Dissertation

Deep Learning Methods for Vehicle Localization and Control under Challenging Conditions

Patrick Wenzel

Chair of
Computer Vision and
Artificial Intelligence

Technical
University
of Munich

TECHNISCHE UNIVERSITÄT MÜNCHEN
TUM School of Computation, Information and Technology

DEEP LEARNING METHODS FOR VEHICLE LOCALIZATION AND CONTROL UNDER CHALLENGING CONDITIONS

PATRICK M. WENZEL

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitz: Prof. Dr. Nassir Navab

Prüfer der Dissertation:

1. Prof. Dr. Daniel Cremers
2. Prof. Dr. Wolfram Burgard

Die Dissertation wurde am 21.12.2022 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 19.06.2023 angenommen.

To Clara.

ABSTRACT

Teaching robots to perceive, understand, and interact with the world around us is one of the fundamental problems for building artificial intelligence systems. Deep learning recently made great progress in solving long-standing research challenges for computer vision problems. However, current artificial intelligence algorithms are still far from achieving human-level performance in real-world applications, even after being trained with enormous amounts of labeled data. Another major challenge is the development of computer vision systems that are robust and can reliably provide detailed and accurate information about their surroundings even under changing environmental conditions. To this end, this thesis presents deep learning-based methods for vehicle localization and control under challenging conditions.

We introduce effective methods for robust vision-based sensorimotor control that do not require annotated data for each changing environmental condition. Our proposed learning-based frameworks combine transfer learning and domain adaptation strategies to transfer knowledge between multiple different conditions. The experiments reveal that our approach achieves equivalent performance with a small subset as it can with the total amount of annotated data.

Furthermore, we present a direct visual localization system that exploits featuremetric instead of photometric error for the optimization and thus improves robustness against visual appearance changes. Our method can be trained in a self-supervised manner, removing the need for manual human annotations. The experimental evaluation shows that our approach outperforms classical methods and is more robust against bad initialization, illumination, and weather changes.

Finally, we present a large-scale benchmark dataset composed of challenging environmental conditions to evaluate visual SLAM and long-term localization algorithms. Our dataset provides accurate reference poses and maps from multiple runs in the same scenes throughout the year. It enables us to understand the shortcomings of current algorithms and helps to advance the state-of-the-art in long-term localization and mapping.

ZUSAMMENFASSUNG

Robotern beizubringen, die Welt um uns herum wahrzunehmen, zu verstehen und mit ihr zu interagieren, stellt eines der grundlegenden Probleme bei der Entwicklung künstlicher Intelligenzsysteme dar. In den letzten Jahren wurden im Bereich des Deep Learnings große Fortschritte bei der Lösung langjähriger Forschungsaufgaben für Computer-Vision-Probleme gemacht. Die derzeitigen Algorithmen für künstliche Intelligenz sind allerdings, selbst wenn sie mit enormen Mengen an annotierten Daten trainiert wurden, noch weit davon entfernt, in praktischen Anwendungen eine Leistung auf menschlichem Niveau zu erreichen. Eine weitere große Herausforderung ist die Entwicklung von Bildverarbeitungssystemen, die robust sind und auch unter wechselnden Umweltbedingungen zuverlässig detaillierte und genaue Informationen über die Umgebung liefern können. Daher werden in dieser Arbeit Deep-Learning-basierte Methoden zur Fahrzeuglokalisierung und -steuerung unter schwierigen Bedingungen vorgestellt.

Wir präsentieren effektive Methoden für eine robuste visuelle sensomotorische Steuerung, mit deren Hilfe nicht für jede wechselnde Umgebungsbedingung annotierte Daten erforderlich sind. Unsere vorgeschlagenen lernbasierten Ansätze kombinieren Transferlernen und Domänenanpassungsstrategien, um Informationen zwischen mehreren unterschiedlichen Umgebungsbedingungen zu übertragen. Die Experimente zeigen, dass unser Ansatz mit einer kleinen Teilmenge die gleiche Leistung erzielen kann wie mit der Gesamtmenge der annotierten Daten.

