Deep Learning-based Localisation for Autonomous Vehicles

Dissertation zur Erlangung des Grades eines Doktors der Naturwissenschaften (Dr. rer. nat.)

vorgelegt von

Ricardo Carrillo Mendoza

am

Fachbereich Mathematik und Informatik
Freie Univesität Berlin

Berlin, den 14.Januar.2021

Erstgutachter:
Prof. Dr. Dr. (h.c.) habil. Raúl Rojas
Institut für Informatik
Fachbereich Mathematik und Informatik
Freie Universität Berlin
Arnimallee 7, 14195 Berlin
raul.rojas@fu-berlin.de

Zweitgutachter:
Dr. Erik Valdemar Cuevas Jiménez
Departamento de Computación
Universidad de Guadalajara
Av. Revolución 1500, 44860, Jalisco, México.
erik.cuevas@cucei.udg.mx

Datum der Disputation: 20.11.2020

Abstract

Autonomous driving has become a priority in the research and development division of the automotive industry. According to the required technical and safety demands of the automobile standardisation organizations, localisation plays a crucial role in achieving the maximum level of automation in a vehicle. The use of deep learning and neural networks to develop modules of artificial intelligence has become the preferred tool in disciplines such as computer vision. Moreover, the method excels at learning complicated representations by employing supervised learning or self-supervised learning through techniques such as deep reinforcement learning. In particular, the estimation of complex parameters from images such as depth or optical flow out-perform classical method baselines under constrained settings. The models extract rich information, which is used for tasks such as semantic and instance segmentation, as well as to compute temporal associations between video frames or stereo-pair images. In general, applying these end-to-end deep learning models and finding such associations is complex. This thesis explores the applicability of end-to-end deep learning architectures for vehicle localisation estimation, using either sensory data from dynamical vehicle parameters or camera images. To achieve this, we observed that the net does not need to learn everything from scratch, and we can use associations that we already know about the physical world. We address these ideas using concepts from physics, geometry, and leveraging transfer learning from large-scale regression data using temporal associations.

We also show that autonomous model cars can be used in the process of data collection and that the learned associations can be transferred to other vehicles to improve accuracy.

Moreover, we show how the localisation estimation generalises to other scenes, allowing us to regress the displacement of the vehicle given a sequence of temporal data and compose the global estimated position.

Zusammenfassung

Autonomes Fahren hat in der Forschungs- und Entwicklungsabteilung der Automobilindustrie einen immer höheren Stellenwert erreicht. Um den maximalen Automatisierungsgrad gemäß den erforderlichen technischen und Sicherheitsanforderungen der Automobilstandardisierungsorganisationen zu gewährleisten, spielt die Lokalisierung des Fahrzeugs eine entscheidende Rolle. Die Verwendung von Deep-Learning und neuronalen Netzen zur Entwicklung von Modulen der künstlichen Intelligenz, ist das dominierende Werkzeug in Disziplinen wie Computer Vision geworden. Darüber hinaus zeichnet sich das Deep-LearningVerfahren unter Einsatz von Supervised-Learning sowie neuerdings Self-Supervised-Learning durch Techniken wie z.B. Deep-Reinforcement-Learning aus. Insbesondere übertrifft die Schätzung komplexer Parameter aus Tiefenbildern bzw. optischem Fluss unter eingeschränkten Einstellungen die klassischen Basislinien-Methoden. Die verwendeten Modelle extrahieren umfangreichere Informationen als die reine Bilderkennung und berechnen zeitliche Assoziationen zwischen Videobildern oder Stereopaar-Bildern.

Im Allgemeinen ist die Anwendung dieser End-to-End-DeepLearning-Modelle und das Finden solcher Verbindungen komplex. Diese Arbeit untersucht die Anwendbarkeit von End-to-End-Deep-Learning-Architekturen für die Schätzung der Fahrzeuglokalisierung durch Bereitstellung von Sensordaten aus dynamischen Fahrzeugparametern oder Kamerabildern. Um das zu erreichen, muss das Netzwerk die zugrundeliegenden Beziehungen nicht von Grund auf lernen. Stattdessen werden bekannte Assoziationen der physikalischen Welt miteinbezogen. Dies geschieht durch Konzepte der Physik und Geometrie, sowie Transfer-Learning von Regressionsdatensätzen durch zeitliche Assoziationen. Es wird gezeigt, dass autonome Modellfahrzeuge zur Datengenerierung benutzt werden können. Die zugrundeliegenden Beziehungen können dann auf andere Fahrzeuge übertragen werden, um die Genauigkeit der Standortberechnung zu erhöhen. Außerdem wird gezeigt, dass die Fahrzeuglokalisierung generalisiert werden kann, welches die Ermittlung der globalen Position anhand einer Sequenz zeitlich zusammenhängender Daten erlaubt.

If you really want to escape the things that harass you, what you're needing is not to be in a different place but to be a different person.

— Lucius Annaeus Seneca

Acknowledgements

I want to thank my advisor Prof. Dr. Raúl Rojas, whose passion for science encouraged me to work on this subject. I appreciate his guidance, which was vital for the successful completion of this thesis. I am grateful to him for inviting me to collaborate and extend my research experience at the Freie Universitaet and the Autonomous car project.

I would also like to express my gratitude towards the AutoNOMOS Team at the Freie Universitaet in no particular order: Claas Norman Ritter, Daniel Neumann, Daniel Göhering, Bingyi Cao, Stephan Sudermann, Khaled Alomari, Fritz Ulbrich, Nicolai Steinke, Tobias Langeberg, Xiuyan Guo, and Zahra Boroujeni for the technical discussions and valuable insights into my research. I thank Susanne Schöttker-Söhl, who always gave me the best support and guidance through the university's challenging administrative side.

I especially thank my brothers: Pabel Carrillo, Miguel Ángel Ríos, and Abdi Escalante, who also took the time to review this thesis and gave expert feedback from their expertise areas.

My deepest gratitude to the Mexican team in the group: José Álvarez, Margarita Diaz, Fernando Wario, and Omar Mendoza. I appreciate their advice, suggestions, and friendship.

I thank my dad, my mom, and my sister for their unlimited love, and because besides the distance, they remained close all the time.

Finally, I would like to thank my beloved wife, Karina Sánchez. Her immense support, technical and emotional, helped me get through this challenging period in the most positive way. Thank you for believing in me always and for giving me strength when I need it.

I gratefully acknowledge the funding sources that made my Ph.D. work possible. My research was supported by the Consejo Nacional de Ciencia y Tecnología and the Deutsche Akademische Austauschdienst (DAAD).

Contents

Acronyms

ABS	Anti-lock Braking System
ACC	Adaptive Cruise Control
ADAS	Advanced Driver Assistance Systems
AFS	Active Front Steering
APE	Absolute Pose Error
ASIL	Automotive Safety Integrity Level
ATE	Absolute Trajectory Error
AWS	Amazon Web Services
BARC	Berkeley Autonomous Race Car
BNN	Bayesian Neural Networks
CAN	Controller Area Network
CAS	Collision Avoidance System
CNN	Convolutional Neural Network
DGPS	Dual GPS
DMI	Distance Measuring Indicator
DNN	Deep Neural Networks
DRL	Deep Reinforcement Learning
DSO	Dicrect Sparse Odometry
DTAM	Dense Tracking and Mapping
DVSO	Deep Virtual Stereo Odometry
EKF	Extended Kalman Filter
ESC	Electronic Stability Control
FAST	Features from Accelerated Segment Test
FU	Freie Universitaet
GALNet	Ground Autonomous Localization Net

GLONASS	Global Navigation Satellite System
GPS	Global Positioning System
GPU	Graphics Processing Unit
HARA	Hazard Analysis and Risk Assessment
IARTK	Inertially Assisted Real Time Linematic
IMU	Inertial Measurement Unit
LDS	Laser Displacement Sensors
LDV	Laser Doppler Velocimeters
LDWS	Lane Departure Warning System
LGS	Laser Ground Sensor
LIDAR	LIght Detection And Ranging
LIDAR	Light Detection and Ranging
LSTM	Long Short Term Memory
MIG	Made In Germany
MIT	Massachusetts Institute of Technology
ML	Machine Learning
MPC	Model Predictive Control
MPC	Model Predictive Control
MPPI	Model Predictive Path Integral Controller
MSE	Mean Square Error
ORB	Oriented FAST and Rotated BRIEF
PDF	Probability Distribution Function
PSPNet	Pyramid Scene Parsing Network
PTAM	Parallel Tracking and Mapping
PWM	Pulse Width Modulation
RADAR	RAdio Detection And Ranging
RANSAC	Random Sample Consensus
RMSE	Root Mean Square Error
RNN	Recurrent Neural Networks

ROS	Robotic Operating System
RPE	Relative Pose Error
RTK	Real Time Kinematic
SAE	Society of Automotive Engineers
SAS	Steering Wheel Angle Sensor
SEGNET	SEGmentation NET
SEIF	Sparse Extended Information Filter
SIFT	Scale-Invariant Feature Transform
SLAM	Simultaneous Localization And Mapping
SONAR	SOund NAvigation Ranging
SSD	Solid State Drive
SSIM	Structural SIMilarity
SURF	Speeded-Up Robust Features
SVD	Singular Value Decomposition
UKF	Unscented Kalman filter
UMTS	Universal Mobile Telecommunications System
UVC	Universal Video Class
VALNet	Visual Autonomous Localisation Net
VIN	Visual Inertial Navigation
VO	Visual Odometry
VSA	Vehicle Slip Angle
VSC	Vehicle Stability Control
VSLAM	Visual Simultaneous Localization And Mapping
WO	Wheel Odometry
YOLO	You-Only-Look-Once

Chapter 1

Introduction

The future of mobility based on autonomous vehicles is an audacious visionary goal with highly complex but achievable technical challenges. This work is part of the general solution to the problem of providing pose estimation to driverless cars in urban environments.

The localisation concept is referenced in this work as it is used in the robotics field, i.e. part of a navigation system.

Navigation is defined as a combination of three fundamental competencies:

$$\text{Navigation} \begin{cases} \text{Path planning} \\ \text{Self-localisation} \begin{cases} \text{Odometry} \\ \text{Path integration} \\ \text{Relocalisation} \end{cases} \\ \text{Map-building} \end{cases}$$

Robot navigation refers to the robot's ability to determine its position within its frame of reference and then plan a path towards some goal location. Path planning requires determination of the robot's current position and the goal location, both poses must be within the same frame of reference or coordinates. To navigate in its environment, the vehicle requires representation, i.e. a map of the environment and the ability to interpret that representation. Map building can be defined with a metric map or any notation describing locations in the robot's frame of reference.

Robot localisation denotes the robot's ability to establish its position and orientation within the frame of reference. Moreover, odometry is the use of data from motion sensors to estimate the change in position over time in relation to a previous instantaneous frame, while path integration or dead reckoning is referred as the total displacement of the vehicle from a known starting point.

The objective of this work is to analyze whether Deep Neural Networks (DNN) can improve current localisation methods through odometry by employing available sensors on the car such as camera, wheel speed sensors and inertial units, and test the robustness against known problems in traditional odometry methods.

Localisation is a fundamental requirement for autonomous cars. Vehicles must be able to avoid obstacles by planning safe paths to reach the desired destination in order to operate autonomously. Planning an efficient and safe path typically requires a map.

Furthermore, following a path safely requires the robot to know its precise location on the map at any point in time, i.e., it must be localized. Hence, mapping new environments robustly while being accurately localized is essential for mobile robots. For this reason, odometry and more robust techniques such as Simultaneous Localization And Mapping (SLAM) are highly relevant for uninterrupted robot operation.

Traditional localisation approaches commonly use sensors such as inertial measurement units, global positioning systems, SOund NAvigation Ranging (SONAR), RAdio Detection And Ranging (RADAR), and Light Detection and Ranging (LIDAR). Unavailability of Global Positioning System (GPS) signals in indoor and below-surface environments, unacceptably high while drift using inertial sensors during extended GPS outages, issues of possible confusion with nearby vehicles for SONAR and RADAR, and the line of sight requirement for laser-based systems are some of the limitations associated with these systems. One promising solution lies in the science of visual odometry which estimates motion, taking advantage of helpful information such as texture of scenes with the help of cameras mounted on the vehicle.

The performance of localisation methods based purely on visual information can be affected by photometric calibration, motion bias, rolling shutter effect, or self-moving illusion. In order to reduce the effects of these phenomena on performance, usually, the estimations are jointly mixed with the estimation, of other sensors.

On the field of artificial vision, the impact of deep learning has been transformational, and it is already making significant inroads into traditional robotics, including SLAM.

With current advances with in end-to-end solutions in deep learning on topics such as segmentation, specifically for autonomous driving with nets like You-Only-Look-Once (YOLO)[63], Segnet[4], U-Net[67] or Pyramid Scene Parsing Network (PSPNet)[94], depth estimation with DeMoN[7] or optical flow estimation like FlowNet[24], and occlusion detection with nets like Netdef[35], deep learning approaches are leading perception research. This opens the way to push forward the development of an end-to-end localisation solution which complements or replaces traditional semantic and appearance-based SLAM algorithms.

Currently, in the Dahlem Center for Machine Learning and Robotics, topics such as localisation and path planning are of high interest. To test the algorithms, the institute has three autonomous vehicle projects: Made In Germany (MIG), e-Instein, and AutoMiny, which are shown in Fig. 1.

Deploying Machine Learning (ML) algorithms in autonomous cars which are allowed to drive on streets has regulatory obstacles. The title of ”self-driving car” may seem self-evident, but there are six

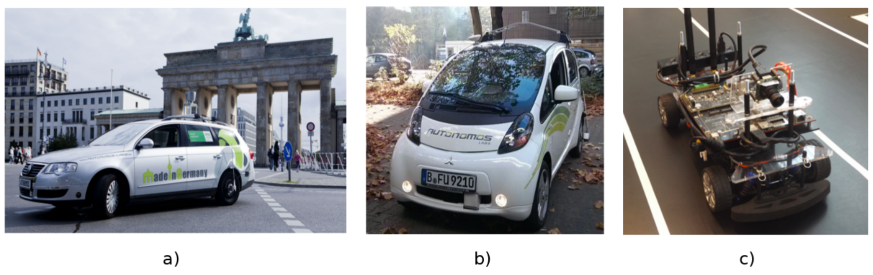


Figure 1: The Autonomous Vehicle Project at the Freie Universitaet Berlin. a) MIG, b) e-Instein, c) AutoMiny.

Society of Automotive Engineers (SAE) Levels used to define autonomous driving. The SAE J3016 standard (SAE Committee, 2014) introduces a scale from zero to five for grading vehicle automation, and techniques such as ML and DNN are leading the race to achieve level five. These automation levels are defined as follows[1]:

- SAE Level 0 (No automation): the full-time performance by the human driver of all aspects of the dynamic driving task, even when enhanced by warning or intervention systems.
- SAE Level 1 (Driver assistance): the driving mode-specific execution by a driver assistance system of either steering or acceleration/deceleration using information about the driving environment and with the expectation that the human driver perform all remaining aspects of the dynamic driving task.
- SAE Level 2 (Partial Automation): the driving mode-specific execution by one or more driver assistance systems of both steering and acceleration/deceleration using information about the driving environment and with the expectation that the human driver perform all remaining aspects of the dynamic driving task.
- SAE Level 3 (Conditional automation): the driving mode-specific performance by an automated driving system of all aspects of the dynamic driving task with the expectation that the human driver will respond appropriately to a request to intervene.
- SAE Level 4 (High automation): the driving mode-specific performance by an automated driving system of all aspects of the dynamic driving task, even if a human driver does not respond appropriately to a request to intervene.
- SAE Level 5 (Full automation): the full-time performance by an automated driving system of all aspects of the dynamic driving

1 The definitions were extracted from https://justauto.nridigital.com/just_auto_magazine_apr19/sae_levels_of_driving_automation

task under all roadway and environmental conditions that can be managed by a human driver.

Table 1 shows on which tasks the autonomous system or the human driver intervene according to the level of automation.

	Steering and Acceleration/ Deceleration	Monitoring of Driving Environment	Fallback Performance of Dynamic Driving Task	System Capability (Driving Modes)
Human driver monitors the driving environment				
0	Human	Human	Human	n/a
1	Human & System	Human	Human	Some driving modes
2	System	Human	Human	Some driving modes
Automated driving system monitors the driving environment				
3	System	System	Human	Some driving modes
4	System	System	System	Some driving modes
5	System	System	System	All driving modes

Table 1: SAE J3016 Levels of Driving Automation

Regardless of the empirical definitions on the five automation scales and possible interpretations of safety, the use of deep learning components in safety-critical systems is still not clear. The ISO 26262 safety standard for road vehicles gives a list of requirements to ensure safety. Nevertheless it does not address the unique characteristics of deep learning-based software.

Salay et al. [69] addresses this uncertanty analyzing the areas where machine learning can be implemented and how the standard would be impacted. The autor provides recommendations on how to manage this impact. The recommendations are focused on identifying hazards, and implementing tools and mechanisms for fault and failure situations. It is necessary to ensure the correct contruction of training datasets and design multi-level architechtures. The standarisation of algorithms along the different stages on the software development life-cycle is mandatory.

The standard ISO 26262 suggest the use of a Hazard Analysis and Risk Assessment (HARA) method to identify hazardous events in the system, and specify safety goals that mitigate the hazards. The standard is composed of ten parts:

- Part 1: Vocabulary
- Part 2: Management of functional safety
- Part 3: Concept phase
- Part 4: Product development at the system level
- Part 5: Product development at the hardware level
- Part 6: Product development at the software level
- Part 7: Production and operation
- Part 8: Supporting processes
- Part 9: ASIL-oriented and safety-oriented analysis
- Part 10: Guideline on the safety standard

Automotive Safety Integrity Level (ASIL) refers to a risk classification scheme defined in ISO 26262 for an item (e.g. subsystem) in an automotive system. In this norm, a hazard is defined as a ”potential source of harm caused by a malfunctioning behaviour, where harm is a physical injury or damage to the health of a person” [72].

A deep learning module in the vehicle can potentially generate new types of hazards. An example of a dangerous scenario happens when the reliability of an automated driver assistance (often developed using learning techniques) is overrated by a human driver [60].

A deep learning module may have hunderds of thousands of parameters, and its due to its complexity, that can fail in unexpected ways. For example, in deep reinforcement Learning systems, false assignations of the reward function can conduct to a negative vehicle behaviour [3]. In such a case, the automated vehicle deduce that it can avoid getting penalized for driving too close to other vehicles by performing a dangerous driving behaviour or by exploiting particular sensor vulnerabilities to bypass how close it is getting. Despite hazards such as these may be unique to deep reinforcement learning components, they can be monitored to faults, thus fitting within the existing guidelines of ISO 26262.

An essential requirement for analyzing the safety of deep learning components is to examine whether the immediate human costs of outcomes exceed the humanitary harm severity thresholds. Undesired outcomes are harmful in a human sense, and their effect takes place in real-time. These unexpected reposnses can be classified as safety issues.

Self-driving vehicles must have fail-safe phisical mechanisms, usually named Safety Monitors. These must stop the autonomous control software once a failure is detected [44]. Specific fault types and failures have been catalogued for neural networks in [29, 34, 47]. Those failures led to the development of specific and focused tools and techniques to help find faults. [13] describes a technique for debugging misclassifications due to inadequate training data and a troubleshooting fault that detects complex interactions between linked machine learning components is proposed in [58].

In order to reduce the testing time and make all test comply with the mentioned norms, while also reducing danger by a third and the vehicle itself ML, we employ a developed scale autonomous car. This work will show how to take advantage of this kind of platforms for localisation matters.

1.1 Thesis Statement

It is possible to develop a deep neural network model to estimate vehicle localisation in urban environments using either wheel velocity and Inertial Measurement Unit (IMU) data or monocular camera images. Since it is difficult for safety and regulatory reasons to perform aggressive driving manoeuvres while driving on public streets, training data from scaled car-like robots can be used as a data augmentation method. The learned associations in the scaled vehicle can be transferred to a full-scale car learned model to improve the results in real-driving situations.

1.2 Thesis Contributions

The contributions of this work are in the fields of localisation and deep neural networks for autonomous driving. They are summarized as follows:

- Development of the scaled robotic architecture AutoMiny to collect training data and test networks on an NVIDIA™arm-architecture Graphics Processing Unit (GPU)
- Development of a neural network architecture called Ground Autonomous Localization Net (GALNet) to estimate localisation given kinematic and dynamic data from the vehicle
- Development of a Siamese deep neural network architecture for visual localisation named Visual Autonomous Localisation Net (VALNet) to estimate ego-motion given a monocular frame sequence

1.3 Thesis Structure

The outline of the thesis is as follows. Chapter 2 gives an overview of the current classical methods and neural networks used to estimate localisation for autonomous cars given dynamics and camera images. The principal drawbacks of these classic methods are discussed, and the advantages of neural networks are highlighted as well as the difficulties of implementing them on an autonomous car. In Chapters 3 and 4 we present a method to estimate motion for dynamics and camera images respectively; each chapter presents how the datasets were made, the architecture and designed loss functions, as well as the experiments and a discussion of results. In Chapter 5 we make overall conclusions, discuss the applicability of this technology, and suggest directions for future research.

Chapter 2

Research Platforms and State of the Art

In this chapter, we present the vehicles employed for this work, and the localisation approaches currently being used in them.

A brief description of the traditional methods based on kinematic-dynamic sensors and cameras used in autonomous cars is presented. Its neural network approach follows each method. The chapter highlights the drawbacks of each method and how neural networks can improve them as well as the opportunities on DNN schemes.

In the localisation methods based on kinematic-dynamic sensors, approaches using wheel speed sensors and IMUs are highlighted with an emphasis on Ackermann steering and slip angle compensation. On the visual localisation methods, a general overview of the best solutions and their inherent drawbacks which make the localisation problem an open research subject are described.

The chapter clarifies why the development of a new platform is necessary to improve the localisation methods available in an MIG vehicle.

2.1 Autonomous Vehicles at the Freie Universitaet

In the following subsections, we present the hardware set-up of the autonomous vehicles used for this work as well as their current localisation systems. In the University we have two autonomous cars and one autonomous model car, each one wholly developed at the robotics department of the Freie Universitaet. Chapters 3 and 4 explain how these platforms work together to get better results in the Visual Odometry (VO) system developed in this work.

2.1.1 *The Autonomous Car MadeInGermany*

The autonomous vehicle project at the Intelligent Systems and Robotics Group of the Freie Universitaet Berlin, Germany started in the year 2006. Since then, the efforts have been focused on the research and development of Advanced Driver Assistance Systems (ADAS). Regarding this project, the AutoNOMOS Labs company was formally founded in 2009, and the autonomous car platform MIG was built. The MIG is based on a Volkswagen Passat Variant 3C. In 2011, also an electrically powered car was equipped for autonomous purposes using a Mitsubishi i-MiEV, its name is `e-Instein`. In this section, just the MIG will be taken to explain the configuration of the system; in general, the sensorial hardware is similar on the e-Instein. The MIG is equipped with drive-by-wire technology and several different sensors allowing the car to be controlled by the AutoNOMOS Software framework which runs on computers connected to the car. Due to extensive simulation and testing of driving scenarios at the former airport Tempelhof, it was possible to obtain in 2011 a special permit to drive the MIG autonomously on public roads in real traffic. However, the safety concept requires monitoring by a human safety driver and a safety observer. The MIG drove a thousand kilometres autonomously since then, including a 2400 km long-distance drive from the USA to Mexico City. Figure 2 illustrates the MIG and the arrangement of some of its sensors. The sensors addressed mainly in this thesis are the SatCam RGB colour cameras, the Applanix POS LV and the Controller Area Network (CAN) bus data.

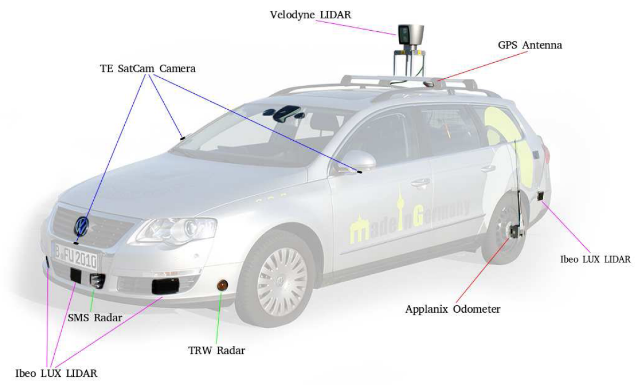


Figure 2: Overview of the Sensor Positioning on MIG autonomous vehicle.

The estimation of the vehicle's position is obtained from an Applanix™POS-LV 520; it calculates the position and orientation fusing information from the integrated inertial technology, a Distance Measuring Indicator (DMI) attached to the rear left wheel and a differential GPS.

The CAN bus provides useful data related to the status of the car, such as:

- Wheel speeds
- Car speed
- Steering wheel sensor
- Wheel Odometry
- Brake pedal
- Gas pedal

There are four TE SatCam RGB colour cameras integrated into the chassis of the car, Figure 2 shows the camera set-up, they are located in the front, in the back and one in each exterior rear mirror on the sides of the car. The cameras are connected to the car via BroadR-Reach automotive Ethernet and communicate with the AutoNOMOS Software framework which is based on Robotic Operating System (ROS) through the camera driver. Table 2 list the essential characteristics of the camera system integrated into the MIG[1].

CAMERA FEATURE	CHARACTERISTIC
Imaging Sensor	OV10635 Family
Horizontal Field Of View (hFOV) [deg]	190
Vertical Field Of View (vFOV) [deg]	118
Output Frame Rate [fps]	30
Spatial Output Resolution [px]	1280 x 800
Output File Format	8-bit RGB JPG
Shutter	Rolling Shutter

Table 2: SatCam Camera features

More information about the autonomous car is available on the Autonomous GmbH website[2] or in the Intelligent Systems and Robotics Research Group web page in the Department of Mathematics and Computer Science at the Free University Berlin.

2.1.1.1 *Implemented Localisation Methods.*

The Applanix system, besides being expensive, it has inherent the drawbacks of the base sensors on which is based and the uncertainties of the filtering method used to fuse the information of them. Among the most important drawbacks, are the ones related to the Dual GPS (DGPS) signal, which in some places, like tunnels, the signal quality decreases, or die. The Inertially Assisted Real Time Linematic (IARTK) based on the IMU data, the odometry estimation based on the DMI and the Real Time Kinematic (RTK) obtained from a Universal Mobile Telecommunications System (UMTS) connection, define the Applanix estimation error which is defined in Table 3 according to the specifications sheet.

Moreover, in practice, the measured precision of the Applanix installed in the MIG, was obtained in Robert Spangenberg Doctoral thesis [73], in his work it is mentioned that for a small trajectory the average positional uncertainty, measured by the size of the ellipsoid error, is $(0.22m, 0.35m)$ for the small and big sides of the ellipse.

Accuracies below 0.1 m are usually only reached during vehicle standstill. Good accuracies can only be reached if ten or more satel-

1 Based on specifications provided by the component suppliers `https://www.ovt.com/sensors/OV10635`

2 `autonomos-systems.de/en/`

lites are available. In typical urban scenarios with limited satellite visibility and multi-path effects, the uncertainty figures of the system can be misleading.

In order to construct an idea of the Applanix system accuracy, its drift was plotted in a repeatability test while driving the car in the University campus (Fig. 3). The given localisation jumps in the range of 0.2m from time to time and provokes sudden quality levels changes (float and fixed RTK) which affect real-time kinematics. When the vehicle is parked, jumps in the position information were also reported.

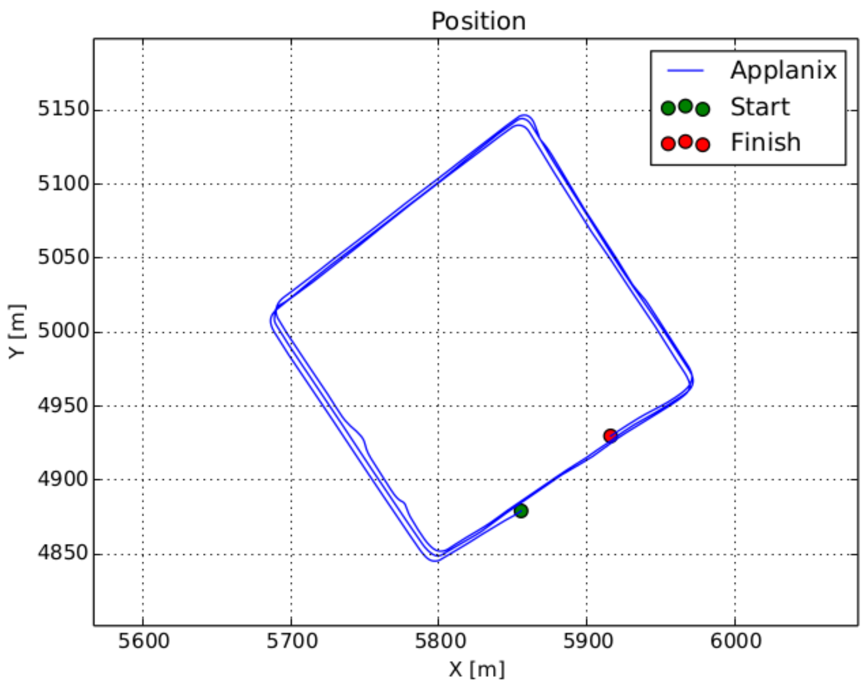


Figure 3: Applanix positional drift while driving on the University campus.[73]

Therefore, the convenience of the DGPS for accuracy estimates is limited; the position information must carefully be analysed when used as a positional reference and is not used as ground truth to evaluate the VO algorithms.

POSITION	RTK	IARTK	DGPS
X,Y Position(m)	0.020	0.035	0.30
Z Position (m)	0.050	0.050	0.500
Roll and Pitch (deg)	0.005	0.005	0.005
True Heading (deg)	0.015	0.020	0.020

Table 3: Applanix Performance specifications [3]

In the Intelligent Systems and Robotics Research Group of the Freie Universitaet (FU) Berlin in 2015, the first localisation method without using Applanix was proposed in Robert Spangenberg Dissertation thesis [73]. On this work a stereo vision-based localisation was used, the features tracked for it were pole-like landmarks, the localisation method is based on particle filters. The overview of the system is shown in Fig. 4

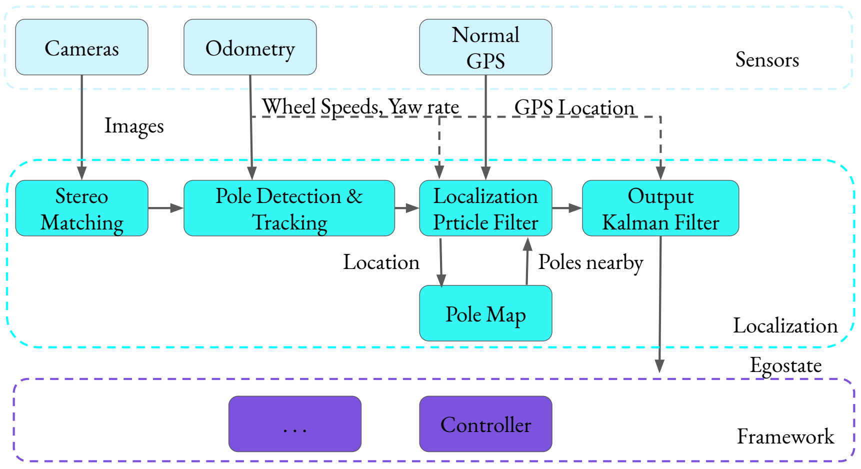


Figure 4: Localization scheme of Robert Spangenberg based on stereo vision and pole detection.[73]

On this approximation, the camera image is used for the disparity computation while, the odometry information is used for pole tracking, the state of the particle filter and the Kalman filter output. The simple GPS location fixes are only used for the initialisation of the particle filter. Map information is queried by the particle filter, using its current best estimate. The output Kalman filter achieves 100 Hz frequency of the state and transmits its results to the autonomous car framework to provide the Egostate.

In 2017 Xiuyan Guo [31] proposed a similar localisation scheme using the LIDAR as the main sensor. The method is a two-point feature-based localisation using a filtering technique with extended Kalman filter to fuse it with wheel odometry. The work is different from the classical EKF localisation scheme in the way of handling the measurement update stage. The system structure is shown in Fig. 5. The localisation module consists of two Extended Kalman Filters. The localisation EKF is the main unit of estimation. The smoothing EKF is employed to smooth the estimated trajectory in real-time. Then an Egostate is created to provide the vehicle state information to other modules.

There are some areas of improvement on the presented localisation methods in order to obtain better accuracy:

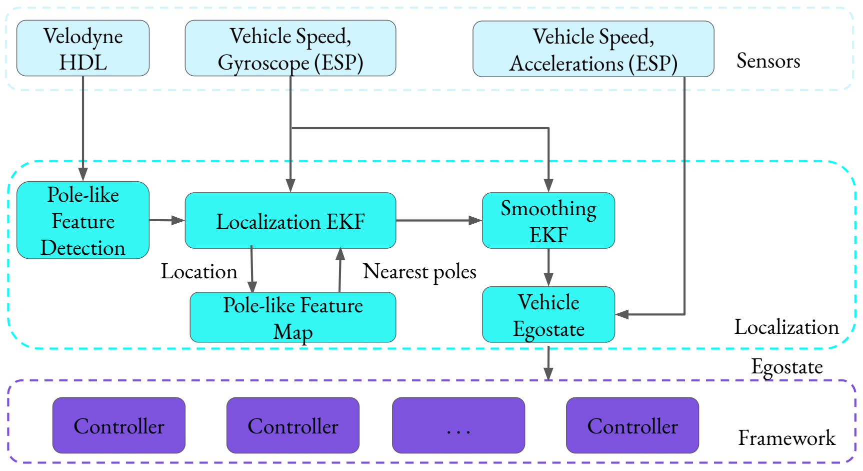


Figure 5: Localization scheme of Xiuyan Guo based on Two-point features LIDAR-based.[31]

Landmarks: The methods are landmark-dependent; therefore, if the map has a lack of trees or poles, the uncertainty grows, therefore is necessary to develop other robust-feature detectors. These methods also are affected by the drawbacks of feature detectors, such as blur, occlusions, high brightness, and issues related to camera calibration and quality of the camera.

SLAM: Since the accuracy of the methods relies entirely on the accuracy of the build map, modules such as loop closure or posterior optimisation to evaluate the consistency of the map is also necessary..

2.2 Kinematic-Dynamic Localisation Approaches

Wheel odometry in robotics as been widely used mainly to compute the prior estimation of more complex odometry estimators or to give the vehicle information by way of emergency when other methods are not available.

There are several mechanical, physical, geometrical and mathematical assumptions or generalisations which demotivate the use of wheel Odometry on severe applications like the principal source of estimation. Despite the drawbacks, it is popular among the roboticist to calculate the position of a wheeled vehicle, the sensors used to get this information are relatively cheap, and already mature in the development and manufacturing process which makes them almost failure-free; those sensors are also widely displayed in the cars since 1987.

In the following sections, the perturbations which affect this method will be explained together with the advantages and disadvantages of the most common methods; this will lead to the idea of having a smart system that can approximate a function to handle the dynamic and kinematic parameters involved on the estimation.

2.2.1 Classical Methods

In this category Odometry methods based on the kinematics of the car are considered, approaches like differential Odometry or Ackermann steering are widely the most employed classical methods in the literature, some examples are [6, 71, 80, 88]. The dynamic parameters of the vehicle, specifically wheel parameters such as rotational velocity or tyre force, usually are used to develop stabilisation control systems and ADAS.

