be sensitive to both detection conditions.

A.4.2 Semantic Image Segmentation

We use semantic segmentation to understand and account for the large dynamism in road scenes, such as moving pedestrians and traffic, as well as variations in road types and markings. Semantic segmentation uses a convolutional neural network (CNN) to classify each pixel on an image frame to constitute an object that it represents. It outputs a series of images, each composed from a set of several distinct colours, with each colour representing an object class. To cater for the dynamism of road environments, we further classify these objects into static and dynamic objects. Through the segregation of static and dynamic objects, the system can then direct and curate computation methods for different regions within an image frame. For instance, calculations on the road region with lane markers will yield results pertaining to lane keeping, centring the vehicle as it drives.

A.4.3 End-to-End Deep Learning

We describe an end-to-end CNN architecture which differs from [129] in that it processes the camera input image (50×200 greyscale pixels) in a separate stack from the combined LiDAR input scan (50×200 distance values). Each side runs several convolutional 5×5 kernel layers until both streams get linked together with convolutional and fully connected layers. A final output layer of five neurons defines the desired steering angle through linear interpolation. This architecture is illustrated as Fig. A.2.

In this application we propose an end-to-end solution which accepts both image data and LiDAR point clouds as input to a deep CNN and generates driving command directly. Training data is being collected using the sensor array on the SAE car itself. Depth information provided by different LiDARs is then merged into a 3D point clouds. Then by projecting the point clouds to the surrounding cylindrical plane of the vehicle, the data can be presented as a 2D array, which can be processed computationally efficiently by CNN. The training data is developed with sensor failures and typical anomalies for each sensor integrated into the training set such that the model is robust under failure of each sensor. The controller output is then used for lane-keeping via the ROS controller node.

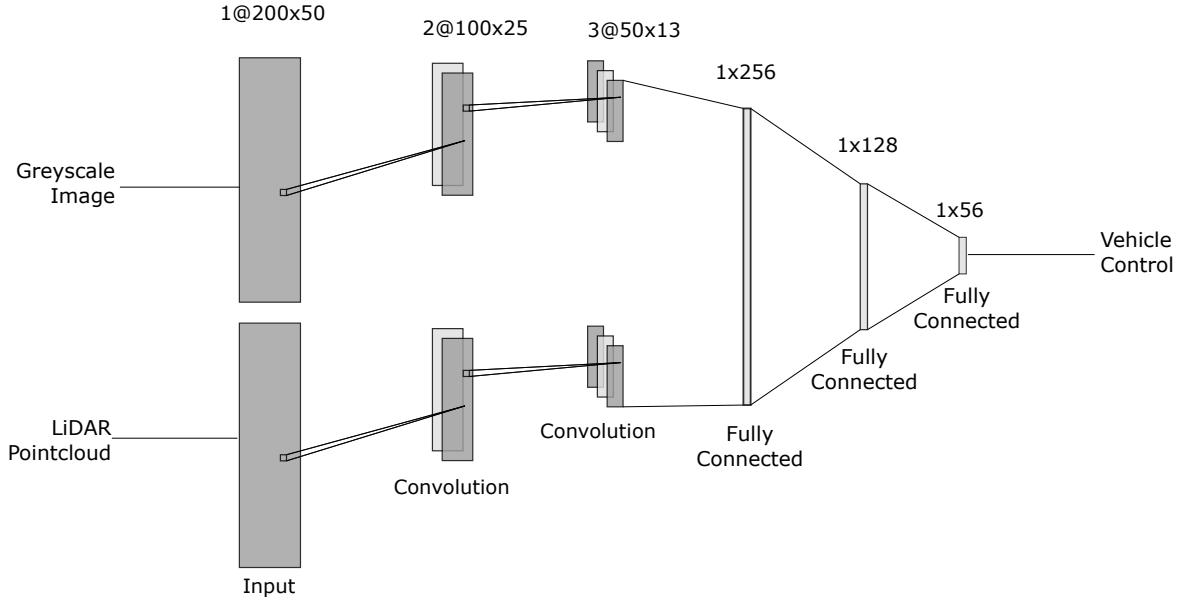


Fig. A.2 Hybrid CNN architecture for camera and LiDAR.

A.4.4 Simulation Framework and Experiments

A hardware-in-the-loop autonomous driving simulation platform centred around the CARLA driving simulator is utilised for both the collection of training data, and early stage testing of the developed algorithms. The use of simulation in the establishment of a training data set allows for improved efficiency in the case of semantic segmentation, with the capability to automatically generate pixel-perfect segmentation masks. It also allows the training set to include a broader range of scenarios than are available to the real hardware platform.

We augment simulation with field testing on roads and poorly defined paths, to prove the performance under varied conditions. The robustness of this approach is proven by demonstration that the redundancy of multiple sensors leads to a fault tolerance and high accuracy suitable to ensure trustworthy road-keeping.