Des Weiteren stellen wir ein direktes visuelles Lokalisierungssystem vor, das für die Optimierung nicht den photometrischen, sondern den merkmalsmetrischen Fehler ausnutzt und damit die Robustheit gegenüber Veränderungen des visuellen Erscheinungsbildes verbessert. Unsere Methode kann selbstüberwacht trainiert werden und macht manuelle Annotationen überflüssig. Die Auswertungen zeigen, dass unser Ansatz klassische Methoden übertrifft und robuster gegen schlechte Initialisierung, Beleuchtungs- und Wetteränderungen ist.

Zudem stellen wir einen umfangreichen Benchmark-Datensatz vor, der aus anspruchsvollen Umgebungsbedingungen besteht, um visuelle SLAM- und langfristige Lokalisierungsalgorithmen zu bewerten. Unser Datensatz liefert bei einer Reihe von Durchläufen in denselben Szenen über das ganze Jahr hinweg genaue Referenzposen und Karten. Er ermöglicht, die Defizite aktueller Algorithmen zu verstehen und trägt dazu bei, den Stand der Technik bei der Langzeitlokalisierung und -kartierung zu verbessern.

ACKNOWLEDGMENTS

This thesis would not have been possible without the help, support, and guidance of many people.

First and foremost, I would like to thank my doctoral advisor, Prof. Daniel Cremers, for his excellent supervision and guidance throughout my PhD. I deeply appreciate the fact that he allowed me to pursue my research interests and supported the directions I took. I am very grateful for having such a great advisor and for becoming a part of an outstanding research group with so many talented and awesome people to learn from. Second, I thank Prof. Laura Leal-Taixé for her guidance, especially during the early stages of this work. Further, I would like to thank Prof. Nassir Navab and Prof. Wolfram Burgard for agreeing to serve as committee members for this dissertation.

I want to thank Quirin Lohr for his continuous technical support and Sabine Wagner for her assistance in administrative matters.

I also want to thank my excellent collaborators for their hard work and all the fruitful discussions: Lukas von Stumberg, Nan Yang, Niclas Zeller, Qadeer Khan, Qing Cheng, Rui Wang, and Torsten Schön.

I have also been fortunate to be surrounded by amazing people who made my life as a PhD student much more enjoyable. I hereby want to thank all my former colleagues at the Computer Vision Group at TUM and Artisense for the amazing time and the inspiring conversations.

I am very glad that I had the opportunity to do exciting research internships at two world-class research labs during my PhD. Firstly, I would like to thank my peers at Meta Reality Labs Research for the invaluable internship experience. Secondly, I would like to thank the whole research team at Niantic for the wonderful time. Even though the internships were only possible remotely due to the pandemic, I've enjoyed the opportunity to collaborate with teams of world-renowned researchers to work on the future of AR/VR technology.

Last but not least, I would like to thank my family and friends who always supported me and never pressured me to take any particular path. Thank you, Leonie, for everything.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

ATE	Absolute Trajectory Error
AR	Augmented Reality
CNN	Convolutional Neural Network
DoF	Degrees-of-Freedom
FPGA	Field Programmable Gate Array
FPS	Frames per Second
GN	Gauss-Newton
GAN	Generative Adversarial Network
GNSS	Global Navigation Satellite System
IMU	Inertial Measurement Unit
ICP	Iterative Closest Point
LM	Levenberg-Marquardt
LiDAR	Light Detection and Ranging
MVS	Multi-View Stereo
ORB	Oriented FAST and Rotated BRIEF
PnP	Perspective-n-Point
PCA	Principal Component Analysis
RANSAC	Random Sample Consensus
RTK	Real-Time Kinematic
ReLU	Rectified Linear Unit
RPE	Relative Pose Error
SIFT	Scale-Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
SfM	Structure-from-Motion
VR	Virtual Reality
VO	Visual Odometry
VIO	Visual-Inertial Odometry

Part I

INTRODUCTION AND PRELIMINARIES

1 INTRODUCTION

1.1 MOTIVATION

The field of deep learning has grown rapidly and has led to tremendous progress in computer vision over the last decade. While steady progress is primarily made for individual annotated datasets, the performance is typically limited to their domain. Hence, the natural next step is to foster the development of computer vision systems that are robust under all perceptual conditions and overcome the following challenges.

ILLUMINATION CONDITIONS In general, most computer vision applications are strongly sensitive to illumination changes. Severe changes in lighting can drastically reduce the accuracy of computer vision systems. One of the main tasks of robot vision is the development of robust and efficient systems that can operate reliably under any environmental condition.