The reason to use this method as a secondary way to calculate the robot position is in its mechanical nature; variable wheel size, different tyre materials, spring effects due to the suspension system, diverse steering and traction mechanisms. Furthermore, some indirect external agents cause wheel odometry to be imprecise, for instance, road conditions, driving habits, weather, variable vehicle weight, components failure, and wear. Therefore, it is necessary to improve the mathematical model or depend continuously on the data from other sensors like cameras, IMU or GPS.

The Ackermann approach is derived from the geometrical assumption that the vehicle has a four-bar steering mechanism and the car moves in perfect circles which centre I is localised on the rear wheel axis and in the intersection point of the perpendicular projections from the pointing wheel direction (see Fig.6).

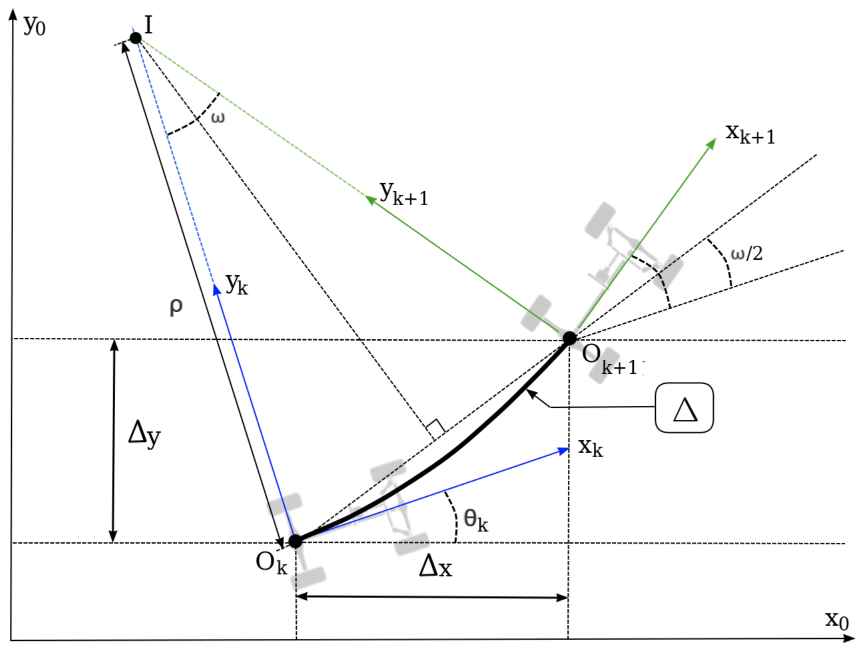


Figure 6: Ackermann steering geometry on a global frame. *I* is the instantaneous center of rotation between the car frames O_k and O_{k+1}, Δ is the traveled distance and ρ is the radius.

Moreover, in the models based on Ackermann, the mathematical approach assumes that the arc on which the car moves between origins, can be approximated up to the second-order as $\Delta = |O_k O_{k+1}|$, taking the following equations to calculate displacement with Ackermann kinematics derived from Fig.6:

$$\begin{aligned} x_{k+1} &= x_k + \Delta\cos(\theta_k + \omega/2) \\ y_{k+1} &= y_k + \Delta\sin(\theta_k + \omega/2) \\ \theta_{k+1} &= \theta_k + \omega \end{aligned} \tag{2.1}$$

Where x, y and θ are the 2D *x, y* and *yaw* coordinates of the car in different k instants of time, Δ is the travelled distance, and ω is the angle between coordinate systems of the car on times k and $k+1$ related to the instantaneous centre of rotation.

In order to calculate Δ and ω the lectures of the wheel encoder should be used. There are diverse methods to calculate the variables related to the kinematics of the car, in the most basic approach, the differential odometry can be calculated as follows:

$$\Delta = \frac{\delta_{RR} + \delta_{RL}}{2} \tag{2.2}$$

$$\omega = \frac{\delta_{RR} - \delta_{RL}}{2e} \tag{2.3}$$

where δ is the linear displacement of the wheel in meters, the sub-indexes *RR* and *RL* correspond to the Rear Right wheel and the Rear Left wheel and *e* is the half-track of the car (see Fig. 7).

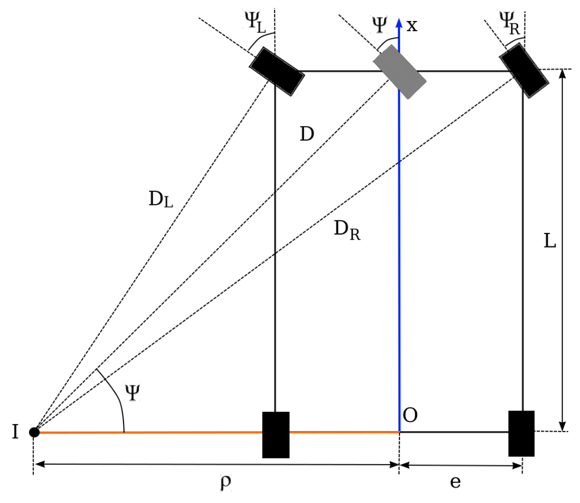


Figure 7: Geometry of the car related to the mobile frame *O*. Ψ is the Ackermann angle which is referenced on a middle virtual wheel, ρ is the radius, *e* is the half track and *L* the wheel-base of the vehicle.

Differential odometry would be enough, but the first dynamic parameter, tyre slip, makes this approach not adequate for practical use. Looking for extra kinematic constraints, the Ackermann angle ψ can improve the results, in order to add the circle movement constraint, is it possible to calculate the radius from Fig.6 as follows:

$$\begin{aligned}\rho &= \frac{\delta_{RL}}{\omega} + e \\ &= \frac{\delta_{RR}}{\omega} - e\end{aligned} \tag{2.4}$$

Ackermann angle is then included on the equation list with the radius or the displacement:

$$\begin{aligned}\tan(\Psi) &= \frac{L}{\rho} \\ &= L\frac{\omega}{\Delta}\end{aligned} \tag{2.5}$$

At this point it is theoretically possible to compute an Ackermann odometry using the steering angle sensor of the car to calculate ω with Eq. (2.5) and integrating the velocity measurements of each wheel from the Anti-lock Braking System (ABS) sensors to calculate Δ with Eq. (2.2).

Nonetheless, no information of the front wheels has been taken into count. They can be included analyzing Fig.6 further. As the orientation between the wheels with respect to the mobile frame is not constant, the displacement from time k to $k+1$ of the virtual wheel in Fig.7 is not Δ, it will be represented as Δ_F, with D representing the distance to the instant point of rotation, the following relation can be obtained:

$$\Delta_F = D\omega$$

and since

$$D = \frac{L}{\sin(\psi)}$$

using (2.4), it leads to:

$$D = \frac{\rho}{\cos(\psi)}$$

By multiplying each side by ω:

$$\Delta_F \cos(\psi) = \Delta$$

Repeating the procedure for each front wheel, is possible to get the following equations:

$$\Delta_{FL} \cos(\psi_L) = \Delta - e\omega$$

$$\Delta_{FR} \cos(\psi_R) = \Delta + e\omega$$

With the previous equations, a non-linear and redundant system to compute Δ and ω with the sensor lectures of the wheel ABS sensors (Δ_{FL}, Δ_{FR}, Δ_{RL}, Δ_{RR}) and steering sensor (ψ) variables can be written:

$$\begin{aligned} \tan(\psi) &= L\frac{\omega}{\Delta} \\ \Delta_{RL} &= \Delta - e\omega \\ \Delta_{RR} &= \Delta + e\omega \\ \Delta_{FL}\cos(\psi_L) &= \Delta - e\omega \\ \Delta_{FR}\cos(\psi_R) &= \Delta + e\omega \end{aligned} \tag{2.6}$$

In order to solve this system, classical estimators or optimisers can be used, in [6] an Extended Kalman Filter (EKF) is implemented, although this approximation takes into count front-wheel measurements, in a vehicle the effective half-track e and the wheel-base L are not constant due to external disturbances like the suspension system, the wheel contact area with the floor which modifies the tyre force and wear of the materials. Additionally, in practice, the Ackermann steering geometry of the mechanism can be changed to affect the dynamic settings of the car, therefore, each car may have different relations between the Ackermann angles of the front wheels.

To physically measure the Ackermann angle in the car, the Steering Wheel Angle Sensor (SAS) of the car must be mapped between the wheel position and the wheel Ackermann angle. Giving an analytical solution to the Ackermann-linkage or steering mechanism of the vehicle is not a convenient answer for generalisation, this means, the mapping may not work for another vehicle since the mechanical parameters are hardly the same.

In the case of AutoMiny, the problem of converting steering commands to actual Ackermann angles is more evident since the toe angle (the symmetric angle that each wheel makes with the longitudinal axis of the vehicle), the caster angle (the angular displacement of the steering axis from the vertical axis of a steered wheel) or the camber angle (from a top view, the angle between the vertical axis of the wheels used for steering and the vertical axis of the vehicle) differ on each car, not to mention the adjustable linkages and manufacturing differences between them.

On full-scale cars, some wheel configurations as the anti-Ackermann, where the inner wheel has less steering angle than the outer wheel, may be used to get more grip on the inner wheel in the curves. Fig.8 shows the Ackermann-linkage of the AutoMiny and the steering mechanism used on the MIG are shown. The relationship between the steering sensor and the Ackermann angle can be described by a function $\Psi_{L/R}(\theta)$.

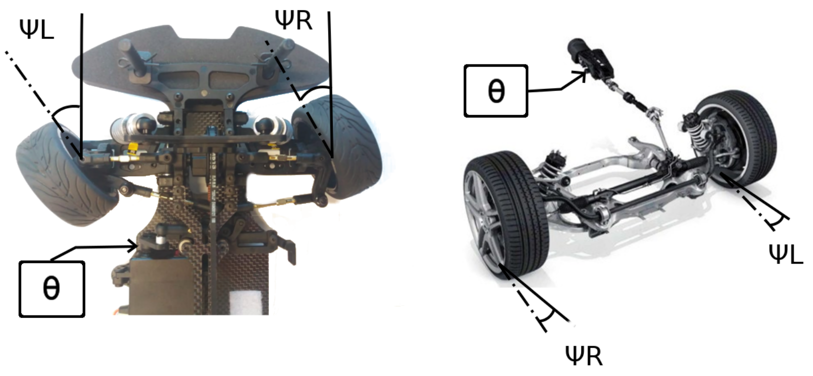


Figure 8: Left: Ackermann-linkage on the chassis of Autominy, right: a common rack and pinion steering system. $\Psi_{L/R}$ are the Ackermann angles of the left and right wheel and Θ is the Steering Wheel Angle measured by the SAS.

It is possible to construct an approximation for $\Psi_{L/R}(\Theta)$ through a lookup table taking samples of the Ackermann angles calculated with Eq. 2.6 and the SAS. In the literature like in [23, 39], the approximation function holds just for a known working acceleration and velocity threshold, defined by the user. On Fig.9 a four-degree curve was fitted using the data of the AutoMiny scaled vehicle, the SAS is normalized, i.e. $\theta \in [-1, 1]$.

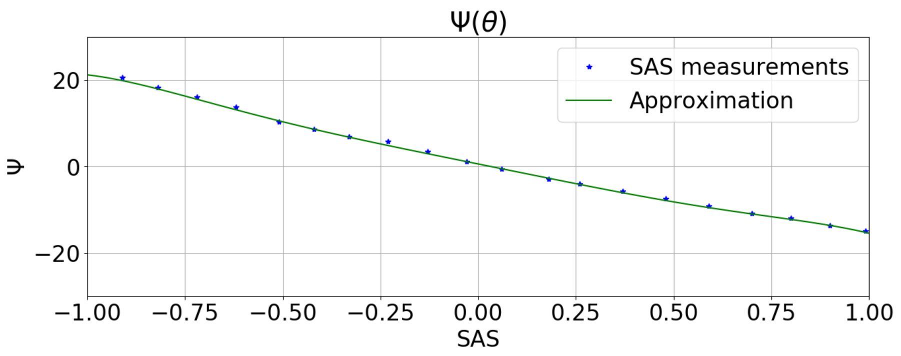


Figure 9: Fitted curve for the mean Ackermann angle function $\Psi(\theta)$ of the AutoMiny chassis

Notwithstanding that the mathematical structure of the Ackermann principle is simple, the tyre force is not taken into count, tyre force tears down the concept of Fig.7 since the wheels do not move on their heading direction.

A more precise concept of odometry must then take into count the Vehicle Slip Angle (VSA). VSA, also known as the drifting angle is "the angle between the vehicle longitudinal axis and the direction of travel, taking the centre of gravity as a reference" [16]. The VSA is the ratio between the actual direction of a wheel and its pointing direction, as shown in Fig. 10.The VSA is calculated as follows:

$$\beta = -\arctan\left(\frac{V_y}{V_x}\right) \tag{2.7}$$

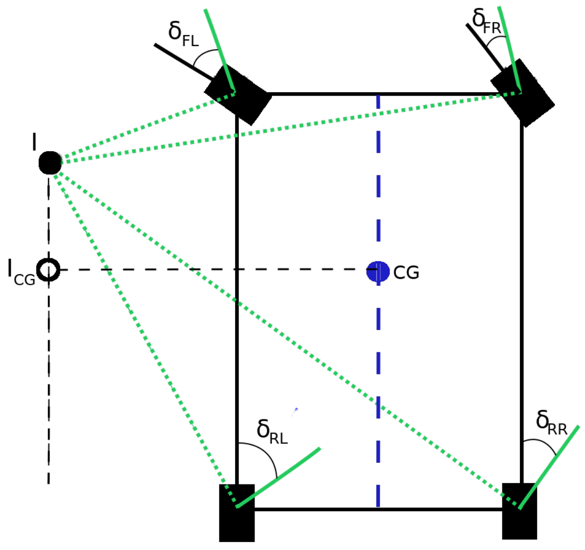


Figure 10: Slip angle of each wheel of the vehicle: $\delta_{RL}, \delta_{RR}, \delta_{FL}, \delta_{FR}$, the instantaneous point of rotation I and the desired rotation point I_{CG} related to the center of gravity CG.

The VSA is widely used for stability controllers such as Electronic Stability Control (ESC) [92] or Vehicle Stability Control (VSC) [27], Active Front Steering (AFS) [5], Model Predictive Control (MPC)[93], among others. Typically for such applications, an accuracy of 0.1 degrees is needed. The stability controllers regulate the tracking force of each wheel employing the brakes in the vehicle, in this way, is possible to have a point of rotation I as close to the perpendicular projection of the centre of gravity as shown on Fig.10. Therefore to the means of localisation, having such controllers help to ensure that the vehicle will have a common rotation centre among the four wheels of the vehicle, with this valid assumption is possible to formulate a new estimation of the trajectory.

Obtaining the VSA is a difficult task since by the time this thesis is written, there is only one type of industrial onboard sensor available that can measure it directly. The Laser Ground Sensor (LGS) [4] measures the ground speed with Laser Doppler Velocimeters (LDV), tyre rotation and with Laser Displacement Sensors (LDS) the tyre radius is measured in real-time. The calculation of the wheel slip angle can be performed with Eq. 2.7. However, this kind of sensors is being used only for research or tunning purposes due to the needed external montage which needs to be supervised constantly, see Fig.11.

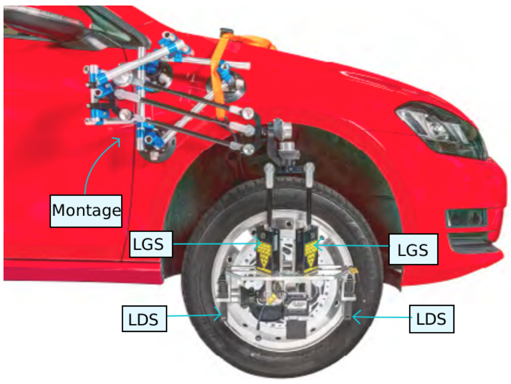


Figure 11: Laser ground sensor to directly measure wheel slip angle.[5]

Another method to obtain VSA is to estimate it through GPS-inertial sensors. There are several methods to compute such estimation. In [41] the slip angle is related to inertial measurements from the bicycle model as follows:

4 https://aanddtech.com/lgs/#tab-id-1

5 Photo taken from: https://aanddtech.com/wp-content/uploads/2018/06/LGS-datasheet-20170418.pdf

$$\begin{aligned}\dot{\beta} =& \frac{1}{mv_g}\left(f_{xF}\sin(\Psi-\beta) - f_{xR}\sin(\beta) + f_{yF}\cos(\Psi-\beta)\right)\\ &+ \frac{1}{mv_g} f_{yR}\cos(\beta) - \dot{\psi}\end{aligned} \tag{2.8}$$

or

$$\dot{\beta} = \frac{a_x}{v_g} - \dot{\psi}$$

where *m* stands for the vehicle mass, $v_g = \sqrt{v_x^2 + v_y^2}$ is the velocity of the vehicle defined with the magnitude of the longitudinal and lateral velocity of the car, Ψ is the steering angle, $\dot{\psi}$ is the yaw rate, $F_i j$ with $i = x, y$, $j = F, R$ are the tyre ground forces applied into the longitudinal (x) and lateral (y) directions at the front (F) and rear (R) axle positions and a_x is the lateral acceleration.

In order to calculate the tyre forces on equation 2.8 is necessary to build a system of equations since the tyre force depends on the slip angle as well. Among the variety of solutions to calculate the tyre force, the most commonly used method is the Pacejka's "magic tyre formula"[59], which can model a tyre force by:

$$f_{ijk} = -\frac{v_{ijk}}{v_g}\mu_{ij} f_{ijz} \quad i = L, R; \quad j = F, R; \quad k = x, y, \tag{2.9}$$

where f_{ijz} is the average load on the corresponding tyre, it can be calculated from [83], μ_{ij} is the total friction coefficient related to each tyre given by the characteristic curve on [59], and v_{ijk} is the relative velocity, not angle as is sometimes used, of each tyre with respect to the road, along with the longitudinal and lateral directions:

$$v_{ijx} = \frac{V_{ijx} - \omega_{ijx} R_j}{\omega_{ijx} R_j} \quad v_{ijy} = \frac{V_{ijy}}{\omega_{ijx} R_j} \tag{2.10}$$

Equation 2.9 can be used on 2.8 and integrate to obtain the slip angle of the vehicle. Some other approaches to estimate the VSA based on observers are summarised in [2, 16, 38, 50, 91].

2.2.2 *Methods with Neural Networks*

Since direct measurement of the vehicle side-slip angle is inconvenient for the reasons explained the last section. Estimating VSA is the most convenient approach. GPS and IMU equipment is easy to integrate into the vehicle. Nevertheless, the estimations do not gather information about different road conditions and inherit the problems of the sensors such as bias (which drifts the estimation), stability (due to temperature), misalignment, latency, among others.

To deal with the drawbacks of the employed sensors, methods with neural networks has been proposed. The main idea is that the system can learn the extrinsic parameters of the estimation and correct it in the prediction. First efforts to integrate a learning agent to the estimations, fuse the two methods, the hybrid estimator use an observer (typically an EKF) which estimates the dynamic of the car while the neural network estimates the tyre data [1, 19].

Most of the authors use a general approach of three-layered neural network, the input layer, one hidden layer and the output layer, first and second layer use log-sigmoid transfer function and a linear activation on the last one [9, 49, 87](see Fig.12).

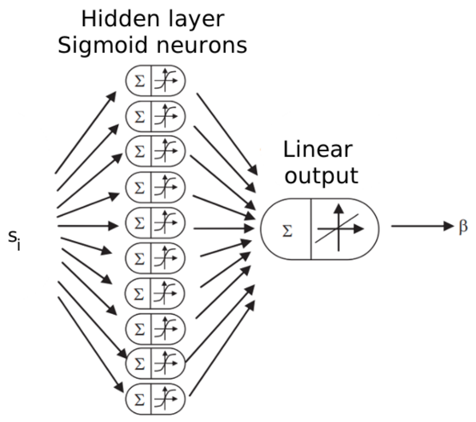


Figure 12: Neural network architecture to estimate vehicle slip angle (β), s_i is the state input.

Melzi et al. [52] mention as relevant to the training, the following characteristics:

- Maneuvers: the database should contain clockwise and counter-clockwise manoeuvres, in fact, selecting the driving scenarios are key to better generalisation.
- Friction: at least two friction conditions (high-low) have to be considered in the training set.
- Velocity: manoeuvres have to be carried at different speeds to count the effect of it on the vehicle while turning.
- Acceleration: there should be one manoeuvre with high longitudinal acceleration in the training set.

Different state inputs to the net are evaluated, the neural network which was able to estimate a better VSA takes into count the change on dynamic parameters respect to time, the input state s_i of the net is:

$$\begin{aligned} s_i = [&v_x(t_i), v_x(t_i - 4\Delta_t), v_x(t_i - 8\Delta_t), \\ &\dot{\psi}(t_i), \dot{\psi}(t_i - 4\Delta_t), \dot{\psi}(t_i - 8\Delta_t), \\ &a_y, \delta] \end{aligned} \tag{2.11}$$

where v_x is the longitudinal speed, a_y is the lateral acceleration, $\dot{\psi}$ is the yaw speed, δ is the steering angle, t_i is the instantaneous time and Δ_t is the interval of time since the last measure.

Besides the VSA estimator with neural networks shows to be accurate enough to be used in practical applications, the main issue up to today remain in the inability of the net to adapt after the training if the vehicle parameters change and the option to deal with a road banking angle which has to be estimated with an external algorithm and then filter out the lateral acceleration component due to gravity as in [53, 70].

However, the use of VSA with neural networks has not been used to perform odometry estimation in an end-to-end neural network, and including other sensor data, as this work proposes.

2.3 Visual Localisation Approaches

2.3.1 *Classical Methods*

The SLAM research done over the past years has itself developed very accurate visual-inertial odometry with small drift (<0.5% of the trajectory length for some dataset like the KITTI odometry dataset [25]). VO is considered a reduced version of a SLAM system. To clarify this, first is necessary to classify the different existing approaches, up to today, the methods can be generalised in two big groups: sparse feature-based methods and dense/direct methods.

On Fig.13a, the principal components of sparse feature-based methods are shown. Sparse methods start by extracting features from the frame of the video employing a descriptor such as Features from Accelerated Segment Test (FAST), Speeded-Up Robust Features (SURF), Oriented FAST and Rotated BRIEF (ORB), Scale-Invariant Feature Transform (SIFT) among others less popular extractors in order to get characteristics of the environment with information about the colour of the pixel, position related to the image coordinate system or even the texture of it. These systems use outlier rejection algorithms like RANSAC in order to keep a good track of the features among the frames of the sequence. The most challenging task about this approach is to correctly match the features and compute the scale estimation if only a monocular camera is being used.

texture *refers to the spatial arrangement of the colours or intensities in an image.*

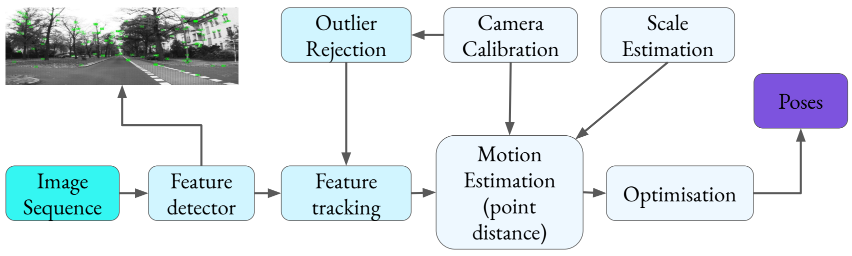
In order to convert the VO approach to a VSLAM navigation system, the modules are shown in Fig.13a must be added. The feature map for drift correction is done along with the estimation. Davison et al. propose the firs Monocular real-time Visual Simultaneous Localization And Mapping (VSLAM) estimating both the map and ego-motion in the framework of EKF [18], this framework which became rapidly the most used on the scientific community with some variants using Unscented Kalman filter (UKF)[51], Sparse Extended Information Filter (SEIF)[77] or particle filter[78] is called filtering-based. The most representative system in this classification of navigation systems is ORB-SLAM[56] which shows excellent performance in various scenarios.

A different approach is the well known keyframe-based VSLAM; this methodology separates the camera tracking and the feature mapping into different procedures; the most accurate method based on this approach is Parallel Tracking and Mapping (PTAM)[43].

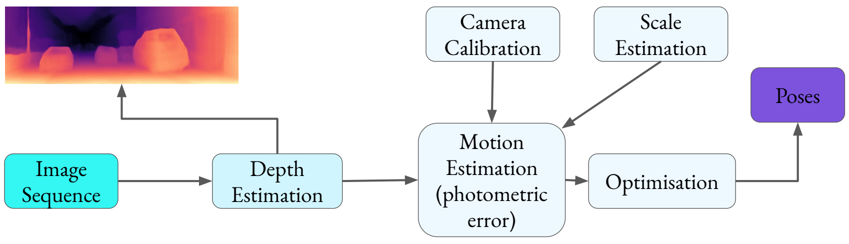
Either filtered-based or keyframe-based, are still feature-based methods, and therefore, sensitive to outliers originated by image motion blur, high brightness, appearance similarity, occlusions, among other factors. Further, this kind of methods does not use all of the information available in the image: this is the reason why dense/direct methods have been proposed.

On Fig.13b the main modules of a dense/direct method are shown. This approach resides on the assumption of photometric consistency. This kind of methods overcome the featured-base methods in large-scale environments, some examples are Dense Tracking and Mapping (DTAM)[57] and LSD-SLAM [21], which uses keyframe-photometric alignment and pose graph SLAM to estimate poses and build the point cloud maps.

The main drawback of this method is that the accuracy of the photometric alignment is severely affected when the baseline of two matching images is long, due to an increased number of local minima under log baseline[22].



(a) Sparse feature methods.



(b) Dense/direct methods.

Figure 13: Visual Odometry methods.

On Fig.14 the most important modules of a complete SLAM system are shown. A SLAM system includes two main components: front-end and back-end SLAM, the first extracts the sensor data into feasible models for the estimation, while the second performs inference on the abstracted data of the front-end. Loop closure detection is an important method to verify the coherence of the built map and associated positions. In case of system failure, relocalisation methods which solve the so-called kidnapped robot problem to give a new first estimate to the SLAM system.

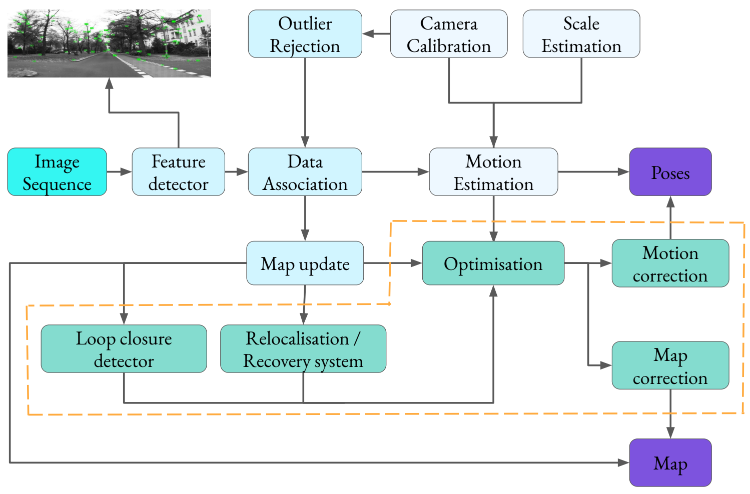


Figure 14: Full SLAM system based on sparse feature visual odometry methods. Modules inside the orange dashed line are called back-end SLAM and the remaining modules are called front-end SLAM.

Nevertheless SLAM algorithms have been proven to be the most accurate methods to estimate the position of a robot fusing information of more than one sensor, there are in general crucial drawbacks that must be addressed:

- Failsafe SLAM and Recovery: Despite the progress made on the SLAM back end, current SLAM solvers are still vulnerable in the presence of outliers; this is mainly since virtually all robust SLAM techniques are based on iterative optimisation of nonconvex costs. This has two consequences: first, the outlier rejection performance depends on the quality of the initial guess of the optimisation; second, the system is not robust: the inclusion of a single outlier diminishes the quality of the estimate, which in turn decreases the capability of rejecting outliers later on.
- Robustness to HW Failure: While addressing hardware failures might appear outside the scope of SLAM; these failures impact the SLAM system and the latter can play a key role in detecting and mitigating sensor and locomotion failures. If the accuracy of a sensor degrades due to malfunctioning, off-nominal conditions or ageing, the quality of the sensor measurements (e.g., noise, bias) does not match the noise model used in the back end, leading to poor estimates. In the case of cameras, brightness, occlusions, blur and hard weather conditions like snow and rain are essential difficulties on the feature recognition modules.

- Relocalization: While appearance-based, as opposed to feature-based, methods are able to close loops between day and night sequences or between different seasons, the resulting loop closure is topological. For metric relocalisation (i.e., estimating the relative pose concerning the previously built map), feature-based approaches are still the norm; however, current feature descriptors lack sufficient invariance to work reliably under such circumstances.

- Automatic Parameter Tuning: SLAM systems (in particular, the data association modules) require extensive parameter tuning to work correctly for a given scenario. These parameters include thresholds that control the quality of the feature matching, some approaches utilise algorithms as Random Sample Consensus (RANSAC) to decide when to add new factors to the graph or when to trigger a loop closing algorithm to search for matches, but they also work in controlled environments.

We refer to the reader of this work to Cesar Cadena et al. [11] to have a more detailed overview of the SLAM research in the scientific community over time.

SLAM is formulated as a maximum a posteriori estimation problem which can be also formulated in a formalism of *factor graphs*[46] to reason about the interdependence among variables. In order to estimate the set of poses $\mathcal{X} = (x_0, x_1, ..., x_n)$ where $x_n = (x_n, y_n, z_n, \theta_n, \phi_n, \psi_n)$ is a vector in the $\mathbf{SE}(3)$ group, is necessary to have a set of measurements $\mathcal{Z} = z_k : k = 1, ..., m$ such that each measurement can be expressed as a function of $\mathcal{X}$ i.e., $z_k = h_k(\mathbf{X}_k) + \epsilon_k$ where $\mathcal{X}_k \subseteq \mathcal{X}$ is a subset of the variables, $h_k(\cdot)$ is a know function constructed with a model of the measurement or observation model and ϵ_k is the measurement noise.

In map estimation, $\mathcal{X}$ is estimated calculating the assignment of variables $\mathcal{X}^*$ that reach the maximum of the posterior belief over $\mathcal{X}$ given the measurements ($p(\mathcal{X}|\mathcal{Z})$).

$$\mathcal{X}^* \doteq \underset{\mathcal{X}}{\arg\max}\ p(\mathcal{X}|\mathcal{Z}) = \underset{\mathcal{X}}{\arg\max}\ p(\mathcal{Z}|\mathcal{X})p(\mathcal{X}) \tag{2.12}$$

If the measurements $\mathcal{Z}$ are independent (the noises are uncorrelated) is possible to factorize the problem like follows:

$$\mathcal{X}^* \doteq \underset{\mathcal{X}}{\arg\max}\ p(\mathcal{X}) \prod_{k=1}^{m} p(z_k|\mathcal{X}_k) \tag{2.13}$$

where the Gaussian measurement likelihood with mean measurement noise zero and with the information matrix Ω_k is:

$$p(z_k|\mathcal{X}_k) \propto \exp\left(-\frac{1}{2}||h_k(\mathcal{X}_k) - z_k||^2_{\Omega_k}\right) \tag{2.14}$$

This problem can be solved via successive linearisation, e.g., the Gauss-Newton or the Levenberg-Marquardt methods. Nevertheless, non-linear filters have shown better accuracy and efficiency than linear methods.

2.3.2 *Methods with Neural Networks*

In recent years, neural networks and more specifically, deep learning applied on computer vision has gained significant attention due to its performance in learning capability and other characteristics such as robustness to camera parameters and challenging environments. These data-driven methods have successfully learned new feature representations from images that are used to improve the motion estimation further; some examples of them are [15, 35, 64].

Deep learning-based VO methods are recent, the first documented attempt that shows it is possible to compute it from a learning agent are [66] and [65], in those investigations, the researchers divide each frame into cells and compute afterwards an average optical flow for each block, a K-Nearest Neighbor (KNN) regressor is trained in [66] and an Expectation Maximisation (EM) algorithm in [65]. In Constante et. al. [17] the interframe pose between two consecutive images is regressed trough convolutional neural networks effectively replacing the standard geometry of visual odometry, being the first considered a deep neural network approach.

Likewise, it is possible to localise the 6DoF of a camera with regression forest from a single image [81] and with deep convolutional neural network like PoseNet in the work of Kendall et. al. in [40].

These approaches divide the current research on end-to-end deep neural networks in two groups, the first, tries to estimate the ego-motion using high-dimensional feature maps such as optical flow, dense map estimation or segmentation, which can be pre-trained or fully trained like in [85]. The second group relies on object detection/classification pre-trained nets; these last methods work as a similar keyframe-based VO system. Thus the open question is whereas a neural network must learn to calculate ego-motion relative to the differences of the image sequences or to calculate the position of the camera relative to the position of the objects on the image. Analogically to the traditional systems, the second mentioned method would be considered more an appearance-based relocalisation method than an odometric algorithm.

The used loss function on the previous methods is an L2 norm euclidean distance.

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\|f(x_i) - y_i\|_2 \tag{2.15}$$

where N is the number of samples, $f(x_i)$ is the estimation of the net and y_i is the ground truth from the database.

Fig. 15 resumes the two most popular end-to-end deep neural network architectures for ego-motion estimation and PoseNet. In the architecture of **CNN-4b VO**, Contante et. al. [17] proposed to divide the image in four quadrants, each quadrant goes trough two convolutional-pooling layers which are trained to penalise the total variation of the flow field by minimising the energy function proposed by Brox et. al. [10], the obtained four feature maps are then feed to a full connected layer in order to obtain the incremental displacement $\Delta\mathcal{X} \in \mathbf{SE}(3)$.

In Wang et. al. [85], the **DeepVO** net uses Long Short Term Memory (LSTM) layers to take advantage of the information carried out by the two consecutive frames from the input. This architecture estimates the relative ego-motion and the parameters of the sparse covariance matrix, which measures the error on the net predictions related to each degree of freedom.

PoseNet generates training labels with structure from motion and uses pre-trained deep neural networks dependent on massive labelled image datasets as an architectural base and refines the training to predict the estimated camera pose $\Delta\mathcal{X} \in \mathbf{SE}(3)$. The PoseNet is a relatively standard deep convolutional network and it has a less complex architecture than FlowNet.

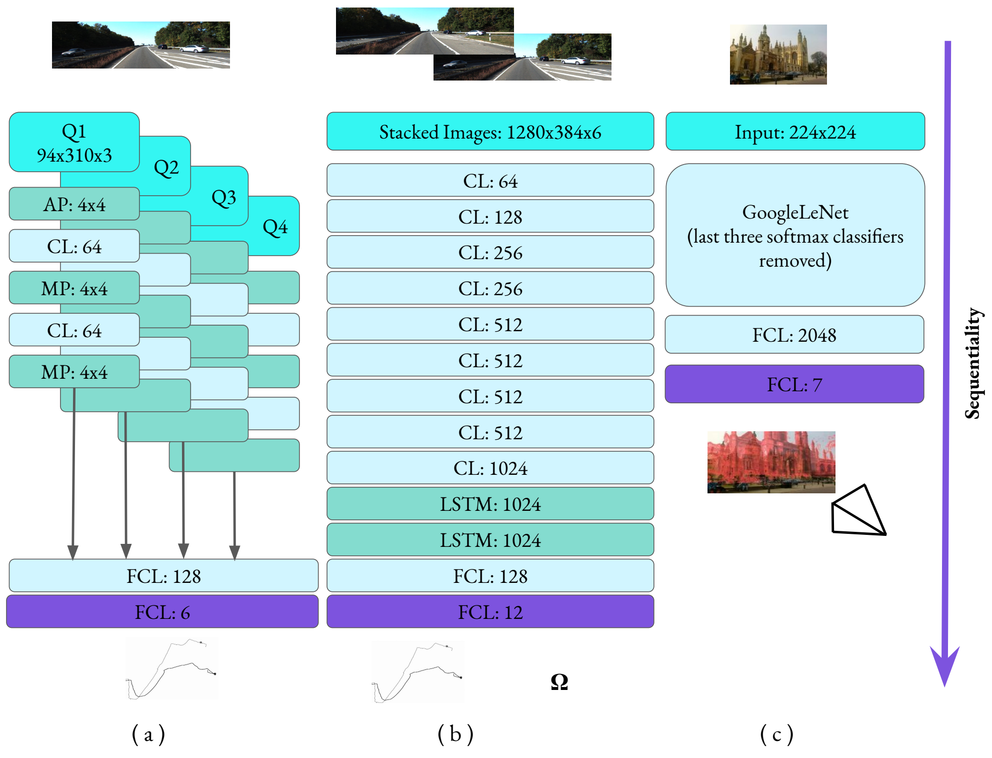


Figure 15: From left to right: **CNN-4b VO**, **DeepVO** and **PoseNet**. Qx are the quadrants of the image, AP is an average pooling layer, CL is a convolutional layer, MP is a max pooling layer, LSTM is a long short term memory layer and FCL is a full connected layer. (a) and (b) predict the ego-motion, but (b) predicts the covariance $\boldsymbol{\Omega}$ as well, (c) predicts the camera pose estimation.

However, in the methods mentioned above, the ground truth of camera poses and the frames associated must be known in advance. In most recent approaches, Ruihao Li et. al. [48] propose an unsupervised deep learning scheme which relays the calculation of the ego-motion with a loss function defined on spatial and temporal dense information. The net loss is designed combining the photo-metric, and the pose consistency, the case of an image pair from a stereo camera and two consecutive frames from a monocular camera are considered. The photometric error is calculated as follows:

$$\begin{aligned} L^{a}_{pho} &= \lambda_s L^{SSIM}(I_a, I'_a) + (1-\lambda_s) L^{l_1}(I_a, I'_a) \\ L^{b}_{pho} &= \lambda_s L^{SSIM}(I_b, I'_b) + (1-\lambda_s) L^{l_1}(I_b, I'_b) \end{aligned} \tag{2.16}$$

where $a = l, k$, $b = r, k+1$ the left and right image for stereo camera or the two consequent monocular frames, $\mathcal{L}^{l_1}$ is the L1 norm operation, L^{SSIM} is the Structural SIMilarity (SSIM) index as calculated in [95] with weight λ_s.

The pose consistency loss is computed as follows:

$$\mathcal{L}_{pos} = \lambda_p L^{l_1}(x'_a, x'_b) + \lambda_0 Ł^{l_1}(\phi'_a, \phi'_b) \tag{2.17}$$

where λ_p is the left-right/subsequent frame position consistency weight, λ_o is the left-right/subsequent frame orientation consistency weight and $[x'_a, \phi'_a]$ and $[x'_b, \phi'_b]$ are the predicted poses for each image respectively.

Thus, this approach predicts the depth map, which is used to predict the ego-motion.

Another similar approach is Deep Virtual Stereo Odometry (DVSO) [20] which integrate just the deep depth predictions into Dicrect Sparse Odometry (DSO)[84] as direct virtual stereo measurements. Whereas this method cannot be considered a full end-to-end VO network, shows how deep neural networks can replace the geometrical modules of actual dense/direct VO schemes. This approach makes clear that the perception is already good enough to get a full end-to-end deep neural net if the regression module is improved.

Deep neural network approaches for VO face several drawbacks and not explored solutions which are listed below:

1. `Regression`: In deep learning is not yet clear if current schemes can overcome the traditional methods for regression, specially for time series forecasting problems. The main problem is the vanishing gradient problem which is an inherent difficulty founded on neural networks with gradient-based learning methods and backpropagation. Despite some special nets like Recurrent Neural Networks (RNN), LSTM, have been developed in order to fight with this problem, it is not clear yet what is the correct path to have the same performance as in classification problems. Univariate forecasting cases are easily tackled with the actual set-up. However, sequence problems with multivariate forecasting as VO are still a challenge, LSTM layers have to be enormous (with more than 1024 units)to learn associations between the sequences and therefore training time for a net the size of **DeepVO** net can take up to 2 days in an NVIDIA RTX 2080Ti GPU. Recently has been demonstrated that a Convolutional Neural Network (CNN) can also learn sequential multivariate forecasting, in Vaswani et al. [82] the `attention-based recurrent networks` were introduced, this approach relies on a convolutional encoder-decoder structure which can dramatically drop the units for training. However, this approach has been tested for VO only in simulation [62]

2. `SLAM modules`: Current research works on developing the most important modules for an SLAM system with DNN such as loop closure [14], re-localization [40] and graph optimization with graph neural networks [68]. Is still an open topic to fit all those efforts together with neural network-base VO in order to produce a full end-to-end SLAM navigation system.

3. `Online and life-long learning:` A big and significant challenge is that of online learning and adaptation, that will be essential to any future long-term SLAM system. SLAM systems typically operate in an open-world with continuous observation, where new objects and scenes can be encountered.

   It is true that currently there are many companies who work on building HD maps for autonomous driving like Tomtom™, CYIENT™or HERE™, and those HD maps store high valuable features which can be used for localisation such as poles, trees and traffic lights. Nevertheless, the problem of life-long maps also adds a monumental work for the companies to keep their maps updated in real-time in order to prevent an accident. This problem can be tackled with a full SLAM system with the ability to assimilate new changes in their environment.

   Even with the upcoming unsupervised learning methods, the net should be able to add new training data if the error increases, for which is mandatory to measure the covariance of the estimation and the given information as the traditional methods do. Therefore it is necessary to explore other neural network concepts as Bayesian Neural Networks (BNN).

   It remains to be seen if similar unsupervised methods can be developed for tasks such as semantic scene labelling.

4. `Practical deployment:` Successes in deep learning have mostly revolved around lengthy training times on supercomputers and inference on special-purpose GPU hardware for a one-off result. A challenge for SLAM researchers is how to provide sufficient computing power in an embedded system. Nowadays is possible to deploy small nets like YOLO or SEGmentation NET (SEGNET) on NVIDIA ARM-GPUs like the Jetson Xavier™but other problems such as transfer learning are still downgrading the quality of the inference in real applications.

`Transfer learning` refers to the situation where what has been learned in one setting is exploited to improve generalisation in another setting.

2.4 Metrics for Localisation

Establishing standard metrics for evaluating the quality of estimated trajectories against ground truth poses is an important task to compare performance of the diverse offer of estimators in the literature.

In this work the Absolute Trajectory Error (ATE) or also known as Absolute Pose Error (APE) and Relative Pose Error (RPE) metrics proposed in [75] are used. The metrics assume a given estimated trajectory $P_1, ..., P_n \in SE(3)$ and a ground truth trajectory $Q_1, ..., Q_n \in SE(3)$, the trajectory must be time-synchronized and in principle, equally sampled. Since in the practice is difficult to equally sample the estimated trajectories, and arbitrary coordinate frames between them can be specified, they must be aligned. The most common method for this task is the Umeyama alignment algorithm [79] which computes the scaling, rotation, and translation that define the transform $\mathcal{S}$ that minimizes the sum of squared errors between the compared trajectory transformations.

ATE is well-suited for measuring the performance of odometry systems since it associates the estimated poses with ground-truth poses.

Given the align, estimate and ground truth transformations, the ATE is define as follows:

$F_i := Q_i^{-1} \mathcal{S} P_i$

The error is evaluated from the Root Mean Square Error (RMSE) for all time indices of the translational components, i.e,

$$\mathrm{RMSE}(\mathbf{F}_{1:n}) := \left(\frac{1}{n} \sum_{i=1}^{n} ||\mathrm{trans}(F_i)||^2 \right)^{1/2} \tag{2.18}$$

Rotational errors typically also manifest themselves in wrong translations and are thus indirectly also captured by the ATE.

The RPE is well-suited for measuring the drift of an odometry system. It computes the error in the relative motion between pairs of fixed timestamps Δ. The error corresponds to the local drift of the trajectory, and it is computed to obtain the relative transformation between the neighboring point of the ground truth and the estimated trajectory. The relative error of each point in the trajectory is given by:

$$\mathbf{E}_i = \left(Q_i^{-1} Q_{i+\Delta}\right)^{-1} \left(P_i^{-1} P_{i+\Delta}\right) \tag{2.19}$$

In the same way as ATE the final error is given by the RMSE of the translational components of the relative pose $\mathbf{E}_i$

$$\mathrm{RMSE}(\mathbf{E}_{1:n}, \Delta) := \left(\frac{1}{m} \sum_{i=1}^{m} ||\mathrm{trans}(E_i)||^2 \right)^{1/2} \tag{2.20}$$

where $m = n - \Delta$

In order to visualise the estimated trajectories and to obtain quatitive evaluation given the mentioned metrics, we used *evo* a python package for evaluation of odometry and SLAM [30], the tool is available at: `https://github.com/MichaelGrupp/evo`.

Chapter 3

Developed Scale Autonomous Car AutoMiny

The Center of Machine Learning and Robotics from the Freie Universität Berlin has had a particular interest in developing robots on which it is possible to test new ideas and ML algorithms. Since 2016 autonomous model cars have been the main activity for education and research purposes, leaving behind the FuManoid Soccer Team project[1].

The aim of building the cars, encouraged by the Autonomos[2] Project, is to have a car-like robot on which Master's and Ph.D. students could learn the underlying architecture and software that drives the autonomous cars MIG and e-Intstein. In order to accomplish this goal, it was mandatory to scale essential systems and sensors, use the same operating system in order to emulate the behavior of a full-scale car as much as possible.

The main advantages of having such a system are:

- `Educational`: students and Scientists wanting to work on full-scale cars can familiarize themselves with the systems, faster through focused robotic and ML courses.
- `Research`: in the current literature, there are perception and control problems related to autonomy which can be tested in a controlled and low-risk environment such as a lab with a setup of several car-like robots.
  - *System Identification*: a critical feature of this system is the possibility to run the cars at high velocity, driving them into drifting or colliding conditions. Such extreme situations can highlight some of the main problems related to system identification, such as wheel odometry, visual odometry, modeling, or design through the collection of vast amounts of driving data in a faster way than with a full-scale car. New control algorithms like Deep Reinforcement Learning (DRL), which learn based on experiences over several iterations before finding an optimal solution, could also benefit from these platforms.

1 `http://www.fumanoids.de/`
2 `http://autonomos.inf.fu-berlin.de/`

- *ADAS*: in the Robotics lab, the University has been possible to test some driver assistance systems with the aim to test them on full-size cars in the future, for example, Adaptive Cruise Control (ACC), Lane Departure Warning System (LDWS), and Collision Avoidance System (CAS). These implementations allow students to notice problematic situations and physical issues while developing the algorithm.
- *Multibody Systems Control*: nowadays, analyzing the behavior of more than one autonomous car on the streets is a hard task due to several obstacles such as budget or state driving permission, not to mention the risk of collisions. The most common solution is to use a simulator on which dynamic features related to the vehicle, and the road must be taken into count. A lab setup with several cars can safely be useful to test algorithms, such as optimal path planning.

This new project was called *"AutoMiny"*. For this work, two versions of the robot were developed, one with educational purposes and another for deep odometry.

AutoMiny is an autonomous model vehicle based on a scaled 1:10 RC car chassis, with a complete onboard system supplied with perception sensors, high computing CPU, and GPU power, as well as LEDs to emulate car lights. The robot is designed as a self-sufficient system that requires no external sensing or computing. It can be controlled remotely (e.g., with a smartphone, RC, or Xbox controller) or can be programmed to drive in fully autonomous mode. The cars run under Ubuntu 18.04 and ROS melodic [74].

There have been other similar platforms developed in other universities. Table 4 the main projects with their main components are shown and Fig. 16 shows a picture of each car.

AUTO	SCALE	HARDWARE
MIT Race car	1/10	-Tegra TX1 -LIDAR -Stereo Camera -Infrared depth camera -IMU
GeorgiaTech AutoRally car	1/5	-GTX 750ti / Intel i7 -GPS -4x Hall-effect wheel encoder -2x Monocular camera -IMU
Berkeley Autonomous Race Car (BARC)	1/10	-Odroid XU4 -LIDAR -GPS -4x Hall-effect wheel encoder -ELP USB Camera -IMU
Amazon DeepRacer Evo	1/18	-Intel Atom™Processor -2 X 4 MP camera -IMU

Table 4: Characteristics of the model autonomous cars for educational and research proposes on the field.

The Massachusetts Institute of Technology (MIT) race car is used mainly for indoor teaching and research activities. The main purpose is to deploy neural networks and machine learning algorithms to drive the car with the included GPU. The code based on ROS, as well as the instructions to build one can be found in `https://mit-racecar.github.io/`.

The AutoRally car is used to develop outdoor applications related to controlling the behaviour of the car during extreme manoeuvres and different soil conditions. The included GPU is used to parallelised computation of the implemented Model Predictive Path Integral Controller (MPPI) [89]. Previous research from this group has shown that it is possible to teach an agent to learn the policies of the MPPI through reinforcement learning to control aggressive manoeuvres [90]. Videos and complete tutorials about building and programming a car are accessible through their github web page at `https://github.com/AutoRally`.

The Berkeley Autonomous Race Car (BARC) is used for outdoor and indoor applications such as MPC, corner drifting [37], lane keeping, obstacle avoidance, or traction control. The car works under ROS framework and is used as an instructional platform in courses for vehicle dynamics and control. More information about the car can be found on the website: http://www.barc-project.com/.

The Amazon DeepRacer was released on March 6, 2019, with a price of $399. With only one camera and an Intel Atom processor, the idea of the company was to have a platform to deploy neural networks trained through reinforcement learning techniques. It is a car with simple hardware, the main business of the company is the use of the Amazon Web Services (AWS) platform and courses offered on machine learning and DRL. In 2020, the Amazon car in Fig. 16 will be released; this model has two cameras in order to get depth maps through stereo matching and a LIDAR. The use of this platform is merely educational.

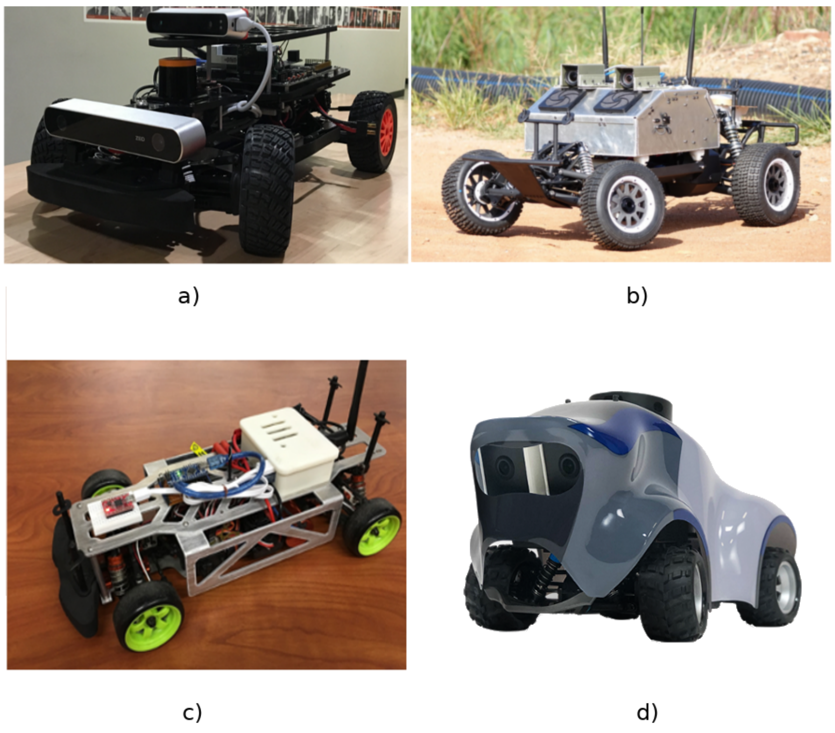


Figure 16: Model Car Projects. a) MIT Race car, b) Georgia Tech Autorally car, c) Berkeley Autonomous Race Car, d) Amazon DeepRacer.

There are fundamental differences between the robots from other Universities and AutoMiny, but in order to understand these, the architecture of the two AutoMiny versions must be explained separately.

3.1 Educational AutoMiny

The base hardware of the car, processes all the algorithms on an Intel NUC CPU and provides the necessary configuration to run the car autonomously. The design was thought to be easy maintenance. In our experience in the classroom with students, the car is not exempt from colliding with other cars or the walls at high speeds. Although, the car can be easily repaired by changing each of the separated modules and spending more time on programming, and testing than repairing hardware. The camera holder is 20cm above the ground to allow the car to see the track ahead and process the images in order to execute tasks such as localization, lane detection, or obstacle detection.

The base platform of the AutoMiny is shown in Fig.17 and the related components are described in Table 5.

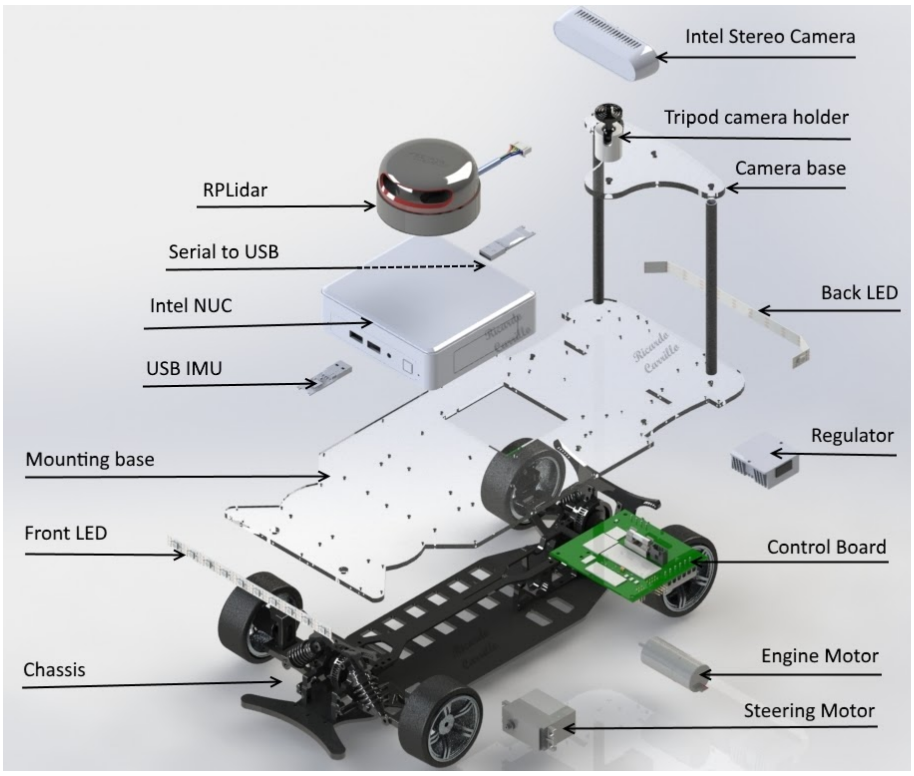


Figure 17: AutoMiny core

AutoMiny is mainly based on two separate processing modules: one controller board with a microprocessor (Arduino nano), and an Intel NUC computer. The controller board is a four-layer PCB where the Arduino controller and an additional IMU is installed; the board is mainly responsible for the following tasks:

- Battery Voltage Checker: the primary source of power the AutoMiny uses is a 14.8V Lipo battery with 4000 mAh. Due to the nature of the battery type, it cannot discharge below 13v, or the battery cells can be damaged. Based on our experience in the lab with dozens of students, we know that being aware of such delicate parameters is not always guaranteed; therefore, the controller board checks the voltage of the battery at the start and during the activities. This task is developed by a voltage divider, the Arduino and a relay. In order to turn on the car, it is

NO.	ITEM	DESCRIPTION
1	Chassis	Xcite RC car chassis (1:10)
2	Steering	Adafruit Servo Motor with Analog feedback
3	Engine	Faulhaber motor with encoder
4	LIDAR	RPLIDAR A2M8 360° one beam Laserscanner
5	Infrared Stereo Camera	Intel Real Sense D435
6	IMU	BOSCH BNO055 USB Stick
7	Control Board	Developed at FU Berlin
8	Voltage Regulator	Converts battery voltage to 5v
9	LEDs	LED strips, 11 LEDs at the back, 10 LEDs at the front
10	Engine	Faulhaber motor with encoder

Table 5: AutoMiny Components

necessary to press the push-button for 5 seconds, during which the Arduino calculates the average of the measurements and diagnoses the battery. If the voltage is enough, the middle, frontal, and back LED, changes from red to green and the software of the car starts. During the activities, the car automatically turns off if the voltage is under 13v.

- Chassis Sensor Data Acquisition: the Arduino reads the information given by both motors: Engine and Steering.
- Engine: the DC motor has an incremental encoder which allows us to know the velocity and direction of the shaft; the pulses are detected by hardware interruptions on the Arduino.
- Steering: the servo motor has an analogical voltage output which is read by an analogue input and transformed to normalized values between -1 and 1 with a previously calculated calibration. The information is sent through serial communication to the NUC computer using a binary protocol in order to publish topics in the ROS environment on the NUC.
- Voltage Distribution: in the AutoMiny, two primary voltages are used: battery voltage (above 12v), and 5v. The board distributes

the battery voltage to the regulator, which converts it to 5v; battery voltage also sources the Engine power and the NUC computer. Optionally, the Jetson Xavier receives voltage from the board as well. The board receives the 5v from the regulator and distributes it to the Arduino, Steering, and Engine electronics. Optionally in the AutoMiny Nano, it also feeds the Jetson Nano from NVIDIA.

- Control of the Chassis: the Arduino communicates with the NUC through a binary protocol. An ongoing interchange of data is happening all the time with the Arduino in order to execute the desired commands published on the topics which control the Engine, Steering, and LEDs.

The second module is the NUC Intel computer which is the main processor of the AutoMiny; it handles the data coming from the controller board, LIDAR, a Bosch USB IMU, and the Stereo Camera to drive autonomously. A diagram with the underlying architecture of AutoMiny can be observed in Fig. 18.

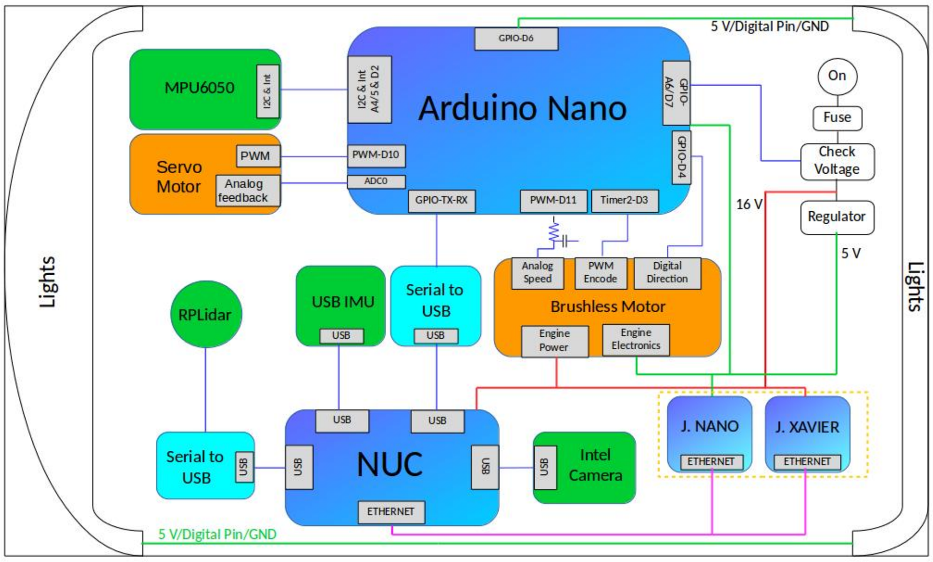


Figure 18: AutoMiny architecture

Regarding the AutoMiny architecture, the behaviour of each omponent in the control board is like follows:

- IMU: We included two IMU on behalf of the locally available budget, one slot for a typical MPU6050, which communicates with the Arduino through I2C protocol. However, such IMU, although cheap, manufacturing issues as well as factors such as operating temperature, showed us through time that lectures may be inaccurate and imprecise among different cars. This issue is a big obstacle for generating shared packages of software. Therefore we added a second more robust option, a Bosch USB IMU, which connects directly to the Intel NUC computer; it is possible to publish the messages through ROS with the manufacturer C++ library.

- Servo Motor: Is the steering motor, control is made through Pulse Width Modulation (PWM) by the Arduino, and the analog feedback allows us to know the position of the servo motor reading the analog voltage in the ADC0 pin.
- RPLidar: the device output is transformed using a serial to USB converter directly to the Intel NUC PC, the RPLidar ROS package automatically publish the information in the system.
- Control board - NUC communication: we use a serial to USB converter from the Tx-Rx pins of the Arduino to a USB port in the NUC PC. The available mini USB port in the Arduino is preferably not used due to the voltage pin, which causes troubles with the 5v source from the battery.
- The Brushless motor: is the engine of the car, the velocity is controlled through the PWM-D11 pin in the Arduino and the Timer2 which gives the encoding of the PWM, digital pin D4 gives the direction of the rotation. The internal electronics are sourced at 5v, but the engine power is sourced with 16v.
- Intel camera: the camera is powered and communicated through USB-C to the Intel NUC. ROS framework is commercially available.
- Check voltage circuit: prevents the battery from running below 14v reading in the analog input voltage pin A6 a voltage divider. In the Arduino, a condition is triggered to activate or deactivate a relay, which can break the main supply if the voltage is low. It also displays a warning to the user using a ROS message.
- GPUs: The model can work with a Jason Nano or Xavier arch GPU, they are powered with 5v and 16v respectively and communicate to the NUC with Ethernet protocol.
- Lights: the front and back stripes of LEDs work to indicate the basic car lightening functionality, i.e., blinking while turning, stopping, and turning on. The middle back LED turns green-orange-red depending on the voltage status of the battery. The stripes are completely programmable, and basic behavior can be controlled through ROS.

With the technology being developed nowadays, based on Artificial Neural Networks, and to exploit their advantages of processing perception data from cameras, AutoMiny can be upgraded to work with

Jetson Nano[3] and Jetson Xavier[4] from NVIDIA. The Intel Stereo camera is accessed by the GPU through the NUC, since the ROS MASTER in the GPU is pointed to the NUC ROS MASTER is possible to access all the published topics. We have tested the image rate in the GPU to be virtually the same as in the Intel CPU. The results obtained from the Nvidia boards are sent to the NUC via Ethernet to control the car with the desired criteria. Additionally, the camera holder has the option to install another Intel Camera, the idea behind this alternative, is to provide the car, the field of view also when is driving backward or obstacle recognition.

The educational AutoMiny cars with and without the updated feature can be observed together in Fig.19

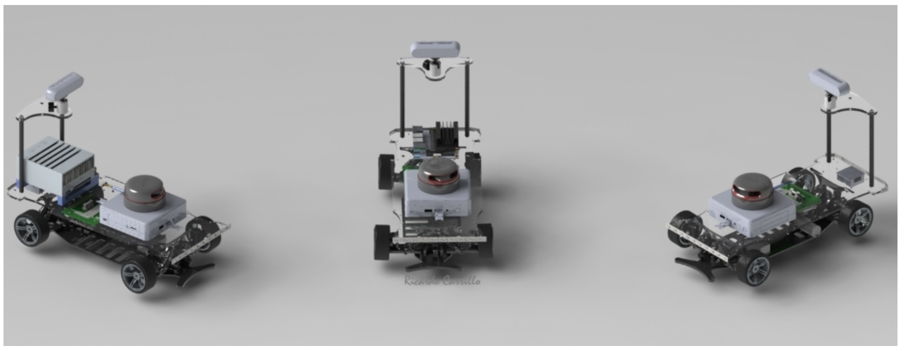


Figure 19: AutoMiny Team

Even more, the developed platforms are the primary educational tool used on the Robotic lectures and practice of the Freie Universitaet Berlin, and students around the world in countries including Mexico, Spain and Italy. Since the project is open source, the research collaboration has been rich and has debugged the error of the car to this last version. Full information about the project can be found in the official AutoMiny web site: `https://autominy.github.io/AutoMiny/`.

3.2 AutoMiny TX1

In addition to the presented versions of AutoMiny, a new version only for neural network-based odometry purposes (i.e., this work), was designed. Figure 20 shows the AutonoMiny TX1 version

This version is equipped with the hardware described in table 6, there are several differences with the educational version of AutoMiny, since AutoMiny TX1 was developed to drive in a more aggressive way, the brushless motor is driven with an ESC, while drifting we measured currents up to 15 Ampers, and angular velocities up to 600rpm while educational AutoMiny we recorded a maximum mean angular velocity of 380rpm. Another difference is the power supply

3 `https://developer.nvidia.com/embedded/jetson-nano-developer-kit`
4 `https://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit`

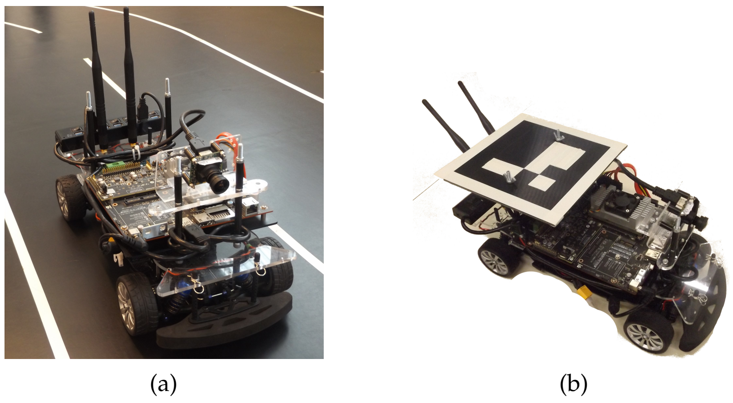
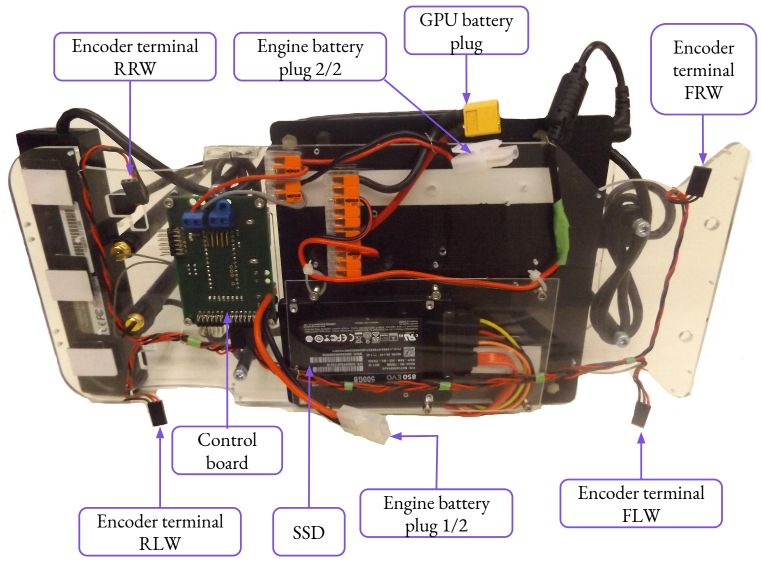


Figure 20: Developed AutoMiny TX1 version with odometry research purposes. a) AutoMiny TX1 on a track made with black rubber used to achieve aggressive driving maneuvers, The ARUCO marker in b) is used to localize the car on the track.

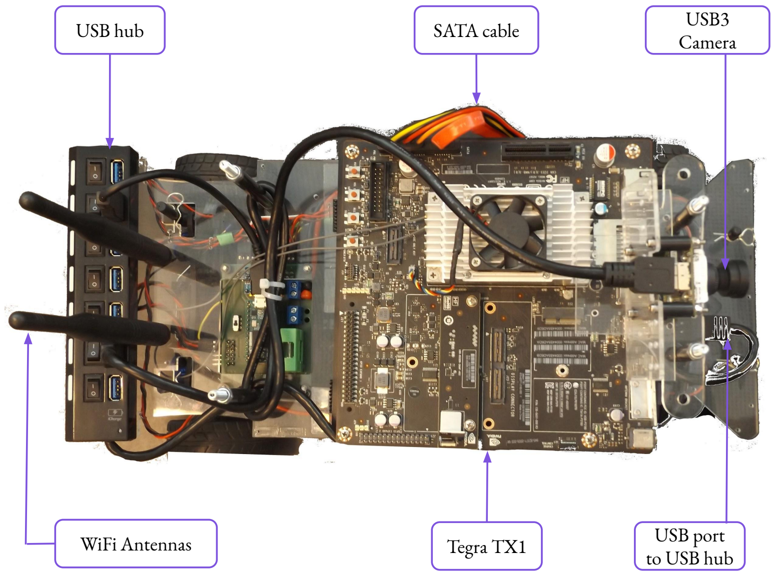
to the car, we use two 7.4v NiMh batteries in parallel, and one separated 14.8v LiPo battery for the GPU in order to avoid a preventive shutdown due to voltage spikes on the engine controller. In order to collect a high amount of images and data, the operating system is loaded from a 500G Solid State Drive (SSD).

The monocular camera showed 20 is an IDS UI-3241LE[5] and was used for indoor porpuses only. For outdoor purposes, the GPU takes the most relevance since we use a ZED™Stereo camera since the necessary computations are not carried on board. The GPU is used for image acquisition, stereo matching, among other processes. Figure 21 shows a detailed view of the components of AutoMiny TX1.

5 https://en.ids-imaging.com/store/products/cameras/ui-3241le.html



(a) Bottom view of AutoMiny TX1



(b) Top view of AutoMiny TX1

Figure 21: Detailed hardware view of AutoMiny TX1

The AutoMiny TX1 framework is based on ROS; the available packages allow the user to control the velocity of the electronic speed controller and the steering wheel as well as read the IMU, the wheel velocities, the position of the steering wheel and the camera images.

NO.	ITEM	DESCRIPTION
1	GPU	NVIDIA™Tegra TX1
2	Chassis	Xcite RC car chassis (1:10)
3	Steering	Adafruit Servo Motor with PWM feedback
4	Engine	Bruschless DC Motor
5	Camera	ZED™Stereo Camera
6	IMU	BOSCH BNO055 USB Stick
7	Control Board	Developed at FU Berlin
10	Encoders	4x AMS™wheel hall-effect encoders

Table 6: AutoMiny TX1 Hardware Components

We installed a set of three cameras on the lab ceiling to obtain ground-truth localisation. With this information, we can develop and evaluate a wide variety of localisation estimators. The cameras provide global localisation of the model car employing an ARUCO marker on the top of the car; the mentioned setup is shown in Fig. 23 and 22. The size of the map is 4.3m x 6m.