VIEWPOINT VARIATIONS Many computer vision algorithms are inherently tailored toward specific viewpoints and thus often fail to generalize to new or unseen viewpoints. Therefore, one fundamental challenge is the development of methods that are viewpoint-agnostic.

VARIETY Improving the model's ability to generalize to new data is crucial for successfully deploying a deep learning system in the real world. Therefore, datasets should provide a significant variety in terms of pose, lighting, and textures allowing the neural network to capture all invariances present in the real world. It is therefore of utmost importance to train deep learning algorithms on large-scale and diverse datasets that provide variations in terms of camera viewpoint, environmental conditions, lighting conditions, weather and seasonal changes, etc.

LIMITED ANNOTATIONS One of the driving factors behind the success of computer vision is the amount of data we generate today. However, deep learning algorithms require enormous amounts of manually annotated training data. Obtaining annotations of high quality is not only a very tedious and expensive process but rather sometimes infeasible to do for every potential object of interest. Therefore, the urge to develop methods that can learn with limited or no human supervision at all is becoming inevitable.

While these challenges are quite broad and generally hold for most computer vision tasks, we approach this research gap and present techniques for vehicle localization and control under challenging conditions. For instance, a visual localization system, i.e. the task of estimating the 6DoF camera pose for a given image in a known map, must work in all conditions, no matter what. Consequently, one needs to develop algorithms that overcome those challenges. In this dissertation, we propose novel approaches for three challenging computer vision problems: robust sensorimotor control that addresses the problem of requiring annotations for every environmental condition, direct visual localization techniques that are robust to illumination and viewpoint changes, and a large-scale real-world dataset including large appearance variations caused by changes in the season and illumination for benchmarking visual SLAM and long-term localization.

1.2 LITERATURE OVERVIEW

In this section, we provide an overview of the relevant literature and the research context. We cover three main research topics. Firstly, we discuss sensorimotor control techniques, with a focus on deep learning-based approaches for vision-based vehicle control. Secondly, we discuss visual odometry and SLAM methods relevant to this work. Lastly, we provide an overview of different benchmarking datasets for long-term localization and SLAM in the context of autonomous driving.

SENSORIMOTOR CONTROL The safe navigation of autonomous vehicles in complex and unstructured environments remains a key challenge in robotics. Existing systems often rely on a modular pipeline consisting of separate, handcrafted components for planning and control [93]. However, the sequential structure of this approach involves manual heuristics and compounding errors. Instead, end-to-end sensorimotor control tackles the vision-based navigation problem by learning to act based on raw sensory data [15, 97]. In contrast to a modular pipeline, this approach directly learns to map from raw input data to control commands. Therefore, autonomous vehicles can directly learn driving policies from data without additional hand-engineering. However, these algorithms mainly perform well under the conditions for which they have been trained. Thus, the generalization capabilities to unseen conditions are typically limited. Therefore, a major challenge is the development of robust and reliable algorithms that are capable of operating in a variety of different conditions with as little annotated data as possible.

In this thesis, we tackle the problem of learning vision-based vehicle control policies that can generalize to challenging conditions with limited annotations. The key idea is to use semi-supervised and unsupervised domain adaptation methods to transfer driving policies across environmental and weather conditions. Usually, driving data with steering is collected once in a certain condition (source domain). However, collecting similar data for all other conditions (target domain) is generally an expensive and time-consuming process. Therefore, in the source domain, we have enough annotated data to train a supervised machine learning algorithm, while in the target domain, we have little or no annotated data, leading to poor generalization capabilities. To alleviate this problem, we propose using domain adaptation techniques that rely on limited or no annotations from the target domain. A great advantage is that it allows the use of passively collected datasets from other domains without labels. We demonstrate that the presented approaches for learning deep sensorimotor policies for vision-based vehicle control with limited annotations from the target domain achieve comparable performance to their supervised counterparts.

VISUAL ODOMETRY AND SLAM In computer vision and robotics, understanding the 3D world around us is a fundamental aspect of perception and navigation. It allows us to explore and navigate in unknown environments, experience augmented reality (AR) applications, or help robots perceive their surroundings to safely move around. A key enabler to reconstructing the 3D world, i.e. incrementally inferring the 3D geometry and tracking the camera motion from a sequence of 2D images, is commonly referred to as simultaneous localization and mapping (SLAM). In general, the task of localization and reconstruction can be performed either *sparsely* or *densely*. Sparse methods [34, 87] usually rely only on a selected set of points (e.g. corners), whereas dense methods [64, 91] attempt to leverage a dense set of points to reconstruct the majority or all pixels of the 2D image domain. Moreover, visual SLAM methods can be classified as *indirect* or *direct*. Direct methods [34, 91] directly operate on raw pixel intensities provided by the camera sensor. For passive sensors, the direct approach relies on optimizing a so-called photometric error, i.e. by comparing the intensities between the pixels in one image and its warped projection in another image. The name direct stems from the fact, that the process uses the actual measurements from the raw sensor without pre-processing. Indirect methods [72, 87] involve an additional pre-processing step, i.e. the matching of correspondences between a reference image and a target image. Usually, this is done by extracting and matching a set of keypoints. It then optimizes a geometric objective, namely the reprojection error. This generally leads to good convergence, since the matching points do not necessarily have to be spatially close to each other. However, one fundamental limitation of an indirect setting is the ability to correct sensor errors, since the approach only leverages matched points rather than raw sensor measurements. Direct methods instead are advantageous in averaging out sensor noise, since this approach is based on raw sensor values. However, directly operating on the raw sensor values comes with a limited convergence basin, requiring good initialization for the pose.

This thesis develops novel solutions for improving two major drawbacks of direct methods: (1) requiring good initialization, and (2) susceptibility to challenging lighting and weather conditions. This is mainly achieved by replacing images with learned multiscale deep features and replacing the classical photometric error with a featuremetric error. Using deep features for the optimization yields a wider convergence despite significant appearance changes compared to classical photometric alignment using images.

BENCHMARKING DATASETS FOR LONG-TERM SLAM A key enabler for advancing the field of visual odometry and SLAM has been the availability of large-scale datasets for benchmarking these algorithms. In the context of autonomous driving, one of the most popular benchmarks is probably KITTI [41]. However, the data has mainly been collected in clear weather and daytime conditions. Nevertheless, it is of utmost importance to develop mapping and localization algorithms that maintain satisfactory performance even under challenging conditions. Concerning long-term SLAM datasets, the Oxford RobotCar Dataset [82] pioneered this area by providing a large-scale dataset consisting of sequences recorded multiple times for the same environment over one year. It covers challenging variations in appearance such as nighttime, rain, and snow as well as structural changes. However, the variety of scenarios is mainly limited to an urban environment. While many of the existing datasets focus on long-term SLAM, they lack sequential structure [128, 131] or accurate ground truth [115, 131], only provide a certain adverse condition [96], or focus on AR scenarios [113].

This thesis presents a benchmarking dataset for autonomous driving containing nine different scenarios collected in all seasons of the year. The dataset consists of various recording areas, ranging from urban driving, i.e. exhibiting many dynamic objects, to rural areas, i.e. with homogenous and repetitive structures. Moreover, our large-scale dataset provides a wide variety of appearance, illumination, and weather changes. The dataset was deliberately recorded by traversing each scenario several times, yielding diversity in the environmental conditions. It allows benchmarking visual odometry, global place recognition, and map-based visual localization. Overall, we believe that this dataset will help to understand the limitations of current state-of-the-art mapping and localization approaches and advance future research on long-term SLAM.

1.3 OUTLINE OF THE THESIS

This thesis is structured into four main parts as follows:

Part I introduces the motivation of the research challenges tackled in this thesis. Moreover, the underlying theoretical foundations are provided. Chapter 1 gives a motivation for the research topic. Chapter 2 summarizes the main contributions of this cumulative dissertation along with an overview of the respective original publications. Chapter 3 introduces the fundamental computer vision and deep learning concepts and mathematical tools used throughout the thesis.