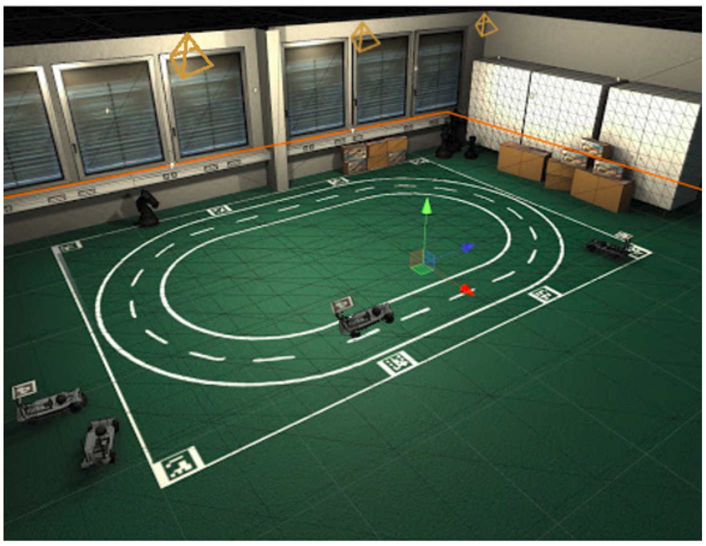


Figure 22: Laboratory setup for global localization. The top figure shows the three cameras on the ceiling (in orange) and the ARUCO codes on the floor and on top of the model cars in order to localize the field and the autos related to coordinate x,y = (0,0) of the map which is located on the top right corner as shown in the bottom picture with the coordinate system.

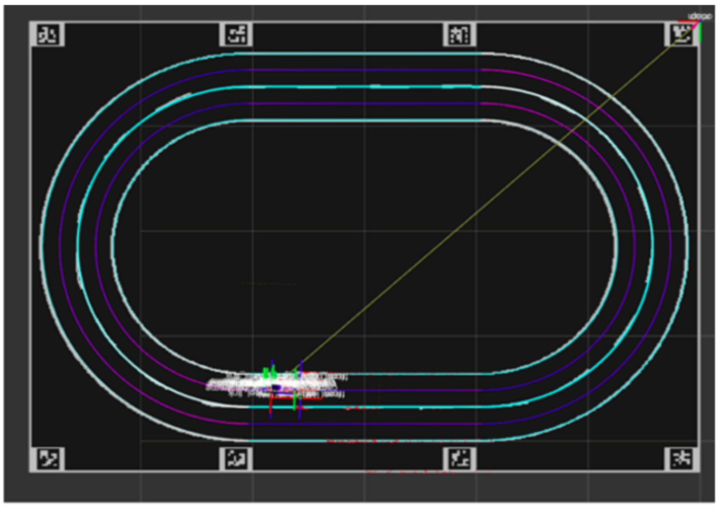
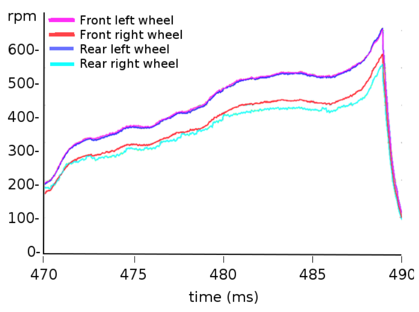


Figure 23: Laboratory test track map. The full car transformation chain and the map origin is shown. The linkage includes camera frame, IMU frame, one frame per wheel, mass center, and base link. The map also shows the defined paths to drive in purple and the lane borders in cyan color.

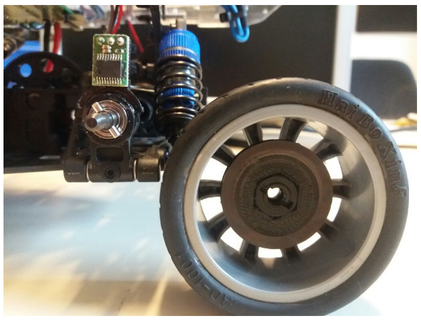
The wheel encoders estimate the angular velocity by taking into account the parameters that affect a conventional ABS sensor which include the presence of mechanical imperfections such as non-concentric montage or non-parallel mounting of the sensor relative to the magnetic ring, as well as variations on the electromagnetic field due to the spinning of the wheel, mechanical adjustments, or intrinsic hysteresis of the sensor. These parameters are difficult to measure since they are inherent in the manufacturing process.

In order to solve this problem, we applied a timestamping algorithm, which consists of capturing via a high-resolution clock the time instants and positions of several encoder events. We sample the pulse transitions of the encoder's output signal at frequency f_s and performing an n-order polynomial fit at the controller's sampling rate $f_c << f_s$ to approximate the position of the wheel. The regression problem can be formulated for the last n events as $AP = B$, P must be calculated from this equation, and the resulting parameters fit the polynomial with respect to time for displacement, velocity, and acceleration [54]. A chain through serial communication with the information of the timestamps of each wheel is sent to the CPU in order to calculate the velocity. Fig. 24a shows a graphic with the plotted wheel velocities while the car is driving in circles and with the sensor installed on the wheels.

The technical characteristics of the sensors can be founded on the official AMS^{TM}website: `https://ams.com/as5311`. Each wheel has 16 pulses per revolution. The sensor is shown in Fig. 24b



(a)



(b)

Figure 24: a) Wheel Angular Velocities of AutoMiny Tx1. Front and rear left wheels have a bigger angular velocity since the car is driving on circles counterclockwise; when the car stops, velocities drop to zero. b) The Hall sensor mounted on the car chassis and the magnetic ring installed on the wheels.

In this work, both the MIG platform and the AutoMiny were used to test the developed algorithms. The AutoMiny Tx1 is particularly essential since the wheel odometry network was improved using the data generated with extreme maneuvers, and driving manually between the limits of the localisation set up in the lab. In the following chapters, it will be shown how the scaled platforms were used to increase the data size for training and validation.

Chapter 4

Deep Learning for Ground Dynamic Localisation

In this chapter, we propose the use of deep neural networks to estimate localisation given data obtained from the vehicle inertial and mechanical sensors.

As described in 2.2.1, finding a better mathematical model to estimate the displacement between two consecutive samples of vehicle kinematic-dynamic information is key to compensating the drift while using sensors such as encoders and IMUs. All the inherited inaccuracies impact the estimation of:

- Continuous wheel velocity from discrete hall sensors and related wheel physical disturbances.
- Vehicle angular and longitudinal velocities as well as accelerations affected by bias, numeric integration, precision oscillation due to temperature, among other disturbances.

Inaccuracies could, in principle, be estimated in a regression problem with an n-dimensional equation. This chapter explains the strategies to find an approximation to this function.

Moreover, we will explain how the vehicle AutoMiny was finally used to expand the training data used to train the final net for the proposed localisation scheme.

4.1 Dataset Construction

Careful generation of the dataset for training, validation and testing is the foundation of every supervised machine learning application. Accuracy is directly affected by the way the data is collected, fed and extracted from the learning agent. Data science is itself a big problem, and in order to minimise the risk of a malformed generation of a database, statistical analysis of the data is needed. This sections presents the followed strategy to extract, and shape the dataset in three subsections, first we explain the nature of the included variables in the dataset, next we show the available architecture in the cars that allows to obtain the data and finally the combination of the MIG and AutoMiny datasets.

4.1.1 Dataset Parameters

To be able to train a neural network though supervised learning, we build the dataset saving an input array of size nx9 every Δ of time, being n the number of timestamps in the dataset. Each row in the matrix has the following features:

$$\mathcal{I} = [\delta, \nu_{ref}, a_x, a_y, \dot{\Psi}_V, \omega_{rl}, \omega_{rr}, \omega_{fl}, \omega_{fr}] \tag{4.1}$$

where: δ is the steering angle, ν_{ref} is the speed over ground, a_x is the longitudinal acceleration, a_y is the cross acceleration, $\dot{\Psi}_V$ is the yaw rate and ω_{xx} is the speed of the front and rear wheels.

The idea behind including just the variables in 4.1 and not other parameters such as constant vehicle mass, size of the wheel-base, wheel diameter or others, is that we are looking for a more complex representation of those parameters that allow the neural network to predict odometry for different size of vehicles. Even that some parameters may be intuitive to add if we are planning to train the network to find dynamic associations such as VSA rate, those can be derived from the input vector. Neural networks are capable of finding constants (in the form of biases) if necessary for the calculus of the desired output. Furthermore, as explained in 2.2.1 one of the drawbacks of the current odometry approaches is not to include physical events that could take place in a real scenario, such as vehicle weight variability that could be measured by a change in the wheel forces, which in turn, can be approximated with the wheel velocity and a constant tire profile according to the contact surface. Deep Learning has shown effective in finding high dimensional associations in other areas, and we are looking to find for this application a network architecture able to estimate them. In this sense, we consider the input vector to be sufficient for the task.

Since an element of the dataset describes the instantaneous dynamic state of the car, in order to be able to estimate a displacement between two timestamps we build the inputs of the net as a concatenated pair of vector 4.1 for instance:

$$\mathcal{X}_k = [\mathcal{I}_i, \mathcal{I}_{i+1}] \tag{4.2}$$

For the net output, we compose an array of size nx4, which includes the poses of the car. Instead of recording the global pose, we stored local relative poses. Local poses allow us to predict local displacements with the net; in order to construct a global trajectory, the corresponding geometric transformations have to be made.

The positional data of AutoMiny was obtained from the global position estimator employing the ceiling camera setup and the ARUCO markers. In the case of the MIG, the ground truth is obtained from the Applanix. To ease the net training, we reduced the problem to a two-dimensional trajectory. Although for two-dimensional estimation, only the Yaw angle to define orientation is needed, we stored the quaternion values; during training, the quaternion representation showed to help the net to converge faster. In a Euler rotation of the format ZYX, the non zero quaternion values are qz and qw. The generated vector is then as follows:

$$y_k = [x, y, qz, qw]$$

Next, we present the employed ROS framework architecture on the AutoMiny and on the MIG that allows us to collect the data.

4.1.2 *Employed Software Architecture for Data Extraction*

The software framework of all the platforms is based on ROS melodic and Ubuntu 18.04. The information gathered from the chassis of the vehicles is sent through CAN bus and serial communication from the MIG and AutoMiny respectively, and the local computer creates virtual sensors based on estimators and chassis information, the local computer also provides node to access the chassis in order to control the engine, brakes, and steering. The simplified architectures of the ROS packages used in the MIG and the AutoMiny focused on obtaining ground localization are shown in Figures 25 and 26 correspondingly.

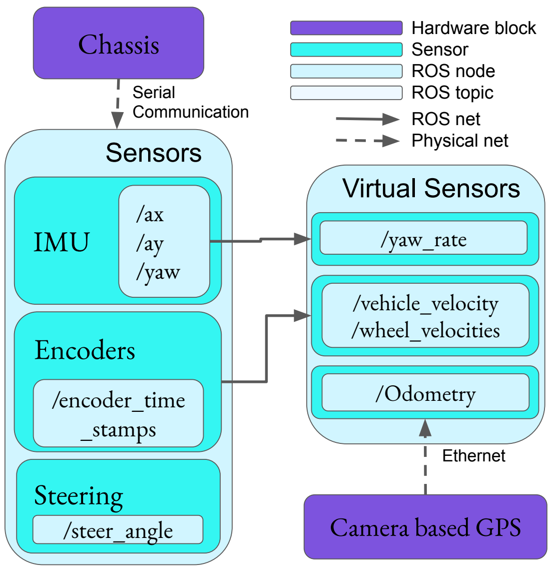


Figure 25: AutoMiny ROS architecture.

In the Table 7 the corresponding topics of the AutoMiny and MIG from which the parameters of the vector 4.1 were obtained are shown. Since the architecture of the ROS packages in the AutoMiny and MIG are similar, having the same message types allowed us to generate the database in the same process.

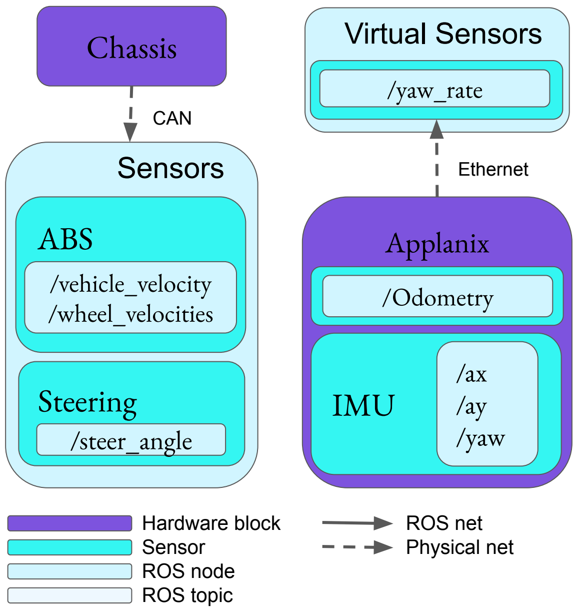


Figure 26: MIG ROS architecture.

One crucial aspect of the data set generation is the frequency at which the data is sampled. The sensors in the chassis, besides their inherent working rate, have a time delay due to communications and prepossessing in the local computer. Virtual sensors, depending on the employed estimator, may have a frequency rate below 30 Hz, which is the minimum to be considered a real-time application.

Third column on Table 7 shows the rate of the topics obtained with `rostopic hz`. The sampling rate is commanded by the slower topic, which is the localisation of the AutoMiny. The low rate of the localisation is due to the frame rate of the ceiling cameras and the post-processing algorithm, which stitch the images, detect the ARUCO markers, and calculate the position of the car. Therefore, the sampling rate of the AutoMiny dataset is 30Hz, while the sampling rate for the MIG dataset can be configured up to 100Hz.

AutoMiny			MIG	
PAR.	TOPIC	HZ	TOPIC	HZ
δ	/feedback_steering	100	/carstate/steering_angle	100
Vref	/wheel_velocity	100	/carstate/speed	100
ax	/acceleration	100	/sensors/applanix/imu_data	200
Ψ'v	/YawPitchRoll	100	/sensors/yaw_rate	200
ω	/wheel_velocity	100	/sensors/can/wheel_speeds	100
X	/localization	30	/localization/odometry/filtered_map	200

Table 7: Used topics for ground localization in Autominy and MIG

We To create a database without redundant data, the reading rate in the MIG was also configured to 30Hz; in this way, we averaged a traveled distance of 0.5 m per time-step. We carried out a sanity check before storing the data. The sanity check verifies that the candidate vector is not similar to any of the already stored vectors through cosine similarity, which is a dimensionless measure for vectors, which are typically sparse.

$$\mathrm{sim}(x, y) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|} \tag{4.3}$$

The similarity equation 4.3 computes the cosine between two multidimensional vectors so that a value of zero means the vectors have ninety degrees between each other, the closer the cosine value to one, the smaller the angle and the higher the match between vectors. In our experiments, this value was set to 0.9.

The sanity check has to cope with another big issue, the periodic GPS corrections of the Applanix. Those corrections produce discrete spatial jumps up to 0.5m. Since those jumps in a position generate a non-continuous trajectory, the dataset is sectioned every time a jump is detected. In this way, we store small continuous trajectories with local transformations instead of a long trajectory with global transformations.

4.1.3 *Dataset Assembling*

The datasets were obtained reading the recorded rosbags showed in Tables 8 and 9. AutoMiny was driven manually in two different trajectories, under variable dynamic behaviors; Fig. 27 shows a trajectory of the AutoMiny periodically increasing its velocity without changing the steering angle, the vehicle develops high VSA which explains the change of the radius in the semi-circle trajectory. The second trajectory was obtained driving on the lab track showed in Fig. 23 counterclockwise and clockwise at different velocities and skidding on the turns.

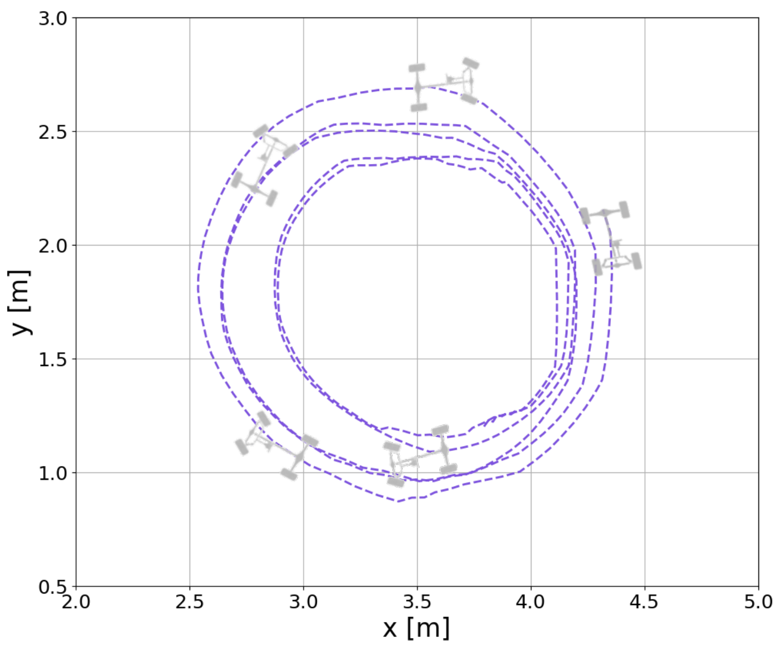


Figure 27: AutoMiny circle trajectory carried out with the same steering angle while increasing velocity, which developed high VSA and VSA rates.

AutoMiny data was recorded using two different surfaces, Cut Pile green carpet, and PVC black rubber. Driving on different surfaces allowed us to generate diverse, dynamic performances. While on the rubber surface, the car was able to drift easily, on carpet it was more stable on the curves. Analysing how the dynamics of the car changes on such different surfaces is key to develop more accurate controllers and estimators. In this sense, another advantage of using scaled models is the ease of studying such properties by changing the surface conditions on the lab floor. On full-scale cars may be difficult to find the conditions to generate a dataset with all the driving conditions that a driver can find in a variety of terrains and, at the same time, having an accurate estimation of pose to measure the results of the developed controller or the estimator.

The dataset of the MIG was recorded in the city of Berlin on eleven different expeditions. The conditions of weather and season of the year are similar; there are no recorded datasets while driving on snow or rain. Therefore, it is of our interest to expand the dynamic scope of the MIG trained network by means of including the founded associations founded under the AutoMiny dataset.

To examine the dynamic range of the datasets, it is possible to visualise the slip angle rate derived from Eq. 2.7. For AutoMiny, the Probability Distribution Function (PDF) of each friction surface is shown in Fig. 28 together to the MIG slip angle rate, considering that the available trajectories of the MIG are only recorded on asphalt, in an average velocity of 20 m/s, therefore only one PDF of slip angle rate is shown.

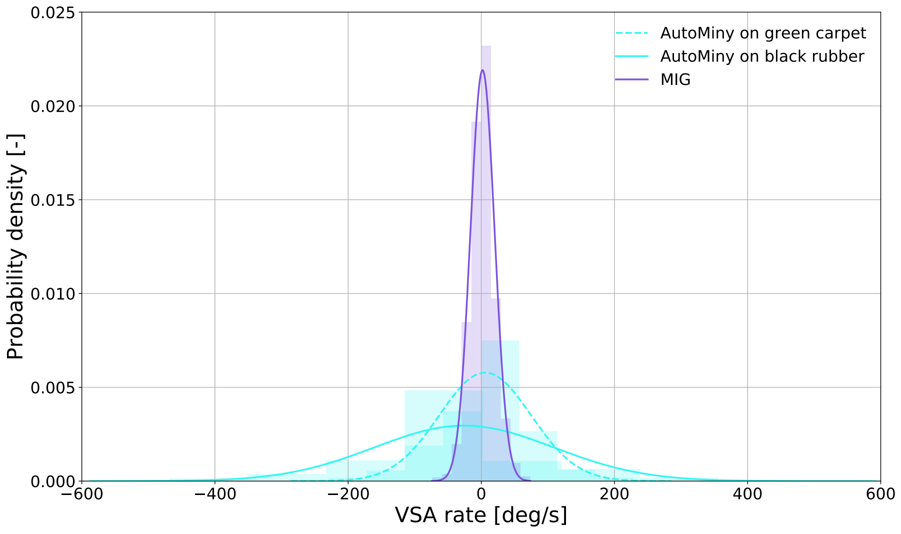


Figure 28: Range of VSA rates ($\dot{\beta}$) in the datasets. AutoMiny developed higher $\dot{\beta}$ over black rubber, while as expected, the MIG performed the smallest values.

As shown in Fig.28, we drove the AutoMiny on a higher dynamical range than the MIG. The differences between the experiments, improve the accuracy of the MIG net by increasing the gamut of the data on which the net is trained. The overlapping between both ranges, allow the net to build a standard feature map between the AutoMiny and MIG dynamics.

While it is true that among different vehicles, associated parameters such as vehicle mass or the kinematic structure are clearly different, the objective of experimenting with deep neural networks is to find relationships between the dynamic parameters of the input to calculate displacement independently of constant parameters. In principle, the mathematical approaches which calculate useful dynamic parameters such as tire forces can be derived from a relationship between wheel angular velocity and constant parameters. Computing the tire forces is possible to know the vehicle weight. Therefore, the goal is to develop a model that finds high dimensional functions to relate different car models with their displacement.

In Table 8, the amount of train and validation timestamps are shown. Since on each dataset, the trajectories are constantly repeated, the test section can be safely taken from the same file and use it to evaluate the results of the net. In the counterpart, the MIG rosbags are unique trajectories, and we are interested in leaving trajectories

		#Timestamps			Driven
Nr.	Scenario	Train	Val.	Test	length(m)
1.	br normal driving	2451	817	817	285.7
2.	br drifting	2375	791	791	216.6
3.	gc normal driving	4716	1578	1578	537.9
4.	gc drifting	3892	1291	1291	349.2
	Total	13434	4477	4477	1389.4

Table 8: Recorded rosbags in Autominy over black rubber (br) and green carpet (gc). Different scenarios where used to increase the range of the dynamic parameters in the vehicle.

entirely for testing, owing to the fact that we like to analyse how the net generalise for unseen trajectories.

We acknowledge that the partition of the datasets into training and validation could avert some interesting dynamic information to the net if the data is indiscriminately divided; therefore, K-fold cross validation was used to evaluate the performance of training on unseen data. That is, to use dedicated data samples in order to evaluate how the model performs in general when used to make predictions on new data not used during the training of the model. It generally results in a less biased or less optimistic estimate of the model skill than other methods. The dataset was divided in four groups ($K = 4$). Unlike the common K-fold validation, the dataset is not randomly shuffled across the members, since it is important to maintain the complete trajectories as much as possible; instead, the entire trajectories were shuffled.

To train the final model, from the total timestamps in the AutoMiny dataset, 60% were taken for training, 20% for validation and 20% for testing. In the MIG dataset, the sequences were divided between 80% training and 20% validation.

The databases are stored using the Pandas[1] data frame, each column of the resulting matrix contains the mean and standard deviation of the column obtained from standardizing the data as follows:

```
mean = train_data.mean(axis=0)
train_data -= mean
std = train_data.std(axis=0)
train_data /= std
```

This way of representing the data is more convenient for the training of the net. The total database is composed of 22388 samples from the AutoMiny and 27338 samples from the MIG.

1 https://pandas.pydata.org/

		#Timestamps			Driven
Nr.	Scenario	Train	Val.	Test	length(m)
1.	20190603-FU_to_OBI	1260	314	0	2356.4
2.	20190307-safari_online	7326	1831	0	12461.4
3.	20190621-thielallee	0	0	3058	4795.3
4.	20191113-eng	0	0	583	944.8
5.	20191122-react4	1560	389	0	2853.9
6.	20191128-reinickendorf	419	104	0	1659.5
7.	20191128-aut7	2784	695	0	5907.6
8.	20191128-auto8	0	0	1061	1487.9
9.	20191210-2tegel	3084	771	0	9968.6
10.	20191210-back2fu	1680	419	0	8326.9
	Total	18113	4523	4702	50762.3

Table 9: Recorded rosbags in the MIG and the number of timestamps in the dataset. Sequences 3, 4 and 8 are used only for training, meanwhile the rest of the sequences are divided between validation and training. The trajectories are shown from Fig. 35 to Fig. 44 in dashed line.

Since the problem is treated as time-series, the dataset is not shuffled randomly; instead, it is divided into several sequences of different sizes with different starts and endings, this helps to expand the learning dataset, in an analogy to dataset augmentation for image-learning tasks. The initial member of each sequence is taken as the initial position of the series, and the transformations are recalculated. Nevertheless, we trained the net to learn the local transformations between timestamps, therefore, to test the net it is necessary to estimate the global position of the vehicle by calculating the $SE(3)$ transformation to the origin frame. It was found that feeding different sequence sizes, and shuffling such sequences each epoch, improved the performance of the net significantly. Validation and test datasets are not divided into smaller sequences; instead, the full sequence is estimated.

A standard ADAM optimiser was used with a learning rate of 0.001. The best results were led from a combination of dropout and L1 regularisation.

4.2 Model for Ground Dynamic Localisation

4.2.1 *Architecture Description*

In this work, the same network architecture is applied to create a dynamic ground localisation for the AutoMiny and the MIG. First, the neural network is trained with the AutoMiny database; then, the feature map is used to train the net for the MIG. The net has been named GALNet.

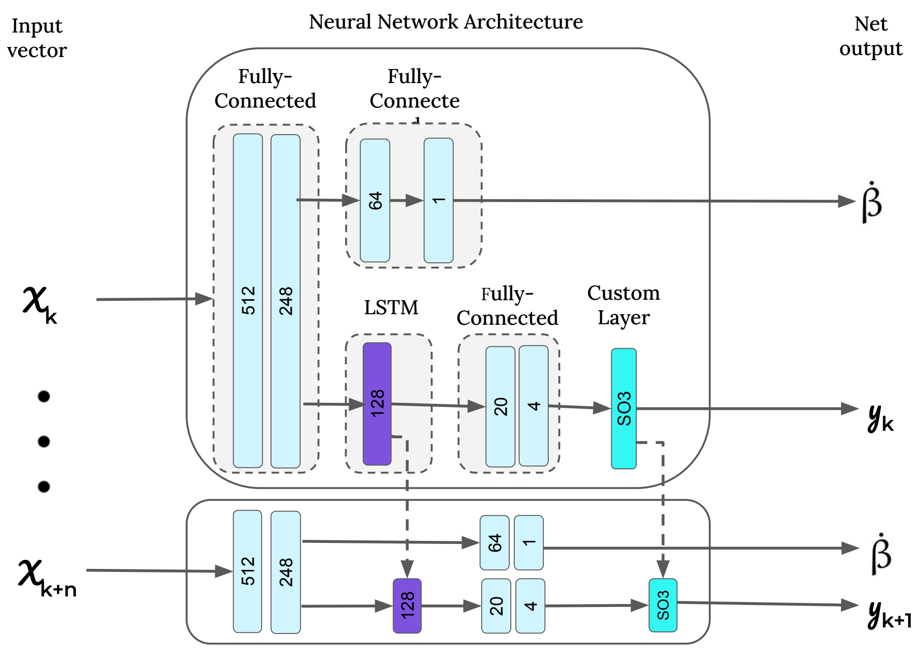


Figure 29: Architecture of GALNet. The net is represented on times k and $k+1$ to show how the states of the LSTM and SO3 layers are forward propagated to the next training step. The first two full connected layers computes the first relationship between the dynamic variables, we use this block in a way of an encoder for the derived two outputs. In one side another dense block computes the VSA rate. The second output is made of a LSTM layer which finds the time series relationships, followed of a fully connected couple of layers that reduce the dimensionality to fit the pose output vector. The net estimates local transformation between two timestamps, therefore, a fixed custom layer projects the displacement to a global frame.

This work utilises LSTM with a projection layer and two regression heads to estimate the slip-slip angle and localisation. The LSTM exploits correlations among the time-correlated data samples, in long trajectories by introducing memory gates and units [33] in order to decrease the vanishing gradient problem [32]. As shown in figure 29 the LSTM is able to explicitly determine the previous hidden states to be discarded or retained for updating the current state, is expected to learn motion and dynamics during pose estimation. In detail, the LSTM net is shown in Fig. 30, the control gates control how information is obtained from previous states and passed to the future. The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates.

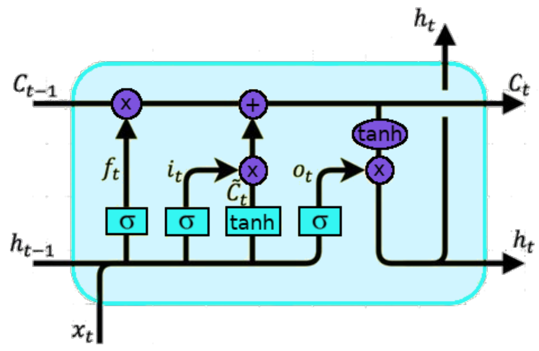


Figure 30: Configuration of a LSTM layer.

Gates are bypasses which let information pass through the next step. They are composed out of a sigmoid neural net layer and a point-wise multiplication operation. The sigmoid layer is defined between zero and one, indicating how much of each component pass forward. A value of zero means that the parameter is not let trhough while a value of one allows the parameter to pass completly through. An LSTM has three of these gates, to protect and control the cell state. Given a convolutional feature $\mathbf{I}_k$ at time k, the hidden state $\mathbf{h}_{k-1}$ and the memory cell $\mathbf{c}_{k-1}$ of the previous LSTM updates at time step k according to:

$$\begin{aligned}
\mathbf{i}_k &= \sigma(\mathbf{W}_{Ii}\mathbf{I}_k + \mathbf{W}_{hi}\mathbf{h}_{k-1} + \mathbf{b}_i) \\
\mathbf{f}_k &= \sigma(\mathbf{W}_{If}\mathbf{I}_k + \mathbf{W}_{hf}\mathbf{h}_{k-1} + \mathbf{b}_f) \\
\mathbf{g}_k &= \tanh(\mathbf{W}_{Ig}\mathbf{I}_k + \mathbf{W}_{hg}\mathbf{h}_{k-1} + \mathbf{b}_g) \\
\mathbf{c}_k &= \mathbf{f}_k \odot \mathbf{c}_{k-1} + \mathbf{i}_k \odot \mathbf{g}_k \\
\mathbf{o}_k &= \sigma(\mathbf{W}_{Io}\mathbf{I}_k + \mathbf{W}_{ho}\mathbf{h}_{k-1} + \mathbf{b}_o) \\
\mathbf{h}_k &= \mathbf{o}_k \odot \tanh(\mathbf{c}_k)
\end{aligned} \tag{4.4}$$

where $\odot$ is the element-wise product of two vectors, σ is the sigmoid non-linearity, tanh is the hyperbolic tangent non-linearity, $\mathbf{W}$ terms denote corresponding weight matrices, b terms denote bias vectors, $\mathbf{i}_k, \mathbf{f}_k, \mathbf{g}_k, \mathbf{c}_k$ and $\mathbf{o}_k$ are: input gate, forget gate, input modulation gate, memory cell and output gate at time k, respectively.

Despite the ability of the LSTM to handle long-term dependencies, learning them is not trivial, for which in this work a series of linear convolutional layers are used in order to extract the model dynamics.

4.2.2 Loss Functions

The model uses the slip angle as an auxiliary input to the pose regression, therefore there are two losses in the model, one to predict the correct slip angle and other to regress the position. The loss regarding slip angle is as follows:

$$\mathcal{L}_{VSA} = \frac{1}{N}\sum_{N} \|x_n - \widehat{x}_n\|_1 \tag{4.5}$$

where N is the size of the batch.

As for pose estimation, the proposed method can be considered to compute the conditional probability of the poses $\mathbf{Y}_k = (\mathbf{y}_1, ..., \mathbf{y}_k)$ given a sequence of vector 4.1 in time *t*

$$p(\mathbf{Y}_t|\mathbf{X}_t) = p(\mathbf{y}_1, ..., \mathbf{y}_k|\mathbf{x}_1, ..., \mathbf{x}_k)$$

In order to maximize the previous equation, the parameters of the net can be found based on Mean Square Error (MSE), the Euclidean distance between the ground truth pose $\mathbf{y}_k = (\mathbf{p}_k^T, \mathbf{\Phi}_k^T)$ and its estimate $\widehat{\mathbf{y}_k} = (\hat{\mathbf{p}}_k^T, \widehat{\mathbf{\Phi}}_k^T)$ at time *k* can be minimized by

$$\mathcal{L}_{POS} = \frac{1}{|N|}\sum_{N} \|\widehat{\mathbf{x}}_k - \mathbf{x}_k\|_2 + \kappa \left\| \widehat{\mathbf{q}}_k - \frac{\mathbf{q}_k}{\|\mathbf{q}_k\|} \right\|_2 \tag{4.6}$$

The factor κ scales the loss between euclidean distance and orientation error to be approximately equal, $\mathbf{q}$ is in quaternion representation in order to avoid problems of Euler singularities in the global coordinate frame; therefore the set of rotations lives on the unit sphere. During training, the values of $\widehat{\mathbf{q}}$ and $\mathbf{q}$ become close enough to be negligibly compared to the euclidean distance, consequently the constant κ play an essential role on the accuracy of the net.

Training individual networks to regress position and orientation separately have inferior performance compared to the proposed full-pose, the network was not able to determine a function to predict the vehicle position sequentially. This problem can be addressed as a Multitasking learning approach [12] where inductive knowledge transfer improves generalisation by sharing the domain information between complementary tasks. As a preliminary setting, we use the approach of [40] where a constant κ was tuned using grid search.

Nevertheless, we observed that the error in orientation and position was related not only to κ but also with VSA. Selecting a constant κ value with low orientation error in small VSA rates (β), increased the orientation error on high VSA rates. When a new value was tested for high $\dot{\beta}$, the orientation error increased for low VSA rates. Fig.31 shows an iteration of κ values related to the associated VSA rate. Best value for almost zero $\dot{\beta}$ was found in $\kappa = 350$ while the best value for the higher rates (400 deg/s) was found to be around $\kappa = 1500$.

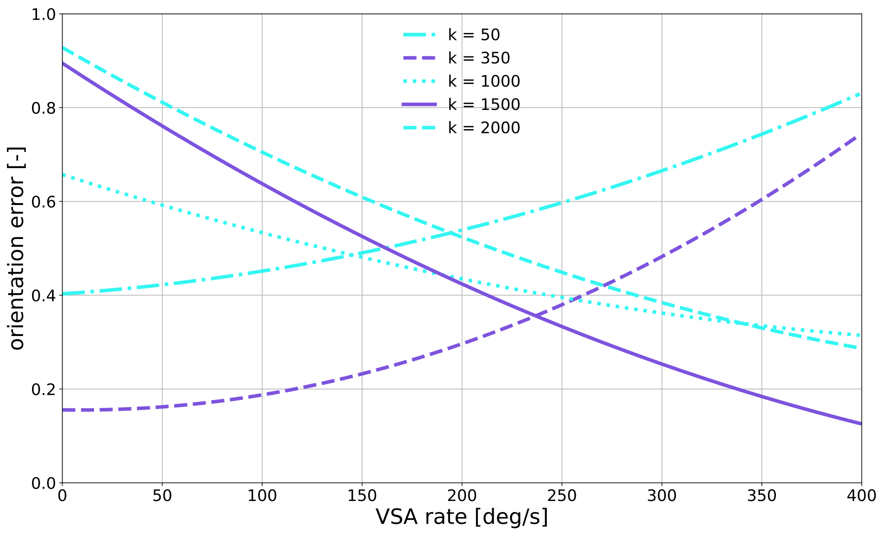


Figure 31: Orientation error for different values of κ related to the VSA rate ($\dot{\beta}$). Best values of κ for low and high $\dot{\beta}$ are showed in purple.