Part II presents the five peer-reviewed publications that form the cumulative content of this thesis. First, in Chapter 4, we propose a method for transferring sensorimotor control commands across different weather conditions by means of semantic segmentations [5]. Second, a framework to deal with limited annotated training data for the application of vision-based vehicle control is introduced in Chapter 5 [2]. Third, Chapter 6 proposes a novel paradigm for 6DoF camera tracking by combining direct alignment approaches with high-dimensional learned feature representations that are robust to large illumination and viewpoint changes [3]. Fourth, Chapter 7 proposes a large-scale real-world outdoor dataset with SLAM localization and maps from multiple runs in the same scenes exhibiting changing environmental conditions [7]. Fifth, Chapter 8 presents a direct method for visual localization based on a loss formulation inspired by the classical Levenberg-Marquardt algorithm, improving the robustness against bad initialization.

Part III provides a summary of the contributions and a discussion of the proposed techniques in Chapter 9. Finally, Chapter 10 concludes the thesis with an outlook on avenues for future research.

Part IV contains the original versions of the papers [2–5, 7] along with the specific individual contributions of the author of this thesis.

2 CONTRIBUTIONS

This thesis develops novel methods for vehicle localization and control under challenging conditions. In this chapter, we summarize the contributions of the publications that investigate this research problem and form the cumulative content of this dissertation.

2.1 LIST OF PUBLICATIONS

This cumulative thesis comprises five full-length publications [2–5, 7] included in Chapters 4, 5, 6, 7, and 8. These peer-reviewed papers are the result of collaborations with Daniel Cremers, Laura Leal-Taixé, Lukas von Stumberg, Nan Yang, Niclas Zeller, Qadeer Khan, Qing Cheng, and Rui Wang. These papers were published in highly ranked and peer-reviewed international conferences and journals. Table 2.1 provides an overview of the publications that contribute to this cumulative dissertation. In addition, the table contains other co-authored papers published while pursuing this degree, which are not included as a contribution.

2.2 MAJOR CONTRIBUTIONS

The key contributions of this cumulative dissertation are threefold. First, we explore vision-based sensorimotor control approaches for learning intelligent agents to perform useful actions based on raw sensory observations under challenging perceptual conditions. Second, we propose novel approaches for visual localization using featuremetric optimization via direct alignment. Third, we present a large-scale benchmark dataset covering seasonal and challenging perceptual conditions for autonomous driving.

2.2.1 *Toward Robust Sensorimotor Control*

Chapter 4 is based on [5] and proposes a knowledge distillation approach for transferring vehicle control steering labels across different weather conditions using semantic maps without requiring to collect new ground truth labels. Even though end-to-end supervised learning methods have shown promising results for sensorimotor control tasks, their performance is mainly influenced by the data it was trained on. Sensorimotor control models that have been trained under one weather condition usually show poor generalization capabilities to unseen weather conditions. In this work, we propose to split the

Table 2.1: **Full list of publications.** Full list of peer-reviewed publications done within the course of this degree, in chronological order. The five publications this cumulative dissertation is based on are listed in **black**, with references to the respective chapters. Research papers not included in this thesis are marked in **gray**.

Modular Vehicle Control for Transferring Semantic Information Between Weather Conditions Using GANs. Patrick Wenzel, Qadeer Khan, Daniel Cremers, and Laura Leal-Taixé. In: *Conference on Robot Learning (CoRL)*. 2018 [5] (Chapter 4).

Towards Generalizing Sensorimotor Control Across Weather Conditions. Qadeer Khan, Patrick Wenzel, Daniel Cremers, and Laura Leal-Taixé. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019 [2] (Chapter 5).

GN-Net: The Gauss-Newton Loss for Multi-Weather Relocalization. Lukas von Stumberg, Patrick Wenzel, Qadeer Khan, and Daniel Cremers. In: *IEEE Robotics and Automation Letters (RA-L)*. 2020 [3] (Chapter 6).

4Seasons: A Cross-Season Dataset for Multi-Weather SLAM in Autonomous Driving. Patrick Wenzel, Rui Wang, Nan Yang, Qing Cheng, Qadeer Khan, Lukas von Stumberg, Niclas Zeller, and Daniel Cremers. In: *German Conference on Pattern Recognition (GCPR)*. 2020 [7] (Chapter 7).

LM-Reloc: Levenberg-Marquardt Based Direct Visual Relocalization. Lukas von Stumberg, Patrick Wenzel, Nan Yang, and Daniel Cremers. In: *International Conference on 3D Vision (3DV)*. 2020 [4] (Chapter 8).