This results lead to the idea of a self-tuned κ depending on $\dot{\beta}$ which is as shown on Fig. 29 also predicted by the net. In order to adapt the scale value, a Gaussian function is proposed as follows:

$$\kappa(\dot{\beta}) = b - a e^{-\left(\dot{\beta}-\mu\right)^2/2\sigma^2} \tag{4.7}$$

where κ is the loss scale value, $\dot{\beta}$ is the VSA rate, b shift vertically the function in order to get the minimum value of κ when $\dot{\beta}$ is close to zero, a is the amplitude of the function, σ is the standard deviation and μ the mean of the normal distribution. σ and μ are tuned looking for the smaller orientation error with respect to $\dot{\beta}$. For the experiments, best values where found around $a = 2300$, $b = 2000$, $\mu = 0$ and $\sigma = 320$.

The final position regressor layer is randomly initialised so that the norm of the weights corresponding to each position dimension was proportional to that dimension's spatial extent.

Since the MIG database is more extent in the number of samples, the first net was trained using only this database. In order to train the AutoMiny net, the same architecture was used. To achieve better accuracy, the feature map of the previously trained GALNet for MIG was used, the layer group which determines the dynamic parameters is removed, as well as the last layer which composes the relative movement.

Finally, the feature map obtained from the second training is used to fine-tune the MIG net model; this exploits the associations made by the agent under the training in the AutoMiny dataset.

4.3 Experimental Results

In this section, we briefly summarise the results of the proposed architecture for dynamic ground localisation. We employ the ATE and RPE metrics from which we spoke briefly in 2.4.

4.3.1 *Evaluation Algorithms*

In order to evaluate the proposed model, we compared the results with two traditional methods.

The first algorithm is an Ackermann model odometry which is based on the wheel velocities, and an estimated Yaw rate obtained from the MIG's ABS subsystem. This method was developed for the autonomous car, since, historically, only the differential velocities of the back wheels were used to determine the direction and displacement of the car. The major drawback is that the wheel ticks have a systematic error that depends, among other things, on the wheel pressure. This physical event provokes the estimated trajectory to drift. Hence, the yaw rate information from MIG's ABS subsystem (MIGYawAndBrakePressure) was involved. The ABS subsystems already compensate for the deviation, and the estimation is, therefore, more accurate. In the following analysis, this algorithm is identified as Wheel Odometry (WO).

The second algorithm is a self-implemented UKF estimator which integrates the WO and IMU measurements from the car inertial unit. The filter is obtained employing the ROS robot-localization package [55].

As described in the previous section, the net was first trained on the MIG dataset, and the feature map was then used to train the net with the AutoMiny dataset, we differentiate the networks in the evaluation as GALNet and iGALNet correspondingly.

4.3.2 *Quantitative Evaluation of Trajectories*

In order to size the errors, on the final trained neural network, Table 10 shows the performance of the net with the ATE metric and Table 11 shows the RPE for the MIG trajectories.

Table 10 shows that in some trajectories GALNet and iGALNet improve the baseline methods. It is interesting to notice that not in all cases iGALNet performs better than GALNet, which means some associations are lost in the over-training process.

Table 11 shows that for the MIG, local displacements are still better estimated by basic wheel odometry. This result is explained since most of the short trajectories that are sampled at 30Hz and have a mean displacement of 0.5m are almost straight lines. Therefore, integration of the angular wheel velocity is still a better approximation

METHOD	WO	UKF	GALNET	IGALNET
FU_to_OBI	86.87	**42.51**	88.48	43.43
safari_online	428.31	**4.60**	483.21	50.00
thielallee	55.97	13.84	199.5	**11.87**
englerallee	132.07	167.5	**9.20**	46.61
react4	24.33	19.10	67.65	**10.39**
reinickendorf	6.47	3.65	2.84	**2.46**
auto7	211.96	17.97	89.86	**12.36**
auto8	17.35	**3.73**	46.73	4.62
tegel	269.55	**4.48**	219.55	23.72
back2fu	506.92	**7.15**	126.48	18.33

Table 10: ATE translational error in meters of the MIG dataset before and after including the AutoMiny dataset in the training. The methods used to compare are: MIG Wheel Odometry (WO), Unscented Kalman Filter (UKF), GALNet trained only with the MIG dataset, and the improved version (IGALNet) trained with the AutoMiny dataset.

METHOD	WO	UKF	GALNET	IGALNET
FU_to_OBI	**0.43**	0.49	0.71	0.59
safari_online	**0.27**	0.28	1.26	0.45
thielallee	**0.41**	0.51	0.79	0.60
englerallee	**0.44**	0.58	0.77	0.74
react4	**0.43**	0.49	0.70	0.55
reinickendorf	**0.51**	0.63	0.70	0.65
auto7	**0.39**	0.48	0.89	0.60
auto8	0.31	0.28	0.42	**0.27**
tegel	**0.95**	1.09	1.49	1.34
back2fu	**2.77**	3.09	4.04	3.13

Table 11: RPE translational error in meters of the MIG dataset before and after including the AutoMiny dataset in the training. The methods used to compare are: Mig wheel Odometry (WO), Unscented Kalman Filter (UKF), GALNet trained only with the MIG dataset and the improved version (iGALNet) trained with the AutoMiny dataset.

for straight driving. Wheel odometry has its major disadvantage on estimating orientation, in a global trajectory, orientation errors are collaterally translated to translational errors; for that reason, wheel odometry performs worse with a ATE metric. For some datasets, the precision of the UKF is achieved with the proposed methods.

Obtaining ground truth is a difficult task; the most reliable sensor on the MIG, which provides position estimation, is the Applanix system, as reported in Fig. 3 and Tab. 3. The error is provoked among other factors, for the number of satellites available at the moment of driving. Therefore, only the trajectories with available DGPS are used as ground-truth for the evaluation. For this same reason, odometry may be corrected by the GPS from time to time, causing the localisation to jump drastically, making the trajectory not differentiable. Therefore, such windows of time were discarded from the dataset, and the net was trained to estimate displacement between two consecutive relative positions. In this sense, it is interesting to examine the RPE metric deeper and observe how the error develops since the GPS adjustments are represented as outliers. For this analysis, we selected the Safari trajectory. Figures 32 and 33 show the RPE violin histograms of the translational and rotational components of the relative transformations.

Fig. 32 corroborates the values in Tab. 10, where the WO estimates more effectively the translational displacement, while its biggest error is around 3m the iGALNet goes up to 9m, the errors accumulate closer to zero than the other algorithms. It also shows the improvement of iGALNet against GALNet. Even that the purpose of training on the AutoMiny dataset was to reduce rotational error focusing on VSA, translational error improved as well. However, this result suggests that a pure translational displacement dataset, generated with the AutoMiny would help the estimation.

In Fig. 33 the reduction of outliers from GALNet to iGALNet is more patent. The proposed deep neural network has a better rotational performance than the evaluation algorithms, which was the purpose of this work. The iGALNet network has the smallest outlier and its error distribution closer to zero than the rest of the algorithms.

The resemblance between the violin distributions of the different algorithms shows that the proposed neural network was able to find the correct associations to estimate composed displacement.

Nevertheless, the appearance of the iGALNet estimated trajectory in Fig. 37 is still less accurate than UKF due to the translational error.

4.3.3 *Qualitative Evaluation of Trajectories*

Although, the net was able to improve for small $\dot{\beta}$ and therefore contribute to the precision in the MIG dataset, the high rates in the Au-

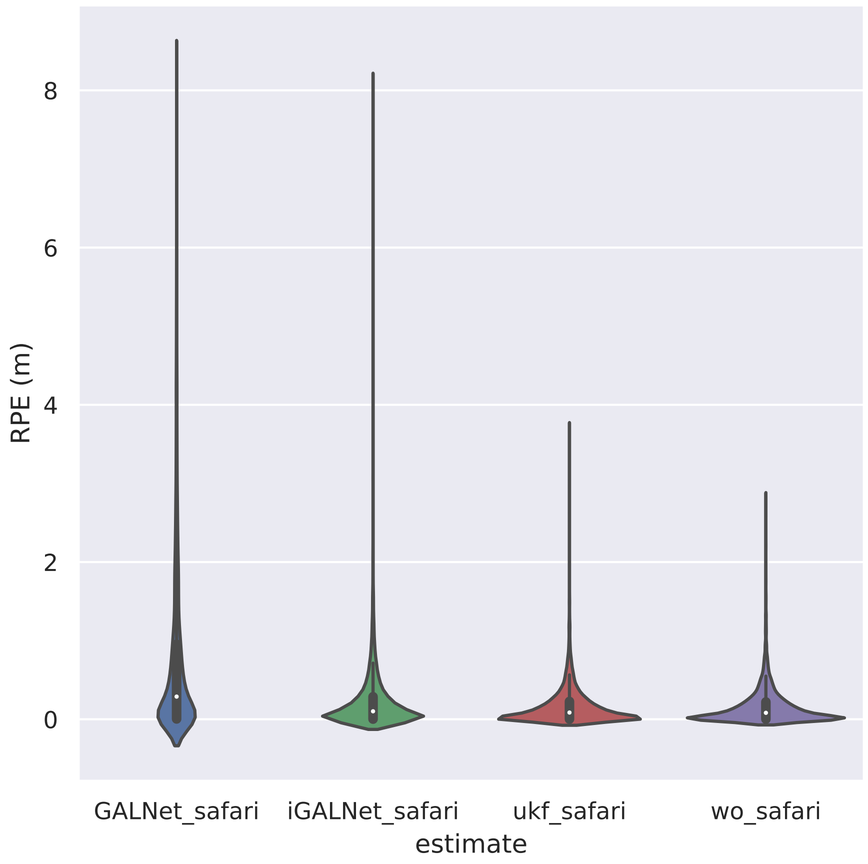


Figure 32: RPE violin histogram of the translational errors in meters in the Safari trajectory.

toMiny dataset showed limited precision. Fig. 34 shows two trajectories with $\dot{\beta}$ values up to $1.2rad/s$.

Figure 35 shows the trajectories of the MIG wheel odometry, UKF and GALNet before and after (iGALNet) being trained with the AutoMiny dataset in Thielallee, the official test area of the MIG in Berlin. Figures [36 - 44] show the rest of the resulting trajectories in the MIG dataset.

Finally, we show the resulted trajectories of each of the methods explored, which give a more visual evaluation of them.

The proposed method shows that, with enough information, a robust net can be trained. The major drawback is that the performance of the net is information-dependent, therefore, if the characteristics of the dynamic system changes, such as tire air pressure, wear, or vehicle weight (due to trunkload or number of passengers), the estimated position could be improved with more data collection. Therefore the system could be refined online on a test vehicle and update the weights of the net to avoid wide drifting errors.

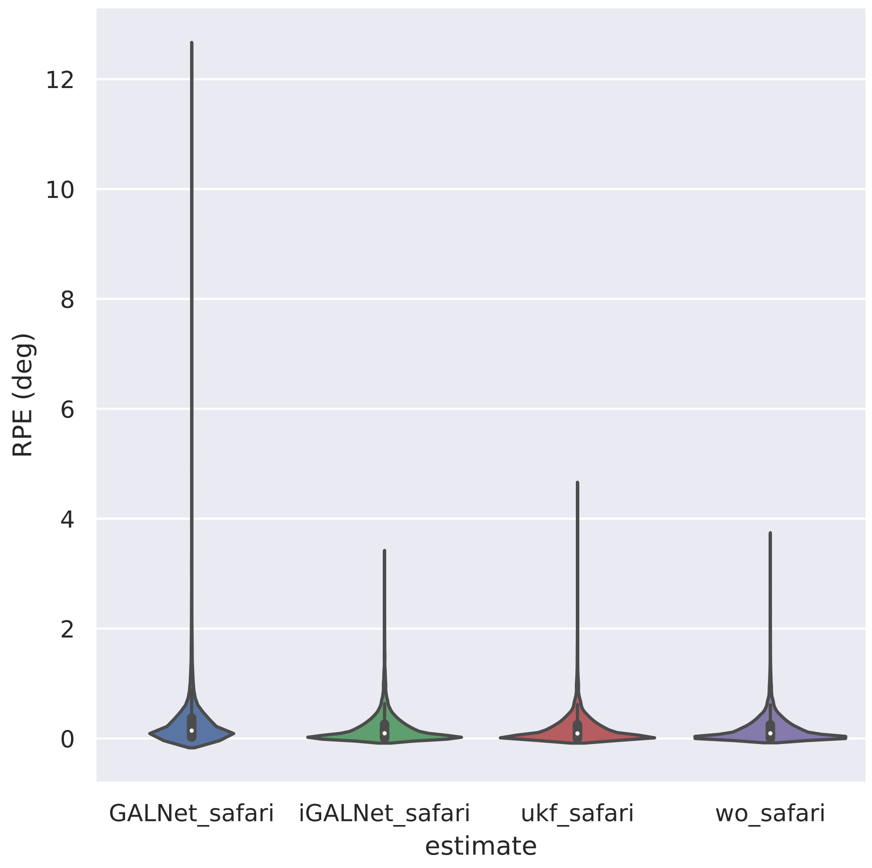
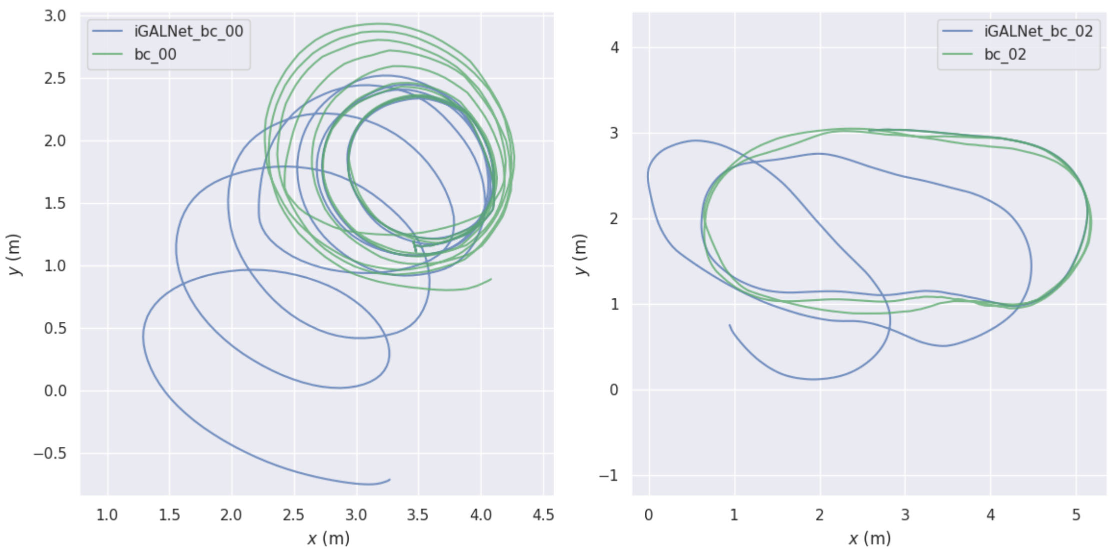


Figure 33: RPE violin histogram of the rotational errors in degrees in the Safari trajectory.

In general convolutional nets require a large amount of training data, and unlike the convolutional nets for computer vision, to our knowledge, there are not open source or available pre-trained nets for VSA, for pose estimation. Therefore, our net applies to our vehicle and must be finely tuned to be used on a different one. Creating a definitive neural network for every vehicle would need more data, and the weights must be fine-tuned for every vehicle since these dynamic parameters would be unique to each car.

Whereas the error between timestamps is small, the accumulated drifting error cannot be corrected only with information from the vehicle. Outside references are critical to correct the drift along the path. Therefore a visual localisation approach based on neural networks was developed to generate a net capable of giving positional information with cheap on-board sensors.



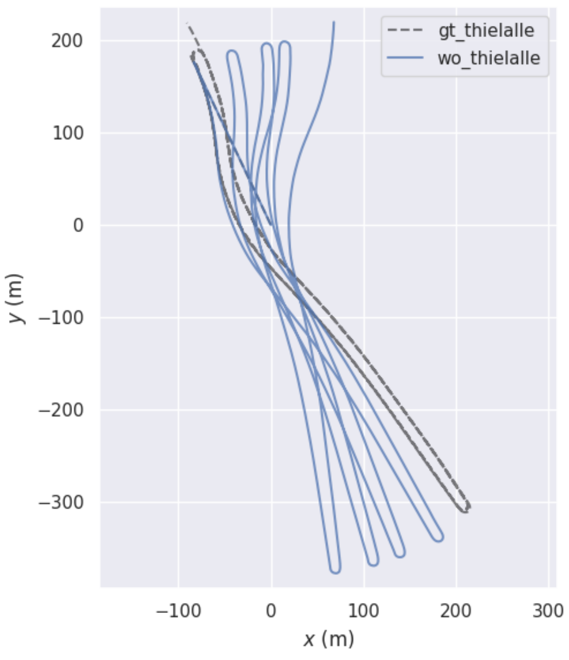
(a) (b)

Figure 34: a) Seq. 00 of the AutoMiny dataset, translational errors are: ATE=1.37 RPE=0.03. b) Seq. 02 of the AutoMiny dataset, translational errors are: ATE=2.16 RPE=0.01. On green the ground truth obtained with the ceiling cameras in the lab, in blue, the estimated trajectory of GALNet.

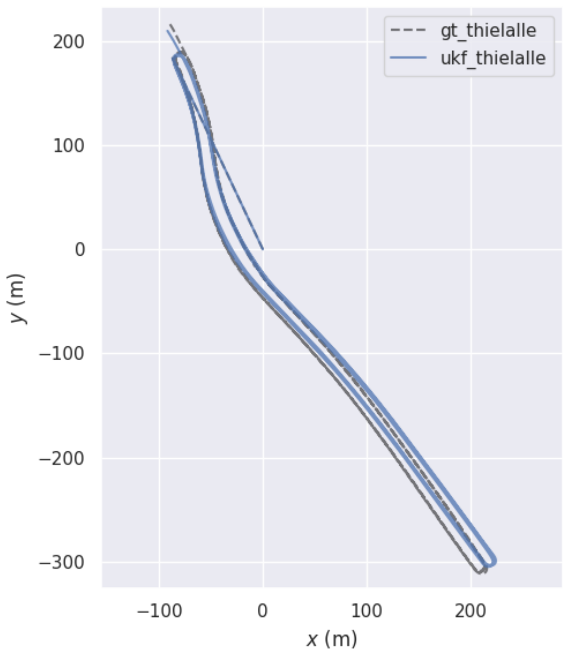
4.3.4 Reported Runtime

The network is implemented based on the TensorFlow framework and trained using an NVIDIA Geforce RTX 2080 ti. Adam optimiser is employed to train the network with starting learning rate 0.001 and parameters $\alpha_1 = 0.9$ and $\alpha_2 = 0.999$ both values recommended on the analysis in [42]. The training was set for 200 epochs but using callback Tensorflow implementations such as early stopping if the loss function does not decrease 0.001 for more than five epochs to reduce the training time.

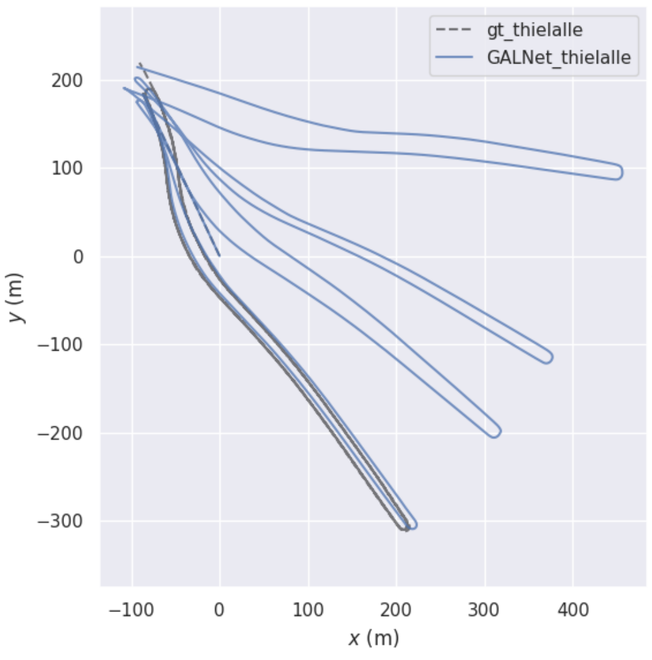
Training time on all the trajectories takes approximately 10k to 50k iterations, or 2 hours to 6 hours. Prediction time for an input vector pair takes on average 25 ms, i.e., 40Hz.



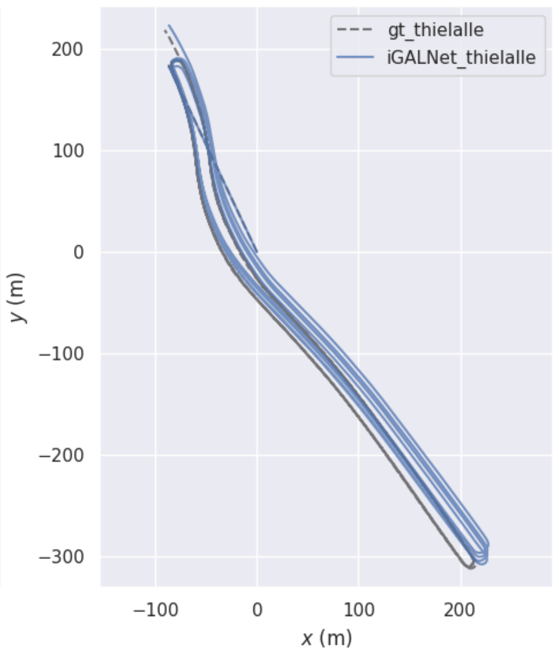
(a)



(b)

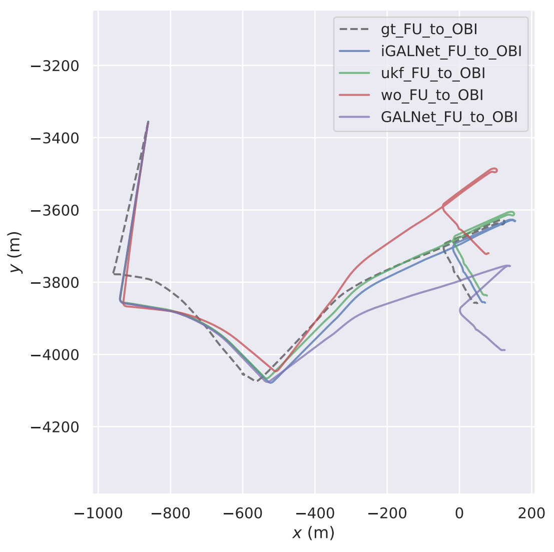


(c)

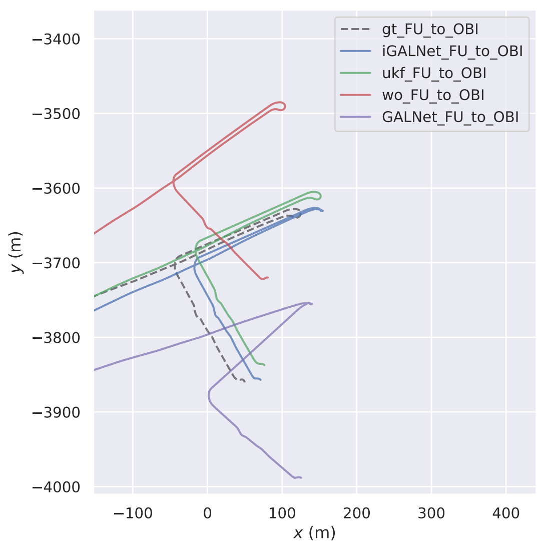


(d)

Figure 35: Resulting trajectories of the proposed methods in Thielallee. a) MIG wheel odometry, b) UKF wheel-inertial odometry, c) GALNet, d) iGALNet.

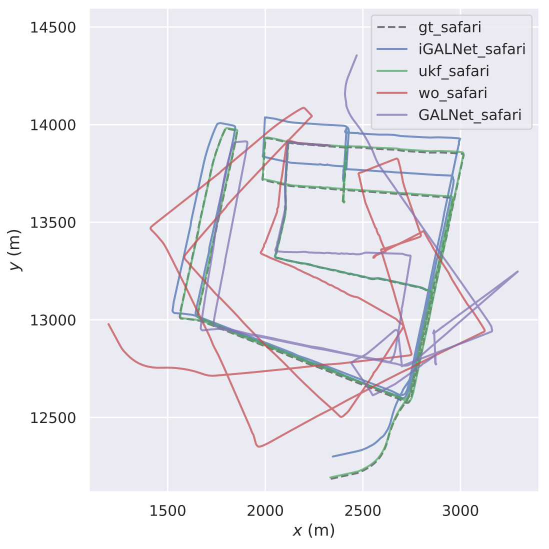


(a)

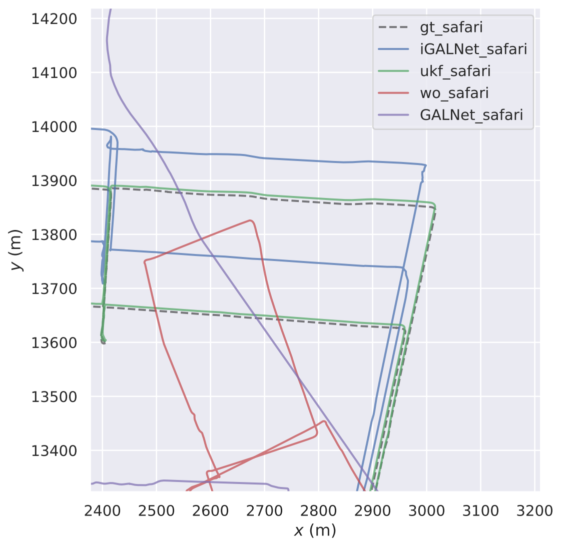


(b)

Figure 36: Resulting trajectories in FU_to_OBI dataset. a) Complete trajectory, b) Close up of trajectory.

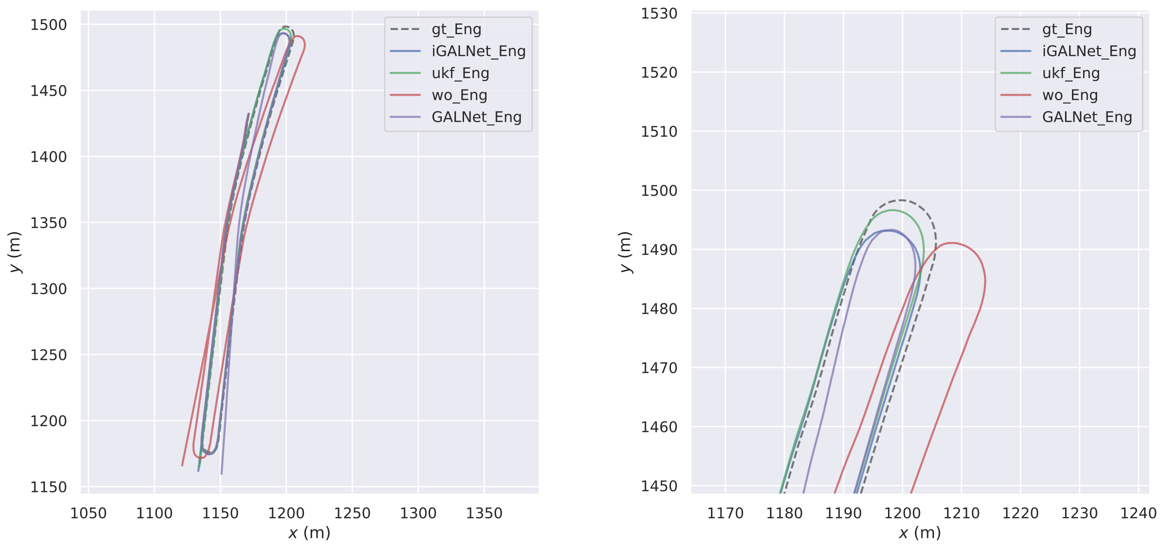


(a)



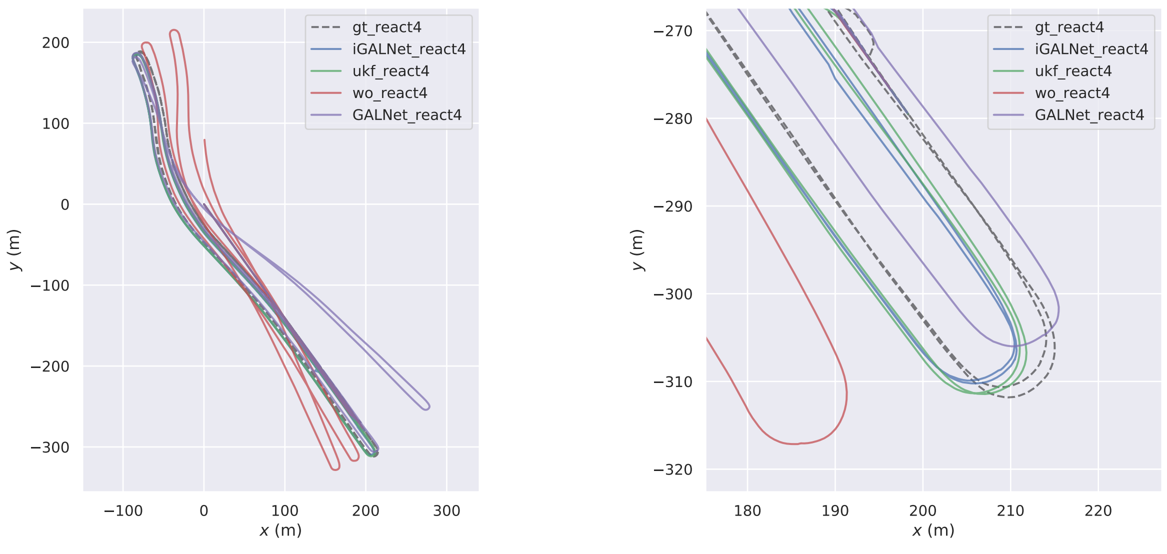
(b)

Figure 37: Resulting trajectories in safari_online dataset. a) Complete trajectory, b) Close up of trajectory.



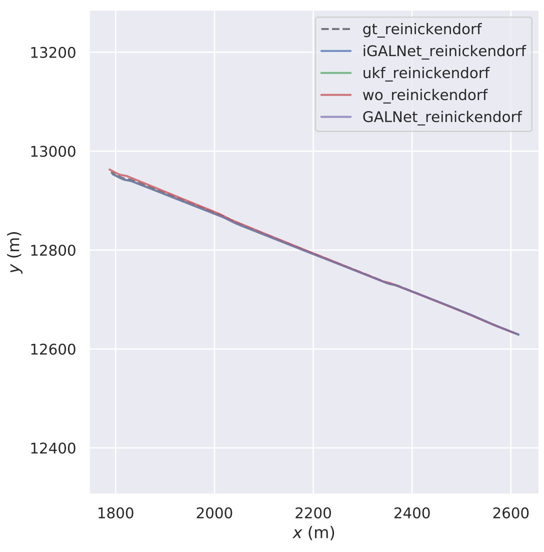
(a) (b)

Figure 38: Resulting trajectories in Englerallee dataset. a) Complete trajectory, b) Close up of trajectory.

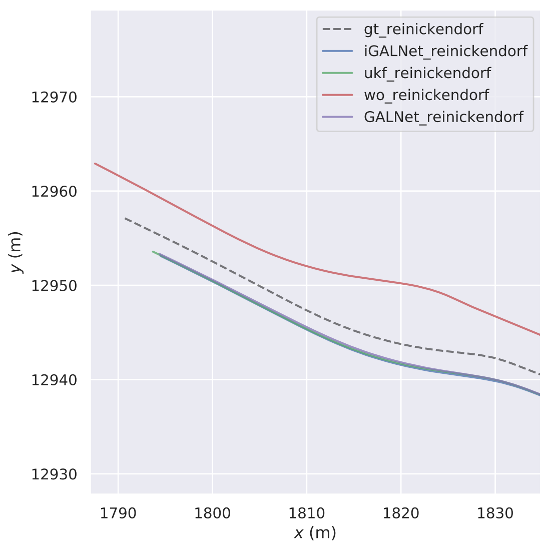


(a) (b)

Figure 39: Resulting trajectories in react4 dataset. a) Complete trajectory, b) Close up of trajectory.

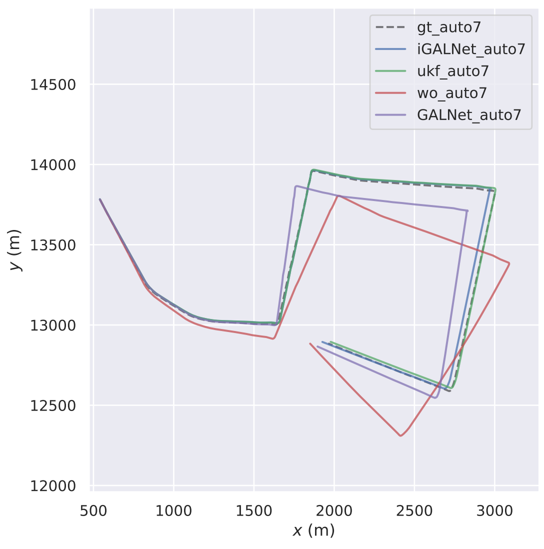


(a)

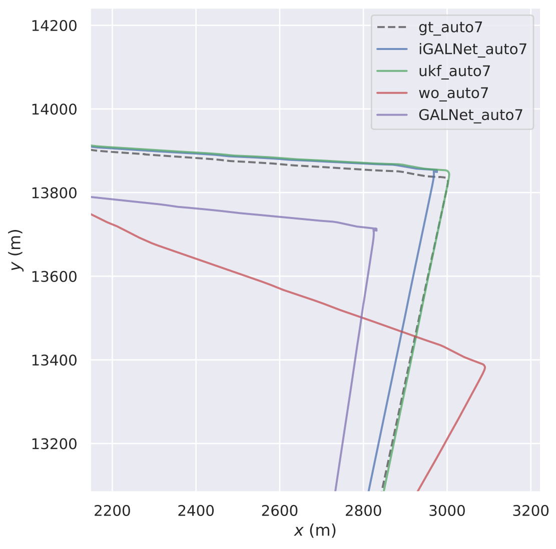


(b)

Figure 40: Resulting trajectories in Reinickendorf dataset. a) Complete trajectory, b) Close up of trajectory.

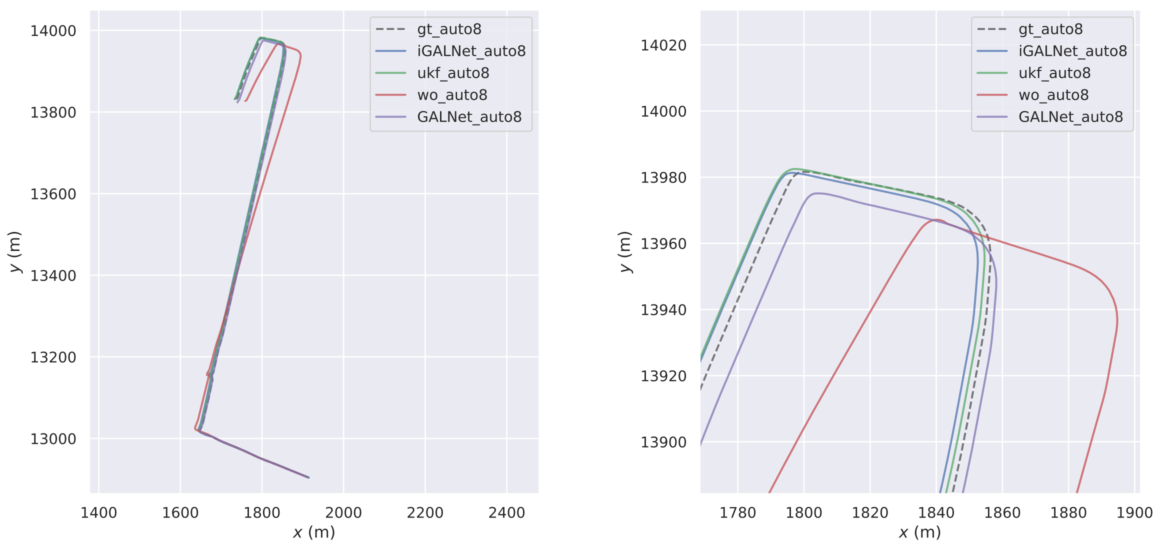


(a)



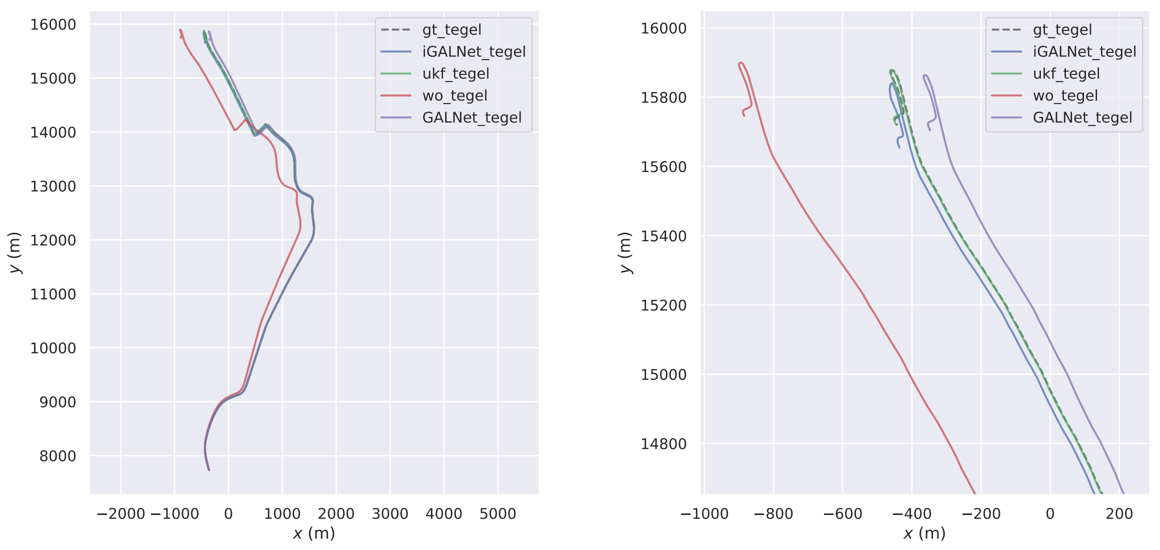
(b)

Figure 41: Resulting trajectories in auto7 dataset. a) Complete trajectory, b) Close up of trajectory.



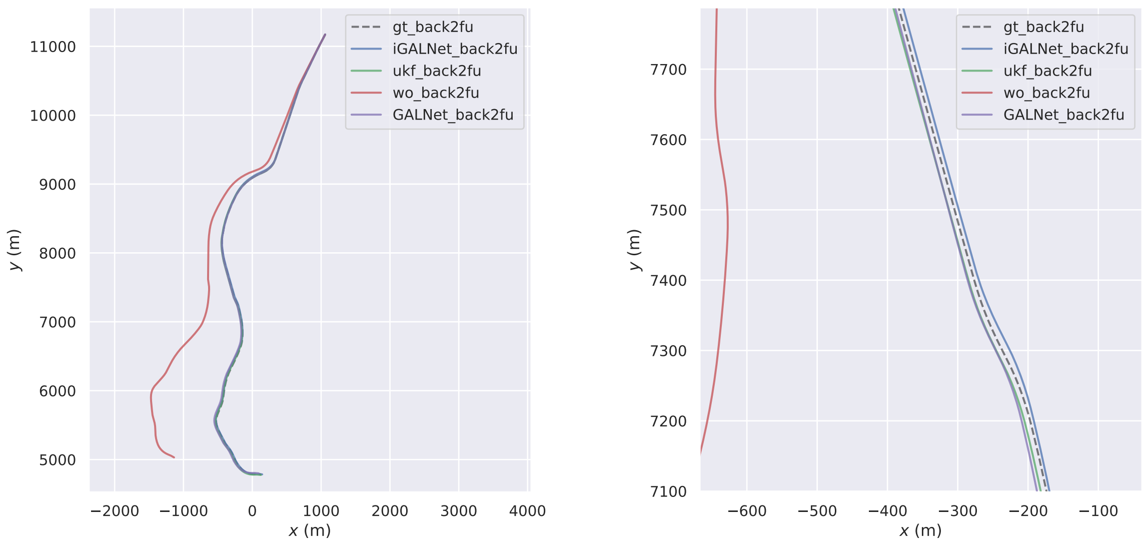
(a) (b)

Figure 42: Resulting trajectories in auto8 dataset. a) Complete trajectory, b) Close up of trajectory.



(a) (b)

Figure 43: Resulting trajectories in Tegel dataset. a) Complete trajectory, b) Close up of trajectory.



(a) (b)

Figure 44: Resulting trajectories in back2fu7 dataset. a) Complete trajectory, b) Close up of trajectory.

Chapter 5

Deep Learning for Visual Localisation

In this chapter, the problem of estimating the relative displacement in position and orientation between two consecutive frames is addressed, this is also colloquially known as visual odometry and is an essential part of the localisation systems in mobile robotics, autonomous vehicles, navigation and augmented reality.

As described in Chapter 2.3, designing a system for reliable large scale localisation is a challenging problem. Moreover, current visual localisation schemes perform accurately within controlled environments [76] but fail in the presence of highly dynamic objects in a scene or under difficult camera visual exposures, like extreme weather and changing lighting conditions. It is in this area where deep neural networks have overcome traditional methods.

Intending to explore the advisability of deep neural networks in visual localisation, this chapter proposes a framework that aims to calculate the geometric vehicle displacement between sequences of frames. The localisation system takes consecutive single RGB images and regresses the camera's relative 6-DoF pose. The net, named VALNet, uses an optical flow pre-trained feature map. It is shown that given a couple of optical flow feature maps, the network is able to estimate camera displacements.

We introduce a siamese architecture and a new loss function to train the regressor. We finally tested the algorithm in a recorded dataset while driving the MIG in Thielallee, the official testing track of the FU Berlin AutoNOMOS project. The remainder of the chapter evaluates the results obtained with the model.

5.1 Dataset Construction

In this section, the employed dataset to test the localisation algorithm is described. Training deep neural networks for computer vision applications often requires a very large amount of labelled images. Annotating ground truth labels on these datasets is often expensive or very labour intensive. Therefore, in addition to the own generated database, we decided to employ an open-source, public alternative.

The dataset construction consists of two stages. First, we define the employed databases, and afterwards, we show how the data is treated before being employed to train the neural network.

5.1.1 *Employed Datasets*

The experiments in this project will use two primary datasets: the KITTI [28] odometry dataset and a generated dataset employing the MIG autonomous car and a Stereo Labs®, ZED camera[1].

Obtaining ground truth is a difficult task; the most reliable sensor on the MIG, which provides position estimation is the Applanix system, as reported in Fig. 3 and Tab. 3, the error is provoked among other factors, for the number of satellites available at the moment of driving. Therefore, only the trajectories with available DGPS are used as ground-truth for the evaluation.

5.1.1.1 *KITTI Dataset*

KITTI is a collection of datasets collected with a focus on applications that are relevant to autonomous driving technologies, such as depth estimation, odometry, object tracking, semantic segmentation, optical flow, among others.

There are different datasets within KITTI for each of these applications; in this project, we will focus on the odometry dataset.

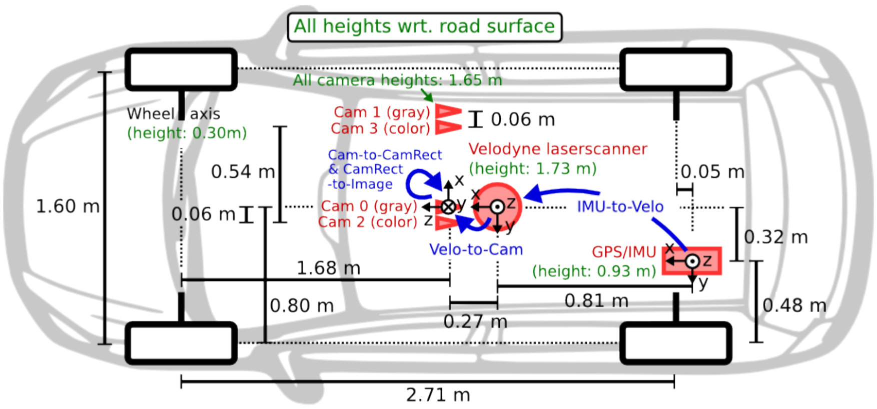
The odometry dataset is a collection of 22 sequences collected from an autonomous driving recording platform with advanced sensors to collect data streams such as RGB video, depth and LIDAR. The particular sensor used for ego-motion is a state-of-the-art OXTS RT 3003 localization system[2]. This system combines GPS, Global Navigation Satellite System (GLONASS), an IMU and RTK correction signals. The OXTS RT 3003 enables centimetre-level accuracy (open sky localisation errors < 5 cm) and provides the ground truth of the dataset. These sensors, as well as the MIG Applanix, are prohibitively expensive for widespread consumer applications (costing on the same order of magnitude as a car itself) and this provides one of the principal reason why visual odometry is still an active area of research despite the impressive accuracy of this sensor.

The images used in the KITTI dataset are collected from 4 Point-Grey Flea2 video cameras (2 colour cameras + 2 grayscale cameras), which have a resolution of 1392x512 pixels, and $90° \times 35°$ field of view.

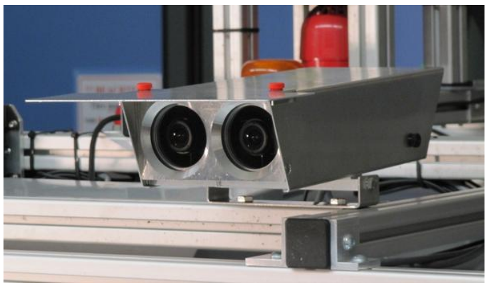
The camera array composes the stereo vision system (one RGB/Grayscale camera on each side) with 54 cm of distance between left/right camera, and 6 cm between the RGB/Grayscale cameras on

1 https://www.stereolabs.com/zed/
2 https://www.oxts.com/products/rt3000/

the same side. The ground-truth produced by the localisation system is projected into the coordinate system of the left camera. Figure 45 shows the arrangement of the sensors on the vehicle.



(a) Vehicle



(b) Camera set

Figure 45: Sensor arrangement in the Vehicle used to collect the KITTI dataset [26].

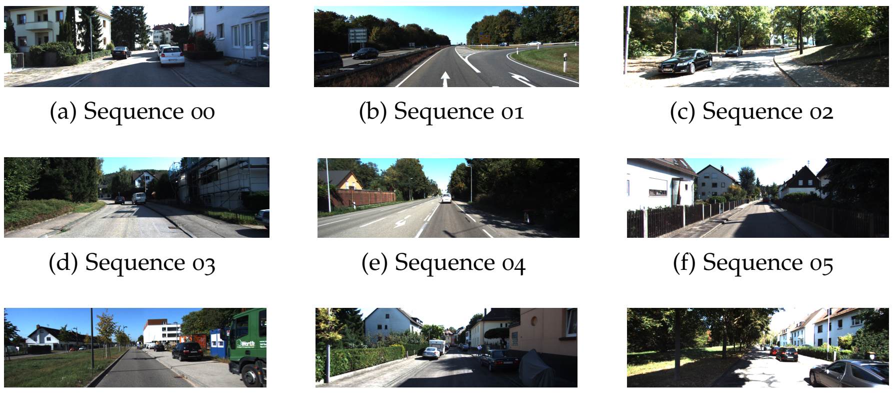
The odometry dataset is comprised of 22 sequences of a car driving outdoors in a variety of scenarios, including middle-size cities, roads, and highways. There are around 41,000 images collected in total, measured over a distance travelled of around 39.2 km. There are only available ground-truth labels for 11 of the 22 sequences, as the rest are used as testing data for a benchmark leaderboard on this task.

We do the train/test split of sequences [00, 01, 02, 03, 04, 08, 09] for training and sequences [05, 06, 07, 10] for testing. Fig 46 shows random images taken from the dataset.

The camera frames are obtained at a 10 Hz rate. It is essential to take into count that it is not possible to drive the vehicle at a constant velocity, which could affect the training directly when the net is tested on sequences that were obtained at a different velocity or in a velocity that is not well represented in the database. Table 12 shows the mean and max velocities of the KITTI vehicle for the different sequences.

5.1.1.2 *AutoMiny-MIG Dataset*

Additionally, a dataset was obtained driving the MIG along Thielallee.



(a) Sequence 00 (b) Sequence 01 (c) Sequence 02

(d) Sequence 03 (e) Sequence 04 (f) Sequence 05

(g) Sequence 06 (h) Sequence 07 (i) Sequence 08

Figure 46: Kitti odometry dataset - over 41, 000 images from 22 sequences.

SEQ.	σ	μ	SEQ.	σ	μ	SEQ.	σ	μ
00	9.74	29.53	*01*	18.75	80.28	*02*	8.39	39.14
03	8.20	25.23	*04*	3.72	52.48	*05*	10.50	28.76
06	10.45	40.34	*07*	11.53	22.73	*08*	10.20	28.50
09	9.41	38.60	*10*	12.04	27.58	*Th.*	3.88	52.32

Table 12: Velocity of the KITTI dataset and Thielallee sequences in Km/h. σ is the standard deviation and μ is the mean velocity.

Although the car has four TE SatCam RGB colour monocular cameras integrated into the chassis, the fisheye lens of the cameras make them not suitable for the pre-trained FlowNet2 neural network since the distortion model of the images would decrease the performance of the estimation. Therefore, in order to capture appropriate images, we used a ZED camera which was installed on the windshield of the MIG as shown in Fig. 47.

The main characteristics of the camera are listed in Table 13.

Resolution	Image Resolution[px]	Focal Length[px]	Pixel Size	Field of View (V-FOV, H-FOV)
HD2K	2208 x 1242	1400	0.002mm	47°,76°
HD1080	1920 x 1080	1400	0.002mm	42°,69°
HD720	1280 x 720	700	0.004mm	54°,85°
WVGA	672 x 376	350	0.008mm	56°,87°

Table 13: ZED Camera features.

The ZED camera is Universal Video Class (UVC) compliant and is possible to capture the left and right video streams without the ZED

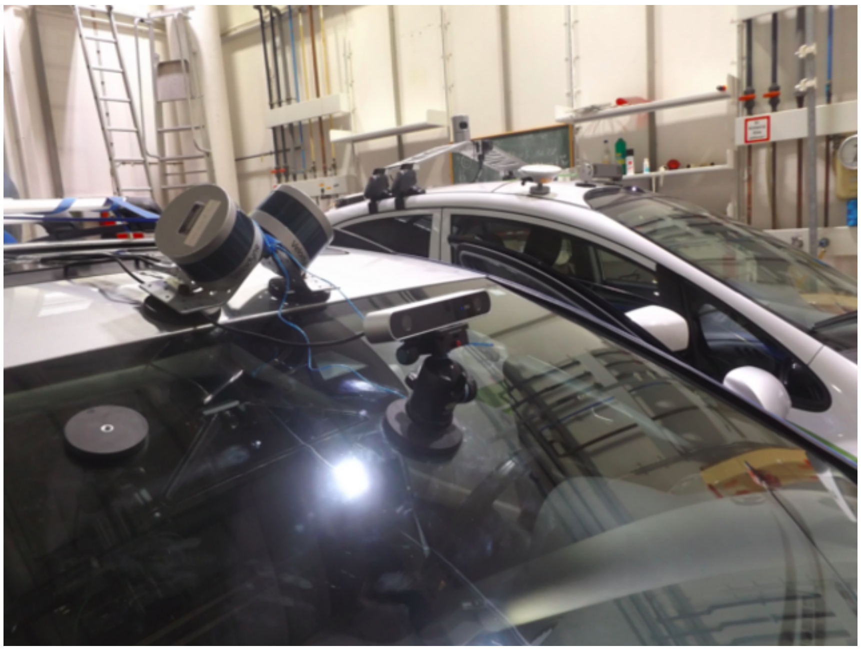


Figure 47: ZED camera mounted on the MIG windshield.

SDK. However, building applications and using the ZED SDK require CUDA to compute the depth map in real-time on the GPU. Video streaming and recording utilities are part of the ZED SDK, and therefore, we employed the NVIDIA TX1 arm GPU of the AutoMiny TX1. Fig. 48 shows the AutoMiny inside of the MIG, the camera is connected through USB 3.0 to the AutoMiny TX1.

The ZED camera has two running modes in order to save the sequential images.

The first tested mode runs the camera as a ROS node. The AutoMiny TX1 works under a ROS framework (see section 3.2), which makes possible to configure the scaled car as part of the ROS network of the MIG by simply connecting the Ethernet port of the AutoMiny with the MIG router. In this way, the ZED ros-package is executed in the TX1, and the topics are broadcasted to the network making possible to record a ros-bag including the localisation data published under the topic: */localization/odometry/filtered_map*. Fig. 49 shows the ros-rviz graphic interface with the available data while driving along Thielallee.

Video streaming can be recorded at different frame-rates depending on the configured resolution. Nevertheless, the frame-rate reported on the system while running other nodes for the MIG were slightly different from the factory registers. Even more, while recording the ros-bag, the frame-rates dropped dramatically, resulting in a deficient quality of the dataset. Table 14 shows the available schemes and the reported frame-rates.

In order to test the results of the neural network on Thielallee, it was necessary to have the images and frame-rate at least at the KITTI

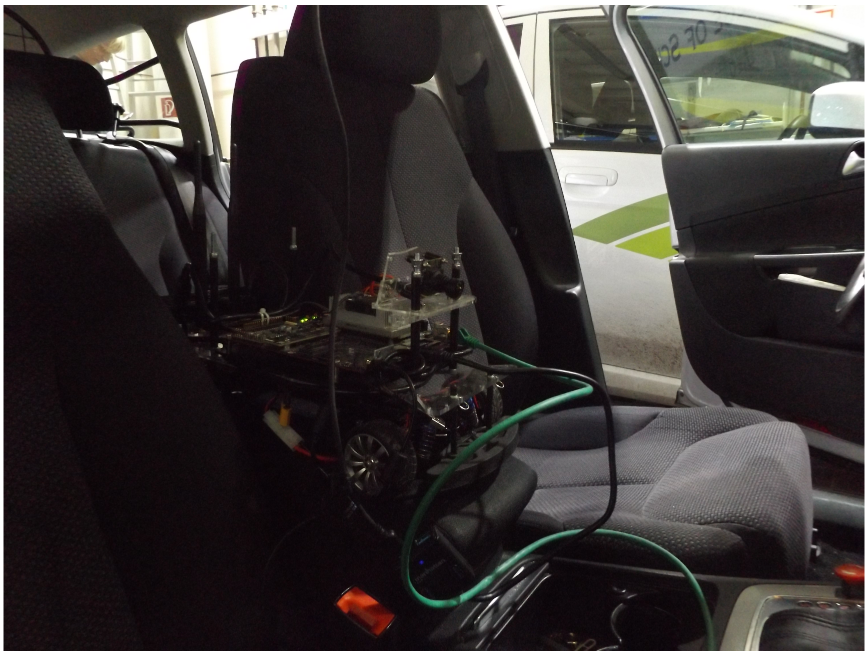


Figure 48: AutoMiny Tx1 mounted in the MIG to record the dataset, the car is connected through Ethernet to the MIG.

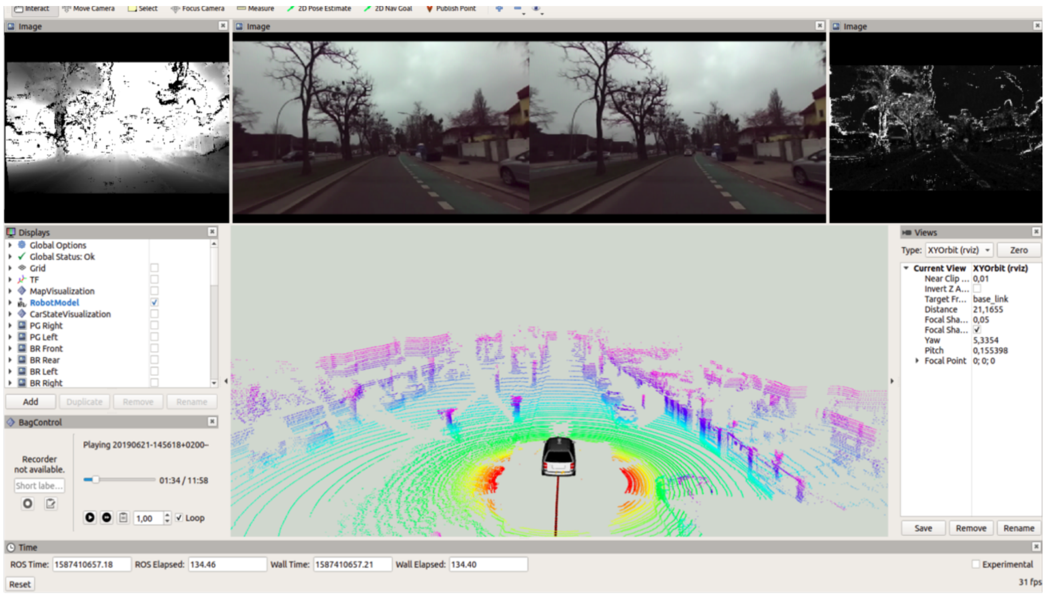
VIDEO MODE	FACTORY FPS	OUTPUT RESOLUTION	RUNNING FPS	FPS WHILE RECORDING
2.2K	15	4416x1242	13.9	8.4
1080p	30	3840x1080	28.2	16.1
720p	60	2560x720	59.3	35.3
WVGA	100	1344x376	98.6	66.5

Table 14: Frame rates reported while running and recording a rosbag with the ZED camera as a ROS node in the MIG network.

dataset scheme, i.e. 10 Hz and resolution of 1392x512 pixel, therefore 720p was the target resolution on the ZED camera. Although one lap was driven with each resolution setting for experimental purposes.

The second tested mode to save the video stream is a particular proprietary datatype of Stereo Labs®, the "*svo*" files. These type of files can save the frames at a factory frame-rate, virtually without recording delay. Unfortunately, the frames are saved ruled only with a Linux timestamp, there is no direct relationship with the ROS clock, and in order to synchronise the data with ros-packages a post matching effort must be made.

To record the ground-truth positional data from the MIG together with *svo* video streams it was necessary to synchronise the Linux system clock between platforms. A script uses the ZED SDK to start and stop the video and ros-bag recording. The *svo* file is saved locally



Playing 20190621-145618+0200–
01:34 / 11:58
1,00 ✓ Loop

Type: XYOrbit (rviz)

Current View	XYOrbit (rviz)
Near Clip ...	0,01
Invert Z A...	☐
Target Fr...	base_link
Distance	21,1655
Focal Sha...	0,05
Focal Sha...	✓
Yaw	5,3354
Pitch	0,155398
Focal Point	0; 0; 0

ROS Time: 1587410657.18 ROS Elapsed: 134.46 Wall Time: 1587410657.21 Wall Elapsed: 134.40

31 fps

Figure 49: **MIG rviz view on Thielallee.** On the top, from left to right, the depth, left, right images and the confidence map obtained from the ZED camera. On the bottom, the car robot model and the point cloud from the LIDAR.

in the AutoMiny SSD, in this way, is not necessary to broadcast the information through the network. The pose information in the rosbag is recorded in the main laptop which runs the ros-packages of the MIG.

The dataset is stored with the Pandas[3] data frame.

5.1.2 *Visual Odometry Dataset Augmentation Tool*

Along with the widely spread applications of computer vision for classification tasks, data augmentation has proven to increase the accuracy and precision of the networks. Is usual to find in deep learning frameworks like Keras, tools which generate synthetic images from a sample image through performing operations such as: perspective and homogeneous transformations, zoom-in, zoom-out, noise-additive and adjusting brightness, among others. This technique generates potentially thousands of images from a single one.

For time series problems, data augmentation is still a research subject. Being visual odometry with end-to-end deep neural networks a not so well know problem, is difficult to make a consensus about the operations that improve estimation accuracy.

We propose a data augmentation tool for visual odometry. Our data augmentation tool can potentially generate thousands of sequences

3 `https://pandas.pydata.org/pandas-docs/stable/index.html`

from a single video. Fig. 50 shows how the segmentation was carried out.

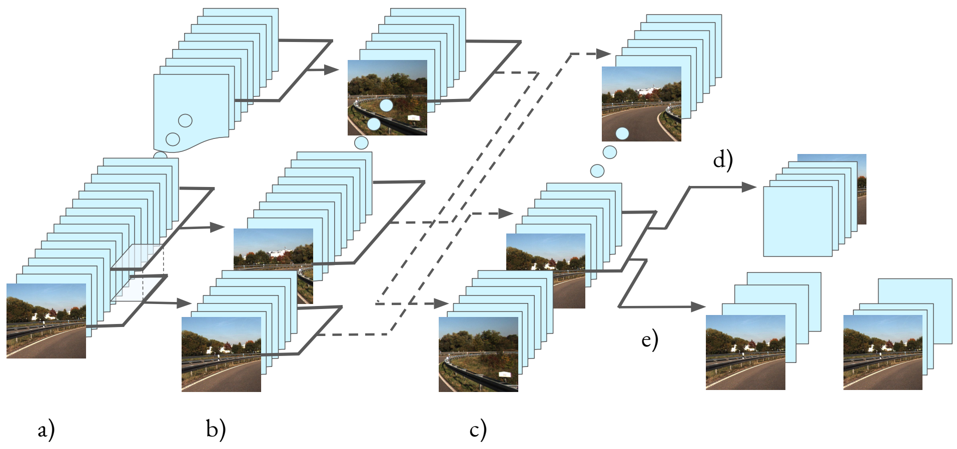


Figure 50: **Dataset augmentation.** a) Complete video frames, b) Sub-sequences of equal or different size, taken from a), sub-sequences can overlap between each other, c) Obtained sub-sequences can be shuffled every epoch to avoid bias in the training, d) The sub-sequence can be reversed, e) Some frames inside the sub-sequence can be skipped with a constant or variable step. The generated sequences must be correlated to the correspondent pose and orientation information, and modified with the proper $SE(3)$ transformation parameterizations. Every sub-sequence related trajectory is modified to represent a global trajectory within the boundaries of the frame recorded positions.The data augmentation tool is available in: `https://github.com/richrdcm/visual-odometry-data-augmentation-tool.git`.

The net trained on the generated sequences with the data augmentation tool showed to perform better.

5.2 Model for Visual Localisation

5.2.1 *Architecture Description*

The design of the architecture is partly based on the original FlowNet2 [24], tuned to take as inputs simultaneously the paired images in sequence (I_t, I_{t+1}), to regress the parametrisation position and quaternion labels $(\Delta X, \Delta Q)$. This parametrisation is simple to deal with but comes with a disadvantage that the quaternion requires normalisation to reduce an extra degree of freedom. The normalisation can cause optimisation issues on regression. Nevertheless, the representation avoids the singularities of other parametrisations.

In our approximation, we looked for a solution that could predict the ego-motion of the vehicle given a continuous analysis of two consecutive images, exploiting all the information available in the image. The image featurisation of the dataset presents several difficult challenges, among which are: data sparsity, dynamic objects, difficult lighting and object distortion due to vehicle velocity. Regarding the featurisation, in the current state of the art (see Chapter 2.3) there are two main approaches, object recognition or motion estimation. Since object recognition depends on nets trained with objects which may be or not part of the driving scenario, this approach would depend highly on the scene object diversity and on detecting those objects correctly among the first and second frame. Not to mention that problems like data association should be solved with the net.

We decided to focus our approach to motion estimation and develop a CNN block able to extract dependencies from two consecutive frames. This approach is advantageous since it does not relates the foreknowledge of the scene structure and could generalise better for driving scenarios.

The FlowNet architecture [24] was one of the first to explore learning optical flow using a deep convolutional network. The network was trained with several real and synthetic datasets. FlowNet consists of a convolution phase and a deconvolution phase. The main structure of the FlowNet used to extract the high-dimensional is shown in Fig.51.

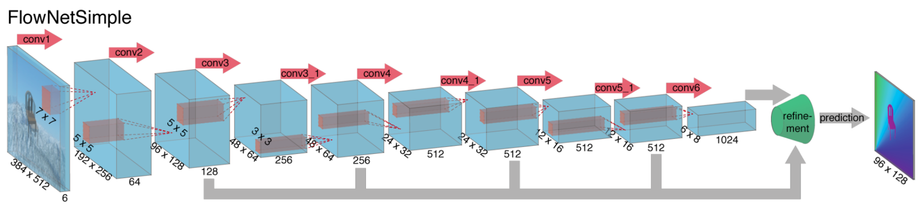


Figure 51: **FlowNet architecture.** Basic structure of the FlowNet, the refinement block on green, is a deconvolutional phase which transforms the feature map into an image representation of the optical flow [24].

.

The feature map from FlowNet can be obtained from the output of the last layer in the convolution phase. The net was trained under different datasets in order to reduce the error in the different particular applications, one of which is the KITTI dataset. Therefore, we decided to use such weights for our neural net.

FlowNet was extended in FlowNet 2.0 [36], where the basic structure of the architecture of FlowNet was trained under different circumstances, which improved the performance in small displacements. It was achieved by focusing on the training data and changing the strategy on the way of presenting data during training, the new architecture is a stacked architecture that includes warping of the second image with the intermediate optical flow as shown in Fig.52, finally a sub FlowNet specialised on small motions was added.

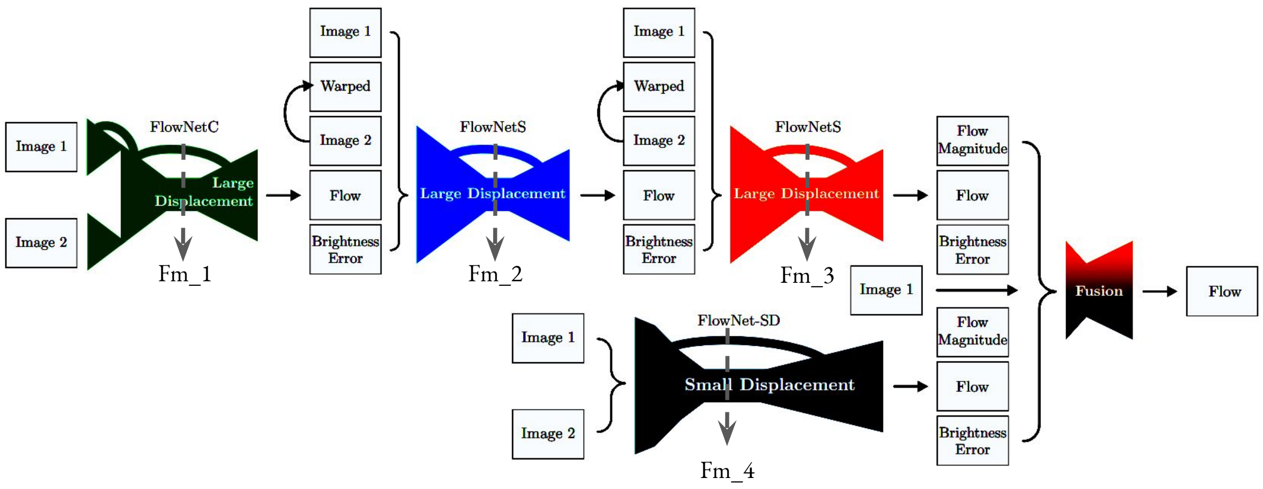


Figure 52: **FlowNet 2.0 architecture.** Structure of the FlowNet 2.0, a combination of multiple FlowNets. Braces indicate concatenation of inputs. Brightness Error is the difference between the first image and the second image warped with the previously estimated flow. To optimally deal with small displacements. A small fusion network provide the final estimate. Fm_1,2,3,4 represents the output of the last layer of the convolution phase of each FlowNet[36].

The feature map to train our net was obtained from the concatenation of the highest level flow features from the last layers of the convolutional phase of each FlowNet Fm_1,2,3,4. This produced a 6x20x1024 feature map for each subnetwork, after channel-wise concatenation, the total size of the feature map produced the final 6x20x4096 image pair flow features. This feature map, process the dependencies between two consecutive frames, in an analogy referring to a direct Odometry method showed in Fig. 13, the feature map delivers the necessary information to compute the motion estimation and optimisation modules. Subsequently, the feature map is flattened and used to create a sequence model, necessary to elaborate the input of the recurrent neural network layers of the proposed architecture.

The proposed visual odometry architecture is a Siamese network, each side of the network is composed by a convolutional block, followed by a LSTM block. The recurrent neural block used in the architecture is a twofold LSTM; every single layer has 1024 hidden units which processes inputs in a past-to-future direction. These initial layers extract useful characteristics for odometry from the optical flow feature maps and find the corresponding associations between timesteps. Both sides of the network share its convolutional-temporal outputs with the opposite side. This convolutional-temporal block extract associations between optical flow frames which allows the following fully-connected block to calculate the local displacement.

Each side of the network computes a different direction of the trajectory. This means the second branch of the net estimates the inverse ego-motion, given the reversed frame sequence.

Additionally, the estimated local poses of each branch are processed by a custom layer which computes the geometric transformations to obtain a global pose.

Since the transformations are complementary, we added an extra layer which computes the geometric transformations between the forward and backward sequences. The result of the net after the transformation should always be the starting point of the displacement, providing a closed trajectory on every batch, thus $\mathcal{X} = [0, 0, 0, 0, 0, 0]$. This attribute helps the net to learn back and forth features on every batch. We also run experiments with this additional property to evaluate the precision of the net. Figure 53 shows the proposed architecture.