Self-Supervised Steering Angle Prediction for Vehicle Control Using Visual Odometry. Qadeer Khan, Patrick Wenzel, and Daniel Cremers. In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2021 [1].

Vision-Based Mobile Robotics Obstacle Avoidance With Deep Reinforcement Learning. Patrick Wenzel, Torsten Schön, Laura Leal-Taixé, and Daniel Cremers. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2021 [6].

task of vehicle control into two separate modules: a control module that is trained on one weather condition with labeled steering data and a perception module that is used as an interface between the unseen weather conditions and the control module. The semantic maps for the unseen conditions needed to train the perception module are generated using a generative adversarial network (GAN) in an unsupervised manner. We use a teacher-student framework for transfer learning, wherein the teacher network (semantic labels are available) trains the student network (semantic labels are not available). The results show that our proposed approach trained on ground truth data from a single weather condition achieves performance on par with an end-to-end model trained with ground truth steering labels for all weather conditions.
Chapter 5 is based on [2] and proposes a framework to cope with scarce annotated training data for the application of vision-based vehicle control. In general, the ability of deep learning models to generalize beyond the domain they have been trained is limited. However, labeling large amounts of data for all possible scenarios that a model may encounter is not always feasible and sometimes impossible. In this work, we overcome the need to have access to labeled data for all conditions and show how limited data with steering labels for a single condition can be transferred to multiple different conditions, i.e. weather conditions. This is achieved by leveraging unlabeled images in a teacher-student framework complemented with an image-to-image translation network. The translation network transfers the images to a new domain, whereas the teacher network provides *soft labels* to teach the student network on this domain. The experimental results show that the proposed approach generalizes across different weather conditions using only ground truth steering labels from one domain.

2.2.2 *Direct Visual Localization*

Chapter 6 is based on [3] and proposes GN-Net: a network optimized with the novel Gauss-Newton loss for training weather and lighting invariant deep features, tailored for direct alignment. To achieve this, we learn dense descriptors using pixel-wise correspondences between images from different scenes. For learning the feature representations of the images, we leverage convolutional neural networks and their ability to extract hierarchical feature maps in a coarse-to-fine manner. These learned representations are robust to large illumination changes and provide a larger convergence basin. In comparison, classical direct image alignment algorithms operate directly on pixel intensities and are therefore not robust to strong appearance changes. The experimental evaluation shows that our proposed approach is more robust against bad initialization, variations in the daytime, and weather changes compared to the state-of-the-art.

Chapter 8 is based on [4] and introduces LM-Reloc: a novel approach for visual localization based on the direct alignment of multiscale deep features. In this follow-up work to GN-Net, we propose a loss formulation inspired by the classical Levenberg-Marquardt algorithm to train LM-Net. Furthermore, we propose a camera pose estimation network, CorrPoseNet, which regresses the relative camera pose to bootstrap the direct alignment. The final relative 6DoF pose between a reference and query image is then obtained by classical Levenberg-Marquardt optimization in a coarse-to-fine manner leveraging the learned multi-scale deep features. The learned deep features significantly improve the robustness of the alignment, especially for localization candidates exhibiting large appearance changes. This results in a reliable camera pose estimation without relying on feature matching and RANSAC, allowing the utilization of any region in an image with sufficient gradients.

2.2.3 *4Seasons Dataset*

Chapter 7 is based on [7] and presents a novel large-scale benchmark dataset for autonomous driving with globally-consistent reference poses with up to centimeter-level accuracy obtained from the fusion of direct stereo visual-inertial odometry with RTK-GNSS. The data has been captured in a large variety of different environments under challenging perceptual conditions. This dataset allows benchmarking long-term visual odometry and visual localization. The data collected under a wide variety of weather conditions and illuminations results in more than 300 km of recordings in nine different environments ranging from multi-level parking garages over urban (including tunnels) to the countryside and highways. Figure 2.1 shows some example images from the dataset.