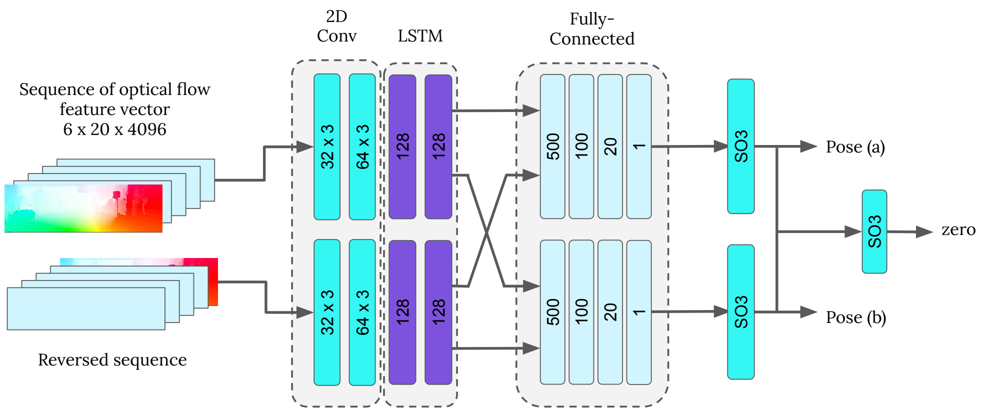


Figure 53: Siamese Architecture for Visual Odometry. In the first input, the sequence in a positive displacement is fed while in the second, the same sequence is reversed and fed. The network has two symmetric convolutional modules. The output is shared on the consecutive full connected layer modules. The final output is trained to be zero since it is the starting point of every sequence.
Note: the showed input optical flow images are merely illustrative, each feature map obtained from FlowNet2 have a size of 6 x 20 x 1024.

The aim of using a Siamese network is to create a self-controlled net, which balances the error among the three outputs. We prove that with a single sequence we are able to use the additional geometric attributes to balance the learning between the net branches, which guides the network to find similar estimations in each branch and the zero output will adjust the precision of the branch estimation. The convolutional block computes a transitional invariant feature map among the four high-level feature maps of the FlowNet2, meaning that the learned patterns could be recognised anywhere in the image. Since this block is trained to recognise back and forth displacements, after training.

To derive the global poses of the vehicle, the relative estimated positions obtained with the net, are composed through a custom layer which performs the transformation $\mathbf{T}$ in $\mathbb{R}^n$ that can be represented as an element in the special Euclidean group $\mathbf{SE}(n)$ as:

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix}$$

where $\mathbf{R} \in \mathbf{SO(n)}$, $\mathbf{t} \in \mathbb{R}^n$ is the translation vector. The transformation lies in the orthogonal group. The concatenation of two poses can be conducted by matrix multiplication in the context of $\mathbf{SE3}$ previous conversion of the quaternion representation into rotation matrix.

5.2.2 *Loss Functions*

In order to control de balance and precision of the net, the loss is computed on different steps.

Since the net minimize the loss through backpropagation, the total loss of the net is then computed using equation 5.1:

$$\mathcal{L} = \frac{1}{N} \sum_{N} (\beta_1 \mathcal{L}_{POSa} + \beta_2 \mathcal{L}_{POSb} + \beta_3 \mathcal{L}_{ZERO}) \tag{5.1}$$

were $\beta_1, \beta_2, \beta_3$ are the weighting factors for each output and N is the size of the batch.

The weights are defined as follows:

$$\begin{aligned} \beta_1 &= \frac{(\mathcal{L}_{POSa} + \mathcal{L}_{POSb})}{\mathcal{L}_{POSb}} \\ \beta_2 &= \frac{(\mathcal{L}_{POSa} + \mathcal{L}_{POSb})}{\mathcal{L}_{POSa}} \\ \beta_3 &= \{x \in \mathbb{R} | 0 \leqslant x \leqslant 1\} \end{aligned} \tag{5.2}$$

All weights in the network's convolutional layers have a gaussian initialisation, whereas the fully connected layers were initialised using the Xavier algorithm [45].

The position estimation loss was designed to compute a L2 (Euclidean) loss between the ground truth pose $\mathbf{y}_k = (\mathbf{p}_k^T, \mathbf{\Phi}_k^T)$ and its estimate $\widehat{\mathbf{y}_k} = (\hat{\mathbf{p}}_k^T, \widehat{\mathbf{\Phi}}_k^T)$. At time k can be minimized by

$$\mathcal{L}_{POS} = \frac{1}{|N|} \sum_{N} \|\hat{\mathbf{x}}_k - \mathbf{x}_k\|_2 + \kappa \left\| \hat{\mathbf{q}}_k - \frac{\mathbf{q}_k}{\|\mathbf{q}_k\|} \right\|_2 \tag{5.3}$$

were N is the size of the batch. LSTM layers were configured as stateful; therefore, the states of each sequence are propagated as the initial state in the following batch. The size of the batch is configured to be the length of the sequence.

The factor κ scales the loss between euclidean distance and orientation error to be approximately equal, $\mathbf{q}$ is a quaternion representation of the orientation in order to avoid problems of Euler singularities in the global coordinate frame. Therefore the set of rotations lives on the unit sphere. The scaling factor was selected as a constant; selecting the correct factor κ is crucial to prevent the net biasing the learning between position and orientation precision. The factor depends highly on the nature of the dataset characteristics, for instance, the step size, overlapping or sequence size. In order to select an acceptable initial κ value, we obtained the mean value of the distribution of the larger euclidean distance on each sequence. In order to find the most suitable factor, four additional trainings were conducted, the best κ was selected from Fig. 54.

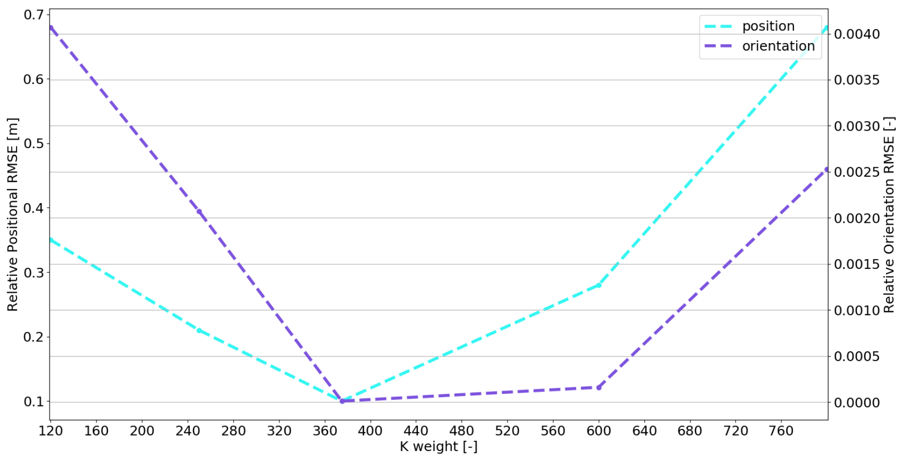


Figure 54: K scaling value for ESP-VO.

5.3 Experimental Results

In this section, we evaluate the proposed visual odometry network VALNet. We employ the ATE/APE and RPE metrics from which we spoke briefly in Chapter 2.4 for quantitative evaluation, and finally, some qualitative aspects of the results are discussed.

5.3.1 *Evaluation Algorithms*

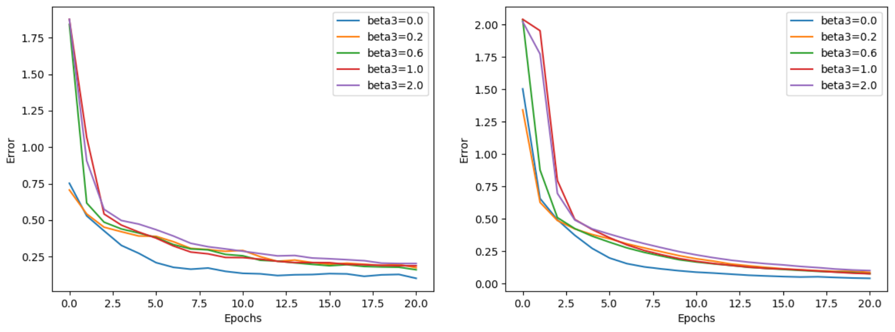
We compare the results with a self-implemented version of the ESP-VO [86] neural network with no covariance output, for more reference about the architecture of the network, we refer the lector to Section 2.3.2.

The well-known sparse feature Stereo ORB-SLAM2 algorithm is also used for evaluation. It is relevant to mention that we were no able to give an equal comparison of the results employing just the monocular visual odometry version of ORBSLAM since it was not able to estimate trajectories on the entire KITTI dataset or the Thielallee dataset without failing at a certain point in the trajectory. The localisation version of ORBSLAM has the loop closure and relocalisation modules off for which was difficult to reinitialise the estimator in a previously visited point or to optimize the trajectory. Also, monocular ORBSLAM drifts in scale severally. Therefore the stereo and full-SLAM version of ORBSLAM2 was employed. On the Thielallee dataset, we were not able to adjust the parameters of the feature detector, losing tracking. This may be due to the low contrast we have in the images due to the bad weather on the day we recorded the dataset.

Although we tried to provide evaluation results comparing our method with direct methods (Chapter 2.3) such as LSD-SLAM, it consistently losses tracking for KITTI dataset, one of the reasons is that the frames are captured at 10Hz, and the driving speed is in average 90 km/h. Aligning images by minimising photometric errors in a large baseline is still challenging for such methods. Therefore no parametric errors are provided.

5.3.2 *Neural Network Training*

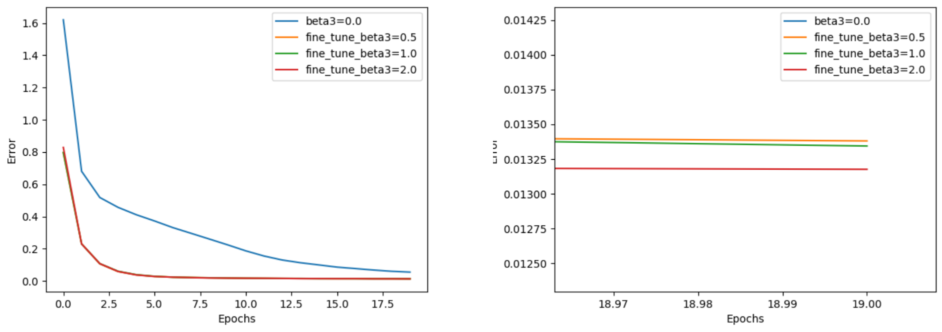
The Siamese net was trained setting different values for β_3 in Eq. 5.1, Fig. 55 shows the training and validation losses. From the results, it was clear that training simultaneously the three outputs with β_3 different of zero, produced a less precise output. Therefore, to train the network, there are necessary two stages. First, $\beta_3 = 0$ and the total loss relay on the balanced error between $\mathcal{L}_{POS_a}$ and $\mathcal{L}_{POS_b}$, this assures that the net is not going to be biased into one trajectory direction, and the convolutional feature map is balanced. In the second stage, the weights of the convolutional block are frozen, and β_3 is set to a higher value.



(a) Training loss (b) Validation loss

Figure 55: Siamese architecture training and Validation loss while setting different values for β_3.

Figure 56 shows the improvement of the net between the first and second training step. Three different values for β are shown. We concluded that higher values of β_3 would unbalance the Siamese behaviour and affect the accuracy of the results.



(a) Validation loss (b) Close up

Figure 56: Hyperparameter tuning of β_3. β_3 changes values from best Siamese trained model $\beta_3 = 0$ to $\beta_3 = 0.5, 1, 2$. Fine-tuning shows that freezing the convolutional blocks while fine-tuning the LSTM and dense layers by forcing the outputs to be complementary improve the accuracy. We have selected the net trained with $\beta_3 = 1.0$ since with $\beta_3 = 2.0$ the validation loss showed to start over-fitting.

5.3.3 *Quantitative Evaluation of Trajectories*

Tables 15 and 16 shows the comparison between performances of the ESP-VO, VALNet and Stereo ORB-SLAM2. It is clear that this last method is quantitatively better than the monocular neural network approaches. Nevertheless, the proposed architecture of this work overtakes the existing deep neural network approaches.

MODEL	03	04	05	06	07	10	Thielallee
ESP-VO	6.39	3.37	38.5	50.3	12.6	15.6	25.3
VALNet	5.8	2.7	34.2	48.4	13.2	14.6	21.3
Stereo ORB-SLAM2	**5.7**	**0.2**	**0.8**	**0.8**	**0.5**	**1.0**	**4.2**

Table 15: ATE translational error in meters of the tested methods on training sequences 03, 04 and KITTI testing sequences 05, 06, 07, 10, and Thielallee.

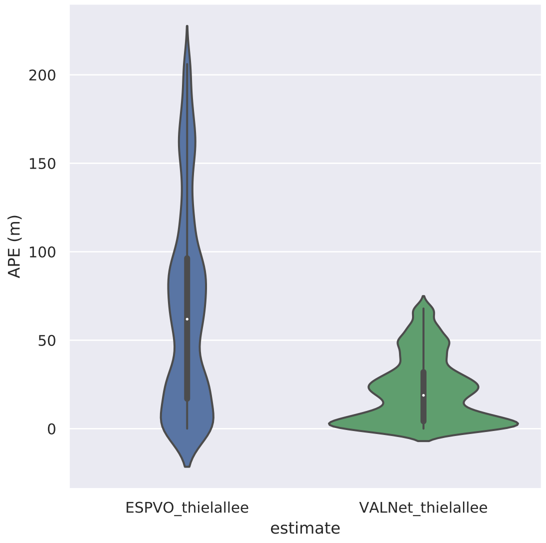
MODEL	03	04	05	06
ESP-VO	3.47e-2	1.03e-1	1.15e-1	1.07e-1
VALNet	1.5e-2	9.0e-2	1.4e-1	1.3e-1
Stereo ORB-SLAM2	**1.4e-2**	**0.99e-2**	**1.3e-2**	**1.1e-2**
MODEL	07	10	Thielallee	
ESP-VO	1.63e-1	2.08e-1	3.37e-1	
VALNet	1.1e-1	2.1e-1	3.32e-1	
Stereo ORB-SLAM2	**1.02e-2**	**1.3e-2**	**1.3e-2**	

Table 16: RPE translational error in meters of the tested methods on training sequences 03, 04 and KITTI testing sequences 05, 06, 07, 10, and Thielallee.

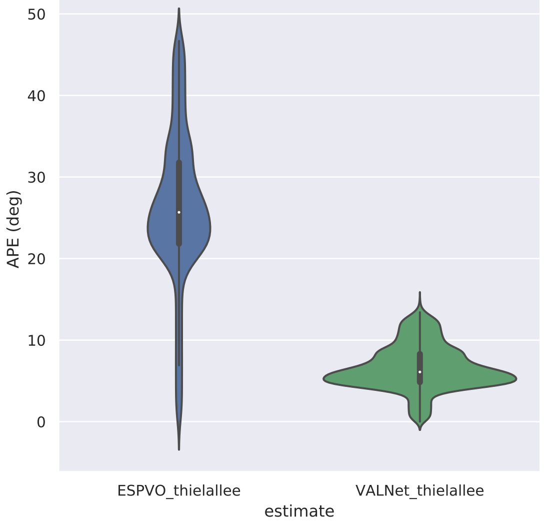
The KITTI dataset has 21 sequences. However, only ten have ground-truth data; therefore, there are only four KITTI testing trajectories, sequences 03 and 04 perform better since the trajectories are included in the training and therefore perform better on the ATE and RPE values.

In order to analyze the performance of the proposed neural network against the architecture of ESPVO, the ATE violin histograms are shown for the fifth Kitti trajectory and thielallee.

Fig. 57 shows the translational and rotational error components of the transformations in Thielallee. In both error components VALNet improves the performance, which explains the drift of ESPVO in Thielallee in Fig. 60f.



(a) Translational error

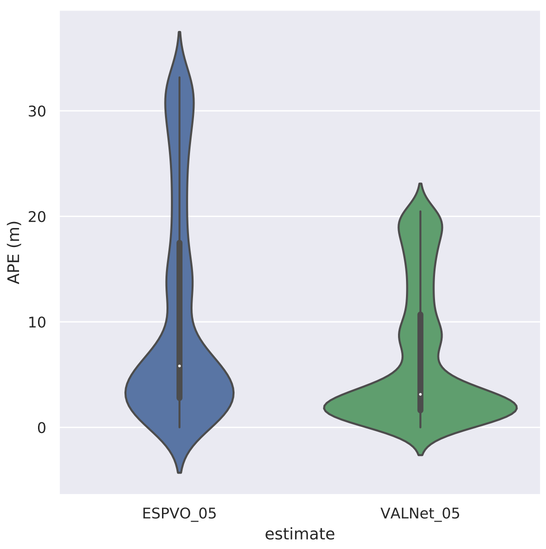


(b) Rotational error

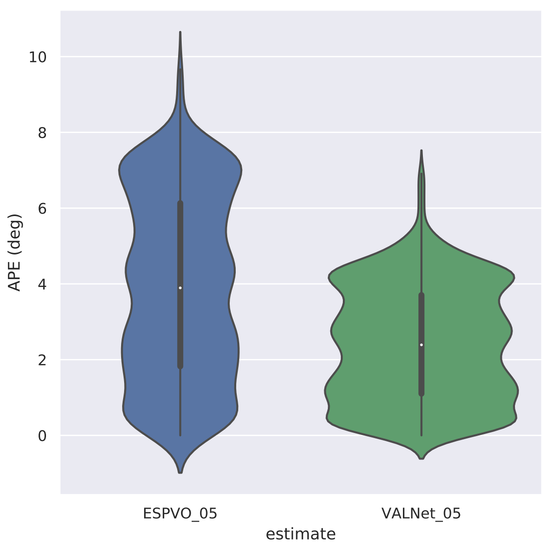
Figure 57: APE violin histogram of the translational[m] and rotational[deg] errors in the Thielallee trajectory.

The results in Thielallee differ from the rest of the KITTI dataset, the main difference is the weather conditions, KITTI dataset was recorded mainly in day-bright light, and since Thielallee is a test sequence with low contrast and different lightening conditions, the error is more significant. We noticed that the neural network still has difficulties in rotational estimations. Moreover, the most significant rotational error was reported while another vehicle drove in the field of view of the MIG camera, which suggests that dynamic objects in the environment are still affecting the estimation.

Fig. 58 test the net in the KITTI sequence No. 5; translational and rotational errors are smaller in magnitude than the compared ESPVO neural network.



(a)



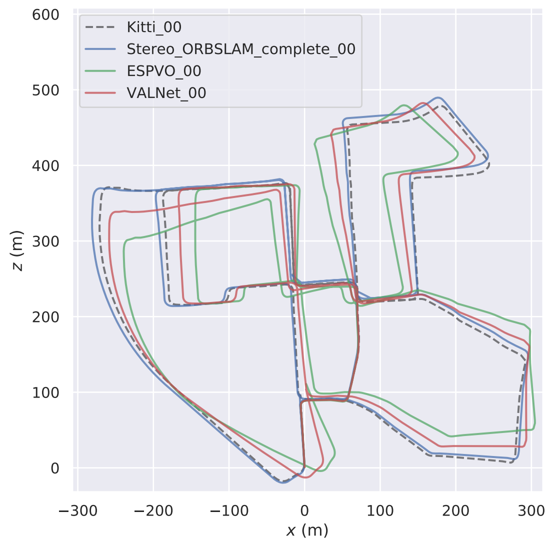
(b)

Figure 58: APE violin histogram of the translational[m] and rotational[deg] errors in the Kitti sequence No. 5.

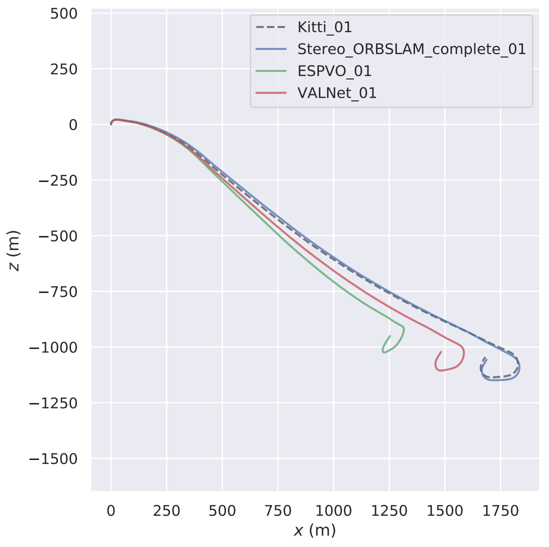
5.3.4 *Qualitative Evaluation of Trajectories*

Fig. 60 and 59 show the trajectories computed with the selected methods. It is observable that VALNet produces relatively accurate and consistent trajectories closer to the ground truth, the scale is estimated with high precision since no additional scale estimation neither post alignment to ground truth is performed to obtain the trajectories. The scale is learned on the end-to-end training. This exposes a major benefit of neural networks against traditional methods like the monocular version of ORB-SLAM which scale drifting contributes to error.

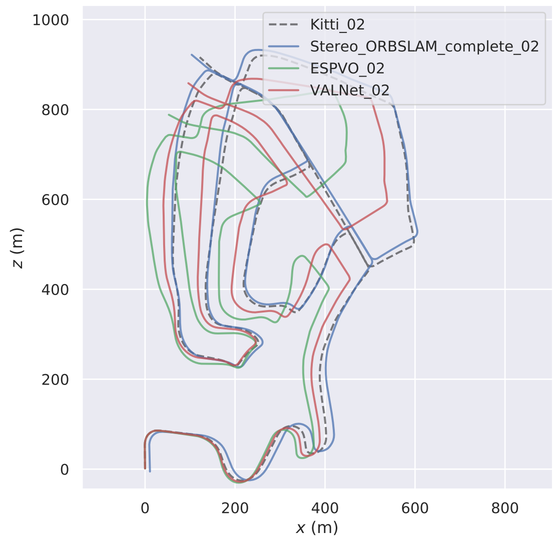
In supervised learning, it is clear that the amount of provided data to the net, improve its accuracy. Therefore we made experiments training the net with sequences from 00 to 10 and running qualitative analysis on the sequences from 11 to 21. Figure 61 shows the results using the three methods.



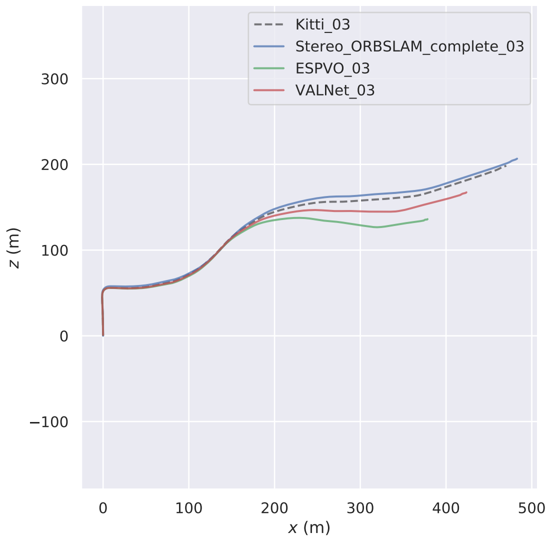
(a) Sequence 00



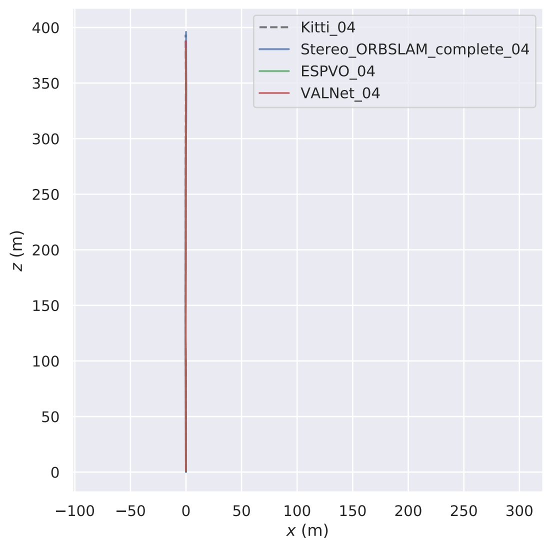
(b) Sequence 01



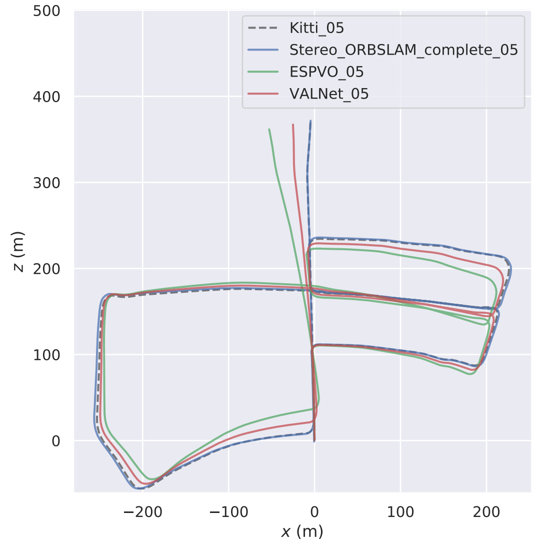
(c) Sequence 02



(d) Sequence 03

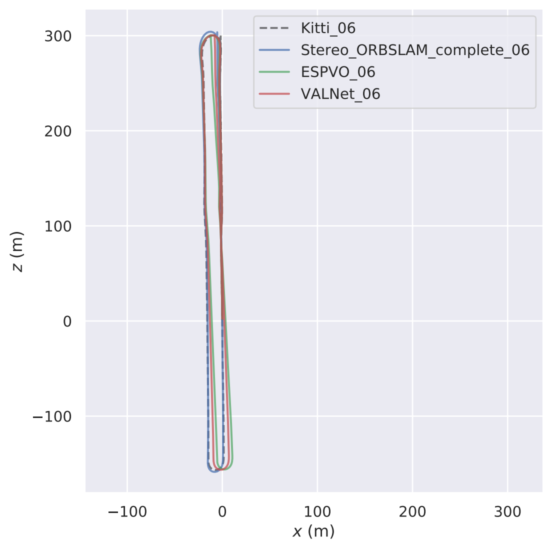


(e) Sequence 04

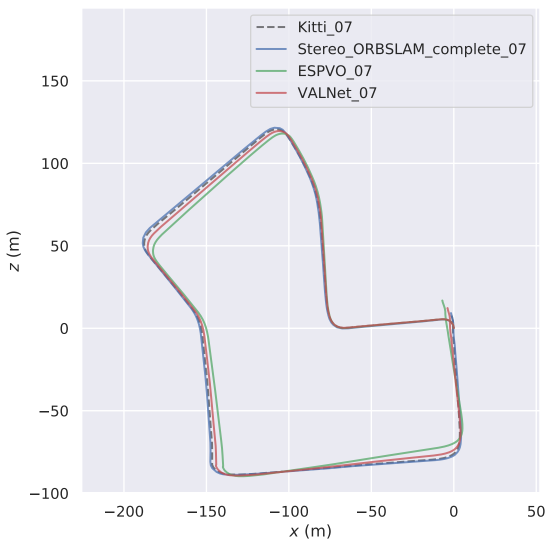


(f) Sequence 05

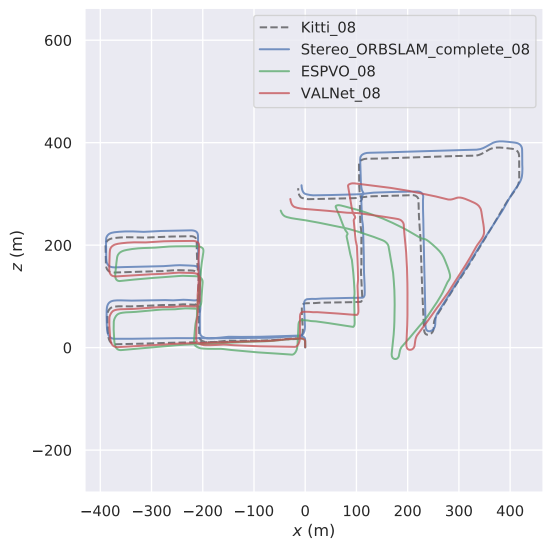
Figure 59: Resulted trajectories of Stereo ORBSLAM2 (complete SLAM), ESP-VO and VALNet on Kitti sequences 00-06. The dashed line shows the ground truth trajectory.



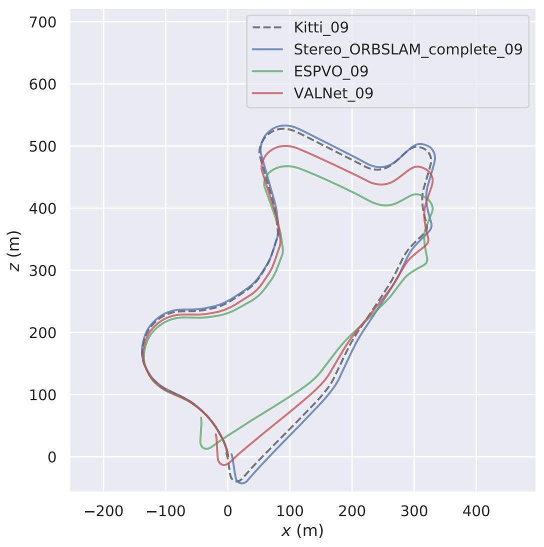
(a) Sequence 06



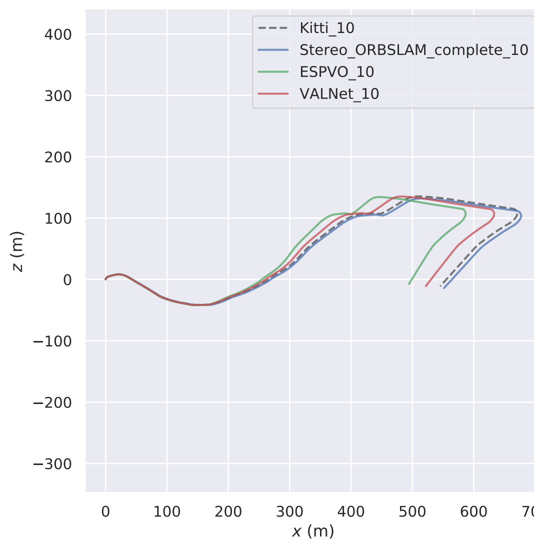
(b) Sequence 07



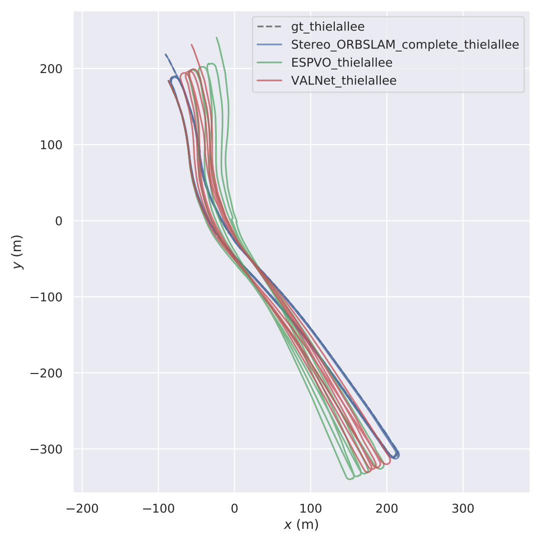
(c) Sequence 08



(d) Sequence 09

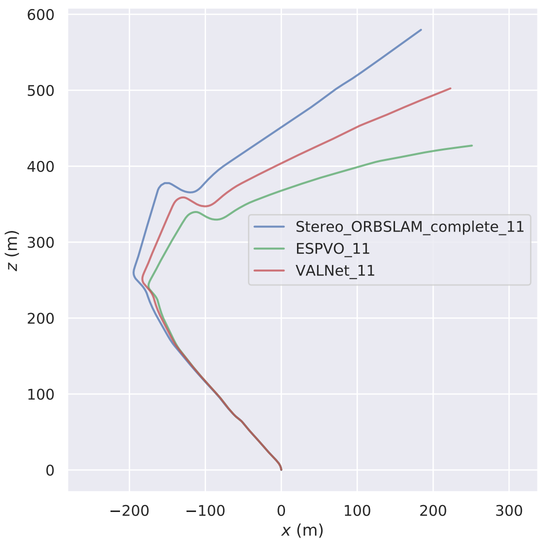


(e) Sequence 10

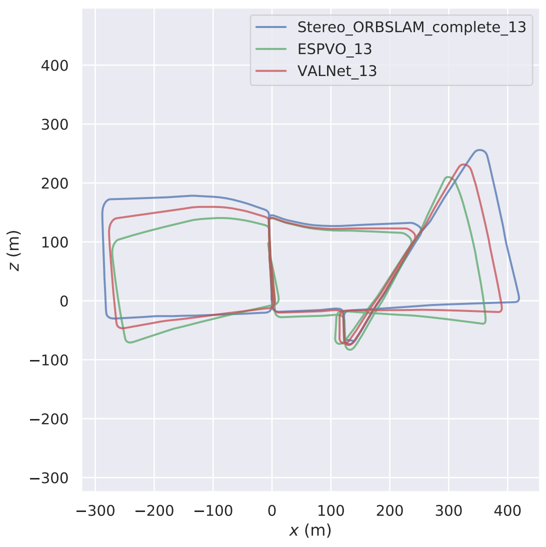


(f) Thielallee

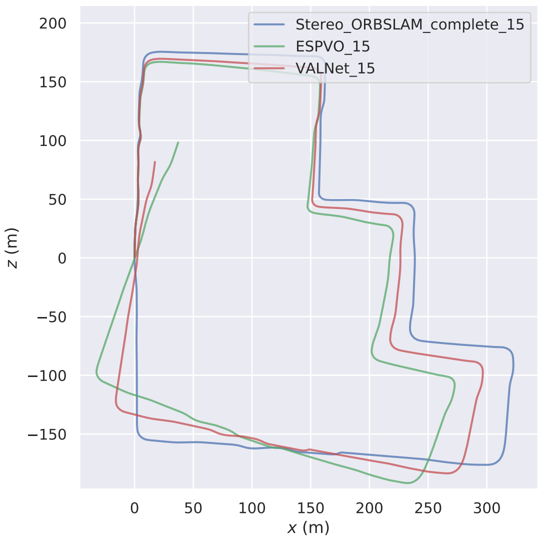
Figure 60: Resulted trajectories of Stereo ORBSLAM2 (complete SLAM), ESP-VO and VALNet on KITTI sequences 06-10 an Thielallee. The dashed line shows the ground truth trajectory.



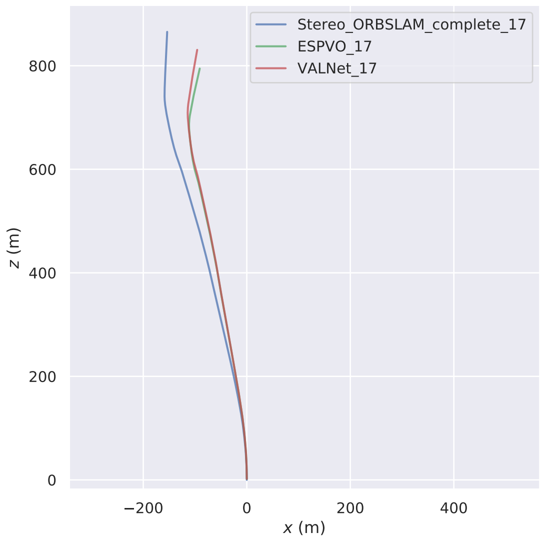
(a) Sequence 11



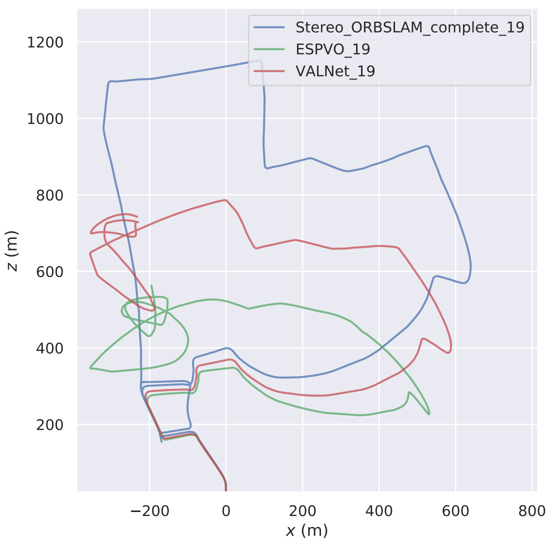
(b) Sequence 13



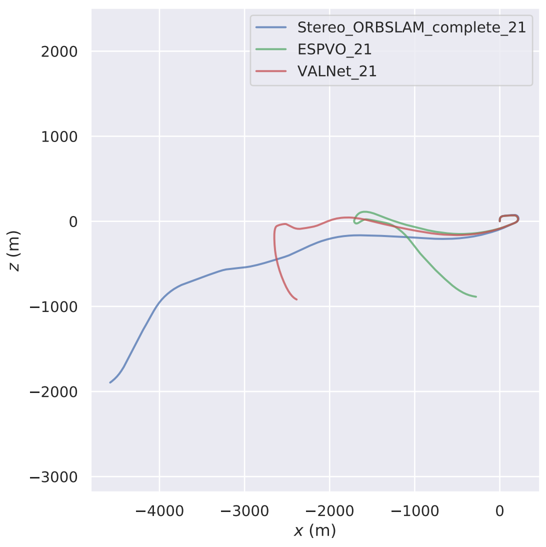
(c) Sequence 15



(d) Sequence 17



(e) Sequence 19



(f) Sequence 21

Figure 61: Resulted trajectories on KITTI testing sequences 11, 13, 15, 17, 19, 21 and Thielallee. Model is trained on KITTI sequences 00-10. On the testing sequences there is no ground truth provided.

The velocity of the car while recording data has a major impact on the accuracy, it is important to remark that the standard deviation and mean of the first training dataset (00, 01, 02, 03, 06, 08, 09) is: $\sigma = 16.10$, $\mu = 36.21$ and for the complete dataset with ground truth is: $\sigma = 15.67$ $\mu = 34.43$. Therefore, in sequences where velocities are high, the estimation could be more challenging, sequence 01, for instance, span velocities up to 98 km/h. At a camera rate of 10Hz, 90 km/h represents a straight displacement of 2.7m, Fig. 62 shows the difference between two frames obtained at 98 km/h in sequence 01.

Fig. 62 also shows that by skipping frames, as explained in Fig. 50 is possible to generate a dataset with different vehicle velocities virtually. Nonetheless, FlowNet2 can better perceive displacement when the vehicle drives at velocities up to 100 km/h, Figs. 62b and 62c, shows that FlowNet2 can effectively distinguish features such as the cars and the right pole. By contrast, in Fig. 62i no features can be distinguished, and it is possible to see some artefacts. Nevertheless, the net can give some information about the environment in a whole, which explains why the odometry still works in cases where traditional methods cannot track any correct feature due to the separation between frames as shown in Fig. 62j and in difficult scene situations like the pole in Fig. 62h which is hard to track âs a consequence of the merging with the building and the low contrast level.

5.3.5 *Reported Runtime*

The network is implemented based on the TensorFlow framework and trained using an NVIDIA Geforce RTX 2080 ti. Adam optimiser is employed to train the network with starting learning rate 0.001 and parameters $\alpha_1 = 0.9$ and $\alpha_2 = 0.999$ both values recommended on the analysis in [42]. The training was set for 200 epochs but using callback Tensorflow implementations such as early stopping if the loss function does not decrease 0.001 for more than five epochs to reduce the training time.

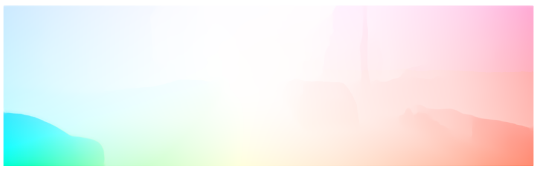
Training time on all the trajectories takes approximately 100k - 200k iterations, or 1 to 3 days. Prediction time per frame pair takes on average 100 ms for FlowNet2 and 75ms for VALNet for a total of 175ms or 5.71Hz.

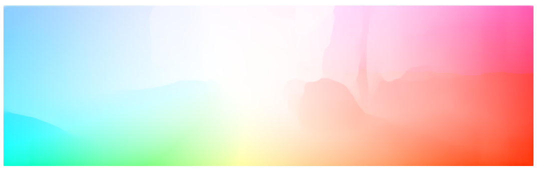
Whereas the system does not run on real-time (>30hz) it shows that is possible to generate an artificial intelligence capable of obtaining similar trajectories as conventional methods in the state of the art datasets such as KITTI. It is very scalable as it does not require an extensive database of landmarks.

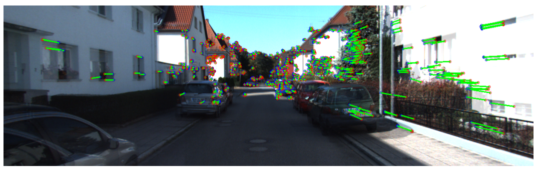
(a) Seq. 00, frame 470

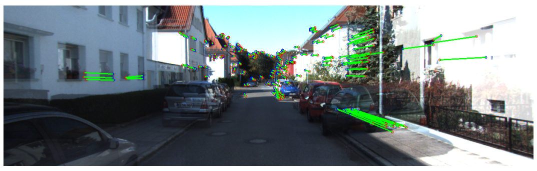
(b) Seq. 00, frame 471

(c) Seq. 00, frame 473

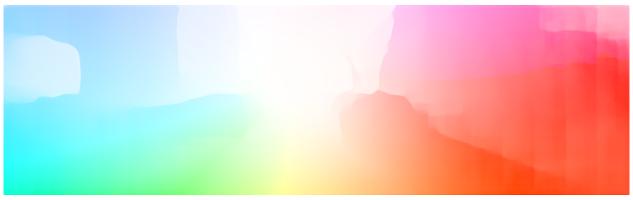

(d) Op. flow 470-471


(e) Op. flow 470-473


(f) ORB Tracking 470-471


(g) ORB Tracking 470-473

(h) Seq. 00, frame 475


(i) Op. flow 470-475

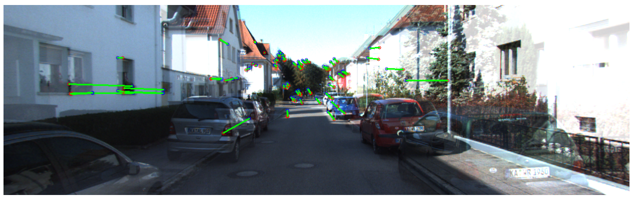

(j) ORB Tracking 470-475

Figure 62: Frames from sequence 01 of the KITTI Dataset with velocity of 31.41 km/h, 94,3 km/h and 156.9 km/h, skipping 0, 2 and 4 frames. In the Flow field color coding, direction is coded by hue and length is coded by saturation.

Chapter 6

Conclusions

In previous chapters, the application of deep neural networks for localisation in autonomous cars was analyzed. From the observed results, this chapter presents some general conclusions and exciting insights for applications and research directions for future work.

6.1 Findings and Limitations

We will concisely summarise the findings and limitations.

Chapter 3 presented AutoMiny, the developed scale autonomous cars from which the version TX1 was a useful tool in the localisation schemes proposed in this thesis. Properly used, the scale platforms are able to accelerate experimentation, data gathering, and algorithm debugging.

Although we intended to approximate as much as possible the AutoMiny working setup to the MIG autonomous car, scaling sensors such as LIDAR and RADAR or outdoor localisation systems are still pending. Having such sensors would provide rich information about the dynamic outdoor environment.

Chapter 4 presented GALNet, a deep learning architecture for pose estimation employing inertial, kinematic, and wheel velocity data from the car. We employed VSA rate as the main characteristic to estimate vehicle displacements.

We showed that it is possible to use the experiments performed by different vehicles to improve the results of the deep neural network model. The results of the estimation were compared with a Classical Unscented Kalman Filter predictor and a basic wheel odometry scheme. Transferring learning between two different experimental platforms brings advantages to the accuracy of the net. However, the model is highly dependent on the sensor, which provides ground truth, and the intrinsic drifting of the device is also transferred to the model.

Chapter 5 introduces VALNet, a Siamese convolutional-recurrent neural network for local camera ego-motion regression. We show that feeding the dataset in the training process in small closed-loop sequences, increment the founded associations by the neural net. The Siamese architecture has the significant advantage of being able to equalize the training error between the network individual sides while looking for specific characteristics. In this work, we utilize it to find associations between optical flow pairs feeding the forward and backward direction of the sequences at the same time.

Finally, we demonstrated that this method is able to build trajectories from images in an end-to-end logic. Nonetheless, improving the accuracy of these methods is still a work on progress.

Although the performance was not ideal for replacing other expensive sensors such as the Applanix due to accuracy and security concerns, we believe that this method will eventually overtake the actual current setup of autonomous vehicles.

Along with the applications of deep end-to-end learning in computer vision for parameter estimation like depth or optical flow, we showed that visual localisation has a big potential to outperform hand-engineered approaches. It reduces engineering effort and performs very well by optimising the model with respect to the end goal.

The proposed method does not require feature engineering or object tracking to estimate ego-motion, which suggests that neural networks can build the necessary associations to calculate displacement.

6.2 Directions for Future Work

To conclude this thesis, we have devised some aspects which will guide future work, and we describe them in this section. As is natural in any research, this thesis submits many questions for future research.

We discussed individual improvements to the algorithms presented along with the work of this thesis. Nonetheless, we would like to highlight the following high-level themes for future research, which are particularly promising.

6.2.1 *Continual Lifelong Learning*

Nonetheless, we showed that it is possible to increase the accuracy of the models by complementing the dataset with information from other platforms. We also demonstrated that this information should be carefully selected. Otherwise, the net performance will decrease in areas where it was attempting to improve. The ability to continually learn over time by accommodating new knowledge while retaining previously learned experiences is referred to as continual or lifelong learning. The principle of continuous lifelong learning has several considerations which, when not complied with, diminishing the learning or the inference performance [61]. Therefore, the implementation of a model capable of integrating additional information just in the identified sections of improvement, while driving, is crucial.

6.2.2 *Model Fusion Through Learned Uncertainties*

The standard methodology in machine learning is to learn one task at a time. Significant problems are broken into small, reasonably independent sub-problems that are learned separately and then recombined.

Multitask learning aims to improve learning efficiency and prediction accuracy by learning multiple objectives from a shared representation. Because the outputs share a standard hidden layer, it is possible for internal representations that arise in the hidden layer for one task to be used by other tasks. Sharing what is learned by different tasks while tasks are trained in parallel is the central idea in multitask learning.

The manually tuned parameters in the localisation loss functions can be transformed on statistical variables, which vary according to the uncertainties of estimation. In the case of equations 4.7 and 4.5 the factor κ would change during training according with the translational and rotational uncertainty estimation.

Moreover, fusing inertial-dynamic and camera data through Multitask Bayesian modeling would take advantage of aleatoric and epistemic uncertainty, which capture noise inherent in the observations and the model. This could strengthen the model against sensor noise or motion noise, give information about the model parameters, uncertainty, and aid the decision-making as well as positively the security standards of autonomous cars.

6.2.3 *Instance Segmentation with 6D Dense Optical Flow*

We have confirmed that deep neural networks can learn scale from monocular images in order to estimate ego-motion. Even though the optical flow estimated by FlowNet2 is a 3D motion vector per pixel.

It is already known that high dynamical objects in classical filtering methods are the most challenging due to the data association. Therefore, subtracting the static elements or background of the scene is an extensive research area of computer vision. Most of the methods use previously-trained neural nets based on instance segmentation [8].

That said, it is in our interest to explore the benefit of developing a deep neural net capable of estimating a dense optical flow of 6D motion vector that, fused with a pre-trained instance model such as YOLO, could predict a more advantageous localisation system.

References

[1] Manuel Acosta Reche and Stratis Kanarachos. "Tire lateral force estimation and grip potential identification using Neural Networks, Extended Kalman Filter, and Recursive Least Squares." In: *Neural Computing and Applications* 2017 (Feb. 2017), pp. 1–21. DOI: 10.1007/s00521-017-2932-9.

[2] Angel Alatorre Vazquez, Ali Charara, and Alessandro Victorino. "Sideslip estimation algorithm comparison between Euler angles and quaternion approaches with black box vehicle model." In: Mar. 2018, pp. 553–559. DOI: 10.1109/AMC.2019.8371153.

[3] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. *Concrete Problems in AI Safety*. 2016. arXiv: 1606.06565 [cs.AI].

[4] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation." In: *CoRR* abs/1511.00561 (2015). arXiv: 1511.00561. URL: http://arxiv.org/abs/1511.00561.

[5] Jakob Bechtoff, Lars Koenig, and Rolf Isermann. "Cornering Stiffness and Sideslip Angle Estimation for Integrated Vehicle Dynamics Control." In: *IFAC-PapersOnLine* 49.11 (2016). 8th IFAC Symposium on Advances in Automotive Control AAC 2016, pp. 297 –304. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2016.08.045. URL: http://www.sciencedirect.com/science/article/pii/S2405896316313672.

[6] Philippe Bonnifait, P. Bouron, Dominique Meizel, and Paul Crubille. "Dynamic Localization of Car-like Vehicles using Data Fusion of Redundant ABS Sensors." In: *The Journal of Navigation* 56 (Sept. 2003), pp. 429 –441. DOI: 10.1017/S037346330300242X.

[7] Philippe Bonnifait, Pascal Bouron, Paul Crubille, and Dominique Meizel. "Data fusion of four ABS sensors and GPS for an enhanced localization of car-like vehicles." In: vol. 2. Feb. 2001, 1597 –1602 vol.2. ISBN: 0-7803-6576-3. DOI: 10.1109/ROBOT.2001.932839.

[8] Thierry Bouwmans, Sajid Javed, Maryam Sultana, and Soon Ki Jung. "Deep Neural Network Concepts for Background Subtraction: A Systematic Review and Comparative Evaluation." In: *Neural networks : the official journal of the International Neural Network Society* 117 (2019), pp. 8–66.

[9] David Broderick, David Bevly, and John Hung. "An adaptive non-linear state estimator for vehicle lateral dynamics." In: Nov. 2009, pp. 1450–1455. ISBN: 978-1-4244-4648-3. DOI: 10.1109/IECON.2009.5414721.

[10] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. "High accuracy optical flow estimation based on a theory for warping." In: *European Conference on Computer Vision (ECCV)*. Vol. 3024. Lecture Notes in Computer Science. Springer, 2004, pp. 25–36. URL: http://lmb.informatik.uni-freiburg.de/Publications/2004/Bro04a.

[11] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age." In: *IEEE Transactions on Robotics* 32.6 (2016), pp. 1309–1332. ISSN: 1941-0468. DOI: 10.1109/TRO.2016.2624754.

[12] Rich Caruana. "Multitask Learning." In: *Mach. Learn.* 28.1 (July 1997), 41–75. ISSN: 0885-6125. DOI: 10.1023/A:1007379606734. URL: https://doi.org/10.1023/A:1007379606734.

[13] Aleksandar Chakarov, Aditya Nori, Sriram Rajamani, Shayak Sen, and Deepak Vijaykeerthy. *Debugging Machine Learning Tasks*. 2016. arXiv: 1603.07292 [cs.LG].

[14] Bai-fan Chen, Dian Yuan, Chunfa Liu, and Qian Wu. "Loop Closure Detection Based on Multi-Scale Deep Feature Fusion." In: *Applied Sciences* 9 (Mar. 2019), p. 1120. DOI: 10.3390/app9061120.

[15] Wenzheng Chen, Jun Gao, Huan Ling, Edward Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. "Learning to Predict 3D Objects with an Interpolation-based Differentiable Renderer." In: *ArXiv* abs/1908.01210 (2019).

[16] Daniel Chindamo, Basilio Lenzo, and Marco Gadola. "On the Vehicle Sideslip Angle Estimation: A Literature Review of Methods, Models, and Innovations." In: *Applied Sciences* 8.3 (2018). ISSN: 2076-3417. DOI: 10.3390/app8030355. URL: https://www.mdpi.com/2076-3417/8/3/355.

[17] Gabriele Costante, Michele Mancini, Paolo Valigi, and Thomas Ciarfuglia. "Exploring Representation Learning With CNNs for Frame-to-Frame Ego-Motion Estimation." In: *IEEE Robotics and Automation Letters* 1 (Dec. 2015), pp. 1–1. DOI: 10.1109/LRA.2015.2505717.

[18] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. "MonoSLAM: Real-Time Single Camera SLAM." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.6 (2007), pp. 1052–1067. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2007.1049.

[19] John Dye and Hamid Lankarani. “Hybrid Simulation of a Dynamic Multibody Vehicle Suspension System Using Neural Network Modeling Fit of Tire Data.” In: Aug. 2016, V006T09A036. DOI: 10.1115/DETC2016-60435.

[20] J. Engel, V. Koltun, and D. Cremers. “Direct Sparse Odometry.” In: (Mar. 2018).

[21] J. Engel, T. Schöps, and D. Cremers. “LSD-SLAM: Large-Scale Direct Monocular SLAM.” In: *eccv*. 2014.

[22] Jakob Engel, Thomas Schoeps, and Daniel Cremers. “LSD-SLAM: large-scale direct monocular SLAM.” In: vol. 8690. Sept. 2014, pp. 1–16. DOI: 10.1007/978-3-319-10605-2_54.

[23] Peter Fejes. *Estimation of Steering Wheel Angle in Heavy-Duty Trucks*. 2016.

[24] Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. “FlowNet: Learning Optical Flow with Convolutional Networks.” In: *CoRR* abs/1504.06852 (2015). arXiv: 1504.06852. URL: http://arxiv.org/abs/1504.06852.

[25] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. “On-Manifold Preintegration Theory for Fast and Accurate Visual-Inertial Navigation.” In: *CoRR* abs/1512.02363 (2015). arXiv: 1512.02363. URL: http://arxiv.org/abs/1512.02363.

[26] Jannik Fritsch, Tobias Kuehnl, and Andreas Geiger. “A New Performance Measure and Evaluation Benchmark for Road Detection Algorithms.” In: *International Conference on Intelligent Transportation Systems (ITSC)*. 2013.

[27] Yoshiki Fukada. “Slip-Angle Estimation for Vehicle Stability Control.” In: *Vehicle System Dynamics* 32.4-5 (1999), pp. 375–388. DOI: 10.1076/vesd.32.4.375.2079. eprint: https://www.tandfonline.com/doi/pdf/10.1076/vesd.32.4.375.2079. URL: https://www.tandfonline.com/doi/abs/10.1076/vesd.32.4.375.2079.

[28] Andreas Geiger, Philip Lenz, and Raquel Urtasun. “Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite.” In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012.

[29] “Google reports self-driving car mistakes: 272 failures and 13 near misses.” In: *The Guardian* (2016). URL: https://www.theguardian.com/technology/2016/jan/12/google-self-driving-cars-mistakes-data-reports.

[30] Michael Grupp. *evo: Python package for the evaluation of odometry and SLAM*. https://github.com/MichaelGrupp/evo. 2017.

[31] Xiuyan Guo. "Feature-Based Localization Methods for Autonomous Vehicles." Tag der Disputation: 11.04.2017. PhD thesis. 14195 Berlin: Freien Universität Berlin, 2017.

[32] Sepp Hochreiter. "The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions." In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6 (Apr. 1998), pp. 107–116. DOI: 10.1142/S0218488598000094.

[33] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-term Memory." In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[34] "How Uber's Self-Driving Technology Could Have Failed In The Fatal Tempe Crash." In: *Forbes* (2018).

[35] E. Ilg, T. Saikia, M. Keuper, and T. Brox. "Occlusions, Motion and Depth Boundaries with a Generic Network for Disparity, Optical Flow or Scene Flow Estimation." In: *European Conference on Computer Vision (ECCV)*. 2018. URL: http://lmb.informatik.uni-freiburg.de/Publications/2018/ISKB18.

[36] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. "FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks." In: *CoRR* abs/1612.01925 (2016). arXiv: 1612.01925. URL: http://arxiv.org/abs/1612.01925.

[37] E. Jelavic, J. Gonzales, and F. Borrelli. "Autonomous Drift Parking using a Switched Control Strategy with Onboard Sensors." In: *IFAC-PapersOnLine* 50.1 (2017). 20th IFAC World Congress, pp. 3714 –3719. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2017.08.568. URL: http://www.sciencedirect.com/science/article/pii/S2405896317309394.

[38] Chi Jin, Liang Shao, Cornelia Lex, and Arno Eichberger. "Vehicle Side Slip Angle Observation with Road Friction Adaptation." In: *IFAC-PapersOnLine* 50.1 (2017). 20th IFAC World Congress, pp. 3406 –3411. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2017.08.593. URL: http://www.sciencedirect.com/science/article/pii/S2405896317309667.

[39] Fabjan Kallasi, Dario Lodi Rizzini, Fabio Oleari, Massimiliano Magnani, and Stefano Caselli. "A novel calibration method for industrial AGVs." In: *Robotics and Autonomous Systems* 94 (2017), pp. 75 –88. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2017.04.019. URL: http://www.sciencedirect.com/science/article/pii/S0921889016303025.

[40] Alex Kendall, Matthew Grimes, and Roberto Cipolla. "Convolutional networks for real-time 6-DOF camera relocalization." In: *CoRR* abs/1505.07427 (2015). arXiv: 1505.07427. URL: http://arxiv.org/abs/1505.07427.

[41] U Kiencke and L Nielsen. "Automotive Control Systems: For Engine, Driveline, and Vehicle." In: *Measurement Science and Technology* 11.12 (2000), pp. 1828–1828. DOI: 10.1088/0957-0233/11/12/708. URL: https://doi.org/10.1088%2F0957-0233%2F11%2F12%2F708.

[42] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. arXiv: 1412.6980 [cs.LG].

[43] Georg Klein and David Murray. "Parallel Tracking and Mapping for Small AR Workspaces." In: *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*. Nara, Japan, 2007.

[44] Philip Koopman. "Challenges in Autonomous Vehicle Validation: Keynote Presentation Abstract." In: Apr. 2017, pp. 3–3. ISBN: 978-1-4503-4976-5. DOI: 10.1145/3055378.3055379.

[45] Saiprasad Koturwar and Shabbir Merchant. "Weight Initialization of Deep Neural Networks(DNNs) using Data Statistics." In: *CoRR* abs/1710.10570 (2017). arXiv: 1710.10570. URL: http://arxiv.org/abs/1710.10570.

[46] F. R. Kschischang, B. J. Frey, and H. . Loeliger. "Factor graphs and the sum-product algorithm." In: *IEEE Transactions on Information Theory* 47.2 (2001), pp. 498–519. ISSN: 1557-9654. DOI: 10.1109/18.910572.

[47] Zeshan Kurd, Tim Kelly, and Jim Austin. "Developing artificial neural networks for safety critical systems." In: *Neural Computing and Applications* 16 (Jan. 2007), pp. 11–19. DOI: 10.1007/s00521-006-0039-9.

[48] Ruihao Li, Sen Wang, Zhiqiang Long, and Dongbing Gu. *UnDeepVO: Monocular Visual Odometry through Unsupervised Deep Learning*. 2017. arXiv: 1709.06841 [cs.CV].

[49] Zhencai Li, Yang Wang, and Zhen Liu. "Unscented Kalman Filter-Trained Neural Networks for Slip Model Prediction." In: *PloS one* 11 (July 2016), e0158492. DOI: 10.1371/journal.pone.0158492.

[50] F. Lin and C. Huang. "State-of-the-art of vehicle side slip angle estimation." In: *Zhongguo Jixie Gongcheng/China Mechanical Engineering* 24 (Jan. 2013), pp. 135–141. DOI: 10.3969/j.issn.1004-132X.2013.01.026.

[51] Ruben Martinez-Cantin and Jose Castellanos. “Unscented SLAM for large-scale outdoor environments.” In: Sept. 2005, pp. 3427 –3432. ISBN: 0-7803-8912-3. DOI: 10.1109/IROS.2005.1545002.

[52] S. Melzi and E. Sabbioni. “On the vehicle sideslip angle estimation through neural networks: Numerical and experimental results.” In: *Mechanical Systems and Signal Processing* 25.6 (2011). Interdisciplinary Aspects of Vehicle Dynamics, pp. 2005 –2019. ISSN: 0888-3270. DOI: https://doi.org/10.1016/j.ymssp.2010.10.015. URL: http://www.sciencedirect.com/science/article/pii/S0888327010003341.

[53] Stefano Melzi, Edoardo Sabbioni, Alessandro Concas, and Marco Pesce. “Vehicle Sideslip Angle Estimation Through Neural Networks: Application to Experimental Data.” In: Jan. 2006. DOI: 10.1115/ESDA2006-95451.

[54] R.J.E. Merry, M.J.G. van de Molengraft, and M. Steinbuch. “Velocity and Acceleration Estimation for Optical Incremental Encoders.” In: *IFAC Proceedings Volumes* 41.2 (2008). 17th IFAC World Congress, pp. 7570 –7575. ISSN: 1474-6670. DOI: https://doi.org/10.3182/20080706-5-KR-1001.01280. URL: http://www.sciencedirect.com/science/article/pii/S1474667016401631.

[55] T. Moore and D. Stouch. “A Generalized Extended Kalman Filter Implementation for the Robot Operating System.” In: *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*. Springer, 2014.

[56] Raul Mur-Artal and Juan D. Tardós. “ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras.” In: *CoRR* abs/1610.06475 (2016). arXiv: 1610.06475. URL: http://arxiv.org/abs/1610.06475.

[57] 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*. 2011, pp. 2320–2327. DOI: 10.1109/ICCV.2011.6126513.

[58] Besmira Nushi, Ece Kamar, Eric Horvitz, and Donald Kossmann. *On Human Intellect and Machine Failures: Troubleshooting Integrative Machine Learning Systems*. 2016. arXiv: 1611.08309 [cs.LG].

[59] “Copyright.” In: *Tire and Vehicle Dynamics (Third Edition)*. Ed. by Hans B. Pacejka. Third Edition. Oxford: Butterworth-Heinemann, 2012, p. iv. ISBN: 978-0-08-097016-5. DOI: https://doi.org/10.1016/B978-0-08-097016-5.02001-5. URL: http://www.sciencedirect.com/science/article/pii/B9780080970165020015.

[60] Raja Parasuraman and Victor Riley. “Humans and Automation: Use, Misuse, Disuse, Abuse.” In: *Human Factors* 39.2 (1997), pp. 230–253. DOI: 10.1518/001872097778543886. eprint: https://doi.org/10.1518/001872097778543886. URL: https://doi.org/10.1518/001872097778543886.

[61] German Ignacio Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. “Continual Lifelong Learning with Neural Networks: A Review.” In: 2018.

[62] Emilio Parisotto, Devendra Singh Chaplot, Jian Zhang, and Ruslan Salakhutdinov. “Global Pose Estimation with an Attention-based Recurrent Network.” In: *CoRR* abs/1802.06857 (2018). arXiv: 1802.06857. URL: http://arxiv.org/abs/1802.06857.

[63] Joseph Redmon and Ali Farhadi. “YOLOv3: An Incremental Improvement.” In: *arXiv* (2018).

[64] Haoyu Ren, Mostafa El-khamy, and Jungwon Lee. *Deep Robust Single Image Depth Estimation Neural Network Using Scene Understanding*. 2019. arXiv: 1906.03279 [cs.CV].

[65] R. Roberts, C. Potthast, and Frank Dellaert. “Learning general optical flow subspaces for egomotion estimation and detection of motion anomalies.” In: *2012 IEEE Conference on Computer Vision and Pattern Recognition* 0 (June 2009), pp. 57–64. DOI: 10.1109/CVPR.2009.5206538.

[66] Richard Roberts, Hai Nguyen, Niyant Krishnamurthi, and Tucker Balch. “Memory-based learning for visual odometry.” In: June 2008, pp. 47 –52. ISBN: 978-1-4244-1646-2. DOI: 10.1109/ROBOT.2008.4543185.

[67] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation.” In: *CoRR* abs/1505.04597 (2015). arXiv: 1505.04597. URL: http://arxiv.org/abs/1505.04597.

[68] Krzysztof Rusek, José Suárez-Varela, Albert Mestres, Pere Barlet-Ros, and Albert Cabellos-Aparicio. “Unveiling the potential of Graph Neural Networks for network modeling and optimization in SDN.” In: *CoRR* abs/1901.08113 (2019). arXiv: 1901.08113. URL: http://arxiv.org/abs/1901.08113.

[69] Rick Salay, Rodrigo Queiroz, and Krzysztof Czarnecki. *An Analysis of ISO 26262: Using Machine Learning Safely in Automotive Software*. 2017. arXiv: 1709.02435 [cs.AI].

[70] Hideaki Sasaki and Takatoshi Nishimaki. “A Side-Slip Angle Estimation Using Neural Network for a Wheeled Vehicle.” In: *SAE Transactions* 109 (2000), pp. 1026–1031. ISSN: 0096736X, 25771531. URL: http://www.jstor.org/stable/44686942.

[71] A. Scorer. “Vehicle Location and Navigation Systems. By Yilin Zhao. Artech House, 1997. 345 pages, £65 Hardback. ISBN: 0-89006-861-5.” In: *Journal of Navigation* 51.3 (1998), 445–447. DOI: 10.1017/S0373463398238045.

[72] Bernd Spanfelner, Detlev Richter, Susanne Ebel, Ulf Wilhelm, and C. Patz. “Challenges in applying the ISO 26262 for driver assistance systems.” In: 2012.

[73] Robert Spangenberg. “Landmark-based Localization for Autonomous Vehicles.” Tag der Disputation: 23. März 2016. PhD thesis. 14195 Berlin: Freien Universität Berlin, 2015.

[74] Stanford Artificial Intelligence Laboratory et al. *Robotic Operating System*. Version ROS Melodic Morenia. May 23, 2018. URL: https://www.ros.org.

[75] Jrgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. “A benchmark for the evaluation of RGB-D SLAM systems.” In: Oct. 2012, pp. 573–580. ISBN: 978-1-4673-1737-5. DOI: 10.1109/IROS.2012.6385773.

[76] Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. “Visual SLAM algorithms: A survey from 2010 to 2016.” English. In: *IPSJ Transactions on Computer Vision and Applications* 9 (Jan. 2017). ISSN: 1882-6695. DOI: 10.1186/s41074-017-0027-2.

[77] Sebastian Thrun and Yufeng Liu. “Multi-robot SLAM with Sparse Extended Information Filers.” In: vol. 15. Jan. 2003, pp. 254–266. DOI: 10.1007/11008941_27.

[78] Sebastian Thrun, Michael Montemerlo, Daphne Koller, Ben Wegbreit, Juan Nieto, and Eduardo Nebot. “FastSLAM: An Efficient Solution to the Simultaneous Localization And Mapping Problem with Unknown Data.” In: *Journal of Machine Learning Research* 4 (May 2004).

[79] Shinji Umeyama. “Least-Squares Estimation of Transformation Parameters Between Two Point Patterns.” In: *IEEE Trans. Pattern Anal. Mach. Intell.* 13 (1991), pp. 376–380.

[80] Michelle Valente, Cyril Joly, and Arnaud de La Fortelle. *Deep Sensor Fusion for Real-Time Odometry Estimation*. 2019. arXiv: 1908.00524 [cs.RO].

[81] J. Valentin, M. Nießner, J. Shotton, A. Fitzgibbon, S. Izadi, and P. Torr. “Exploiting uncertainty in regression forests for accurate camera relocalization.” In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 4400–4408. DOI: 10.1109/CVPR.2015.7299069.

[82] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. *Attention Is All You Need*. 2017. arXiv: 1706.03762 [cs.CL].

[83] Efstathios Velenis, Emilio Frazzoli, and Panagiotis Tsiotras. "Steady-state cornering equilibria and stabilisation for a vehicle during extreme operating conditions." In: *Int. J. of Vehicle Autonomous Systems* 8 (Oct. 2010). DOI: 10.1504/IJVAS.2010.035797.

[84] R. Wang, M. Schwörer, and D. Cremers. "Stereo DSO: Large-Scale Direct Sparse Visual Odometry with Stereo Cameras." In: *International Conference on Computer Vision (ICCV)*. Venice, Italy, 2017.

[85] Sen Wang, Ronald Clark, Hongkai Wen, and Niki Trigoni. "DeepVO: Towards end-to-end visual odometry with deep Recurrent Convolutional Neural Networks." In: *2017 IEEE International Conference on Robotics and Automation (ICRA)* (2017). DOI: 10.1109/icra.2017.7989236. URL: http://dx.doi.org/10.1109/ICRA.2017.7989236.

[86] Sen Wang, Ronald Clark, Hongkai Wen, and Niki Trigoni. "End-to-End, Sequence-to-Sequence Probabilistic Visual Odometry through Deep Neural Networks." In: (Apr. 2018).

[87] Wang Wei, Bei Shaoyi, Zhang Lanchun, Zhu Kai, Wang Yongzhi, and Hang Weixing. "Vehicle Sideslip Angle Estimation Based on General Regression Neural Network." In: *Mathematical Problems in Engineering* 2016 (Aug. 2016), pp. 1–7. DOI: 10.1155/2016/3107910.

[88] Alejandro Weinstein and Kevin Moore. "Pose estimation of Ackerman steering vehicles for outdoors autonomous navigation." In: Apr. 2010, pp. 579 –584. DOI: 10.1109/ICIT.2010.5472738.

[89] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou. "Aggressive driving with model predictive path integral control." In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 1433–1440. DOI: 10.1109/ICRA.2016.7487277.

[90] Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James Rehg, Byron Boots, and Evangelos Theodorou. "Information theoretic MPC for model-based reinforcement learning." In: May 2017, pp. 1714–1721. DOI: 10.1109/ICRA.2017.7989202.

[91] Lu Xiong, Xin Xia, Yishi Lu, Wei Liu, Shunhui Song, Yanqun Han, and Zhuoping Yu. "IMU-Based Automated Vehicle Slip Angle and Attitude Estimation Aided by Vehicle Dynamics." In: *Sensors* 19 (Apr. 2019), p. 1930. DOI: 10.3390/s19081930.

[92] S. Yim. "Coordinated control of ESC and AFS with adaptive algorithms." In: *International Journal of Automotive Technology* 18.2 (2017), pp. 271–277. ISSN: 1976-3832. DOI: 10.1007/s12239-017-0027-3. URL: https://doi.org/10.1007/s12239-017-0027-3.

[93] Mario Zanon, Janick Frasch, Milan Vukov, Sebastian Sager, and Moritz Diehl. "Model Predictive Control of Autonomous Vehicles." In: vol. 455. Mar. 2014. DOI: 10.1007/978-3-319-05371-4_3.

[94] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. "Pyramid Scene Parsing Network." In: *CoRR* abs/1612.01105 (2016). arXiv: 1612.01105. URL: http://arxiv.org/abs/1612.01105.

[95] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. "Image quality assessment: from error visibility to structural similarity." In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612. ISSN: 1941-0042. DOI: 10.1109/TIP.2003.819861.

Selbstständigkeitserklärung

Hiermit versichere ich, Ricardo Carrillo Mendoza, dass ich alle Hilfsmittel und Hilfe angegeben habe und auf dieser Grundlage die Arbeit mit dem Titel "Deep Learning-based Localisation for Autonomous Vehicles" selbstständig verfasst habe. Ich erkläre weiterhin, dass die Arbeit nicht schon einmal in einem früheren Promotionsverfahren eingereicht wurde.

Berlin, January 14, 2021

Ricardo Carrillo Mendoza