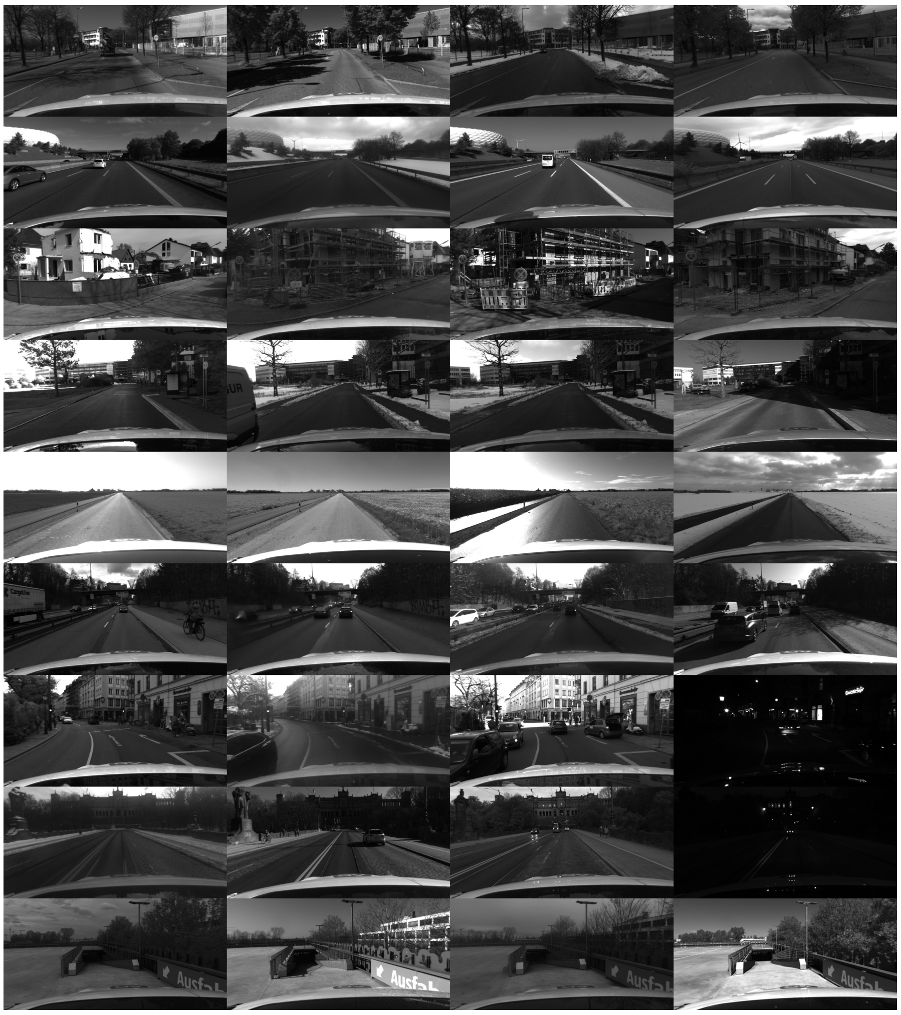


Figure 2.1: **The 4Seasons dataset.** Illustration of a single frame from each of the captured sequences. Note the wide variety of covered environments and large changes in appearance. It includes short-term illumination and weather changes, as well as long-term seasonal and structural changes.

3 FUNDAMENTALS

This chapter introduces the fundamental computer vision and deep learning concepts and mathematical tools that are the foundation of the approaches presented in this thesis. We first describe the mathematical preliminaries together with the pinhole camera model and nonlinear optimization techniques. We then give an overview of deep learning, in particular an overview of feed-forward neural networks, convolutional neural networks, and techniques for regularization and optimization functions that can be used for training the networks.

3.1 MATHEMATICAL PRELIMINARIES

Notation: Scalars are denoted as $c \in \mathbb{R}$ with regular (lowercase) letters, while vectors $\mathbf{x} \in \mathbb{R}^n$ (lowercase), and matrices $\mathbf{A} \in \mathbb{R}^{m \times n}$ (uppercase) are written in bold letters. We denote 3D points as $\mathbf{p} = [x, y, z]^\top \in \mathbb{R}^3$, and 2D pixels as $\mathbf{x} = [u, v]^\top \in \mathbb{R}^2$. The respective homogeneous coordinates are denoted as $\tilde{\mathbf{p}} = [x, y, z, 1]^\top \in \mathbb{R}^4$, and as $\tilde{\mathbf{x}} = [u, v, 1]^\top \in \mathbb{R}^3$.

3.1.1 *Rigid Body Motion*

Elements of the 3D rotation group, $\mathbf{SO}(3)$ with 3 Degrees-of-Freedom (DoF) are represented by 3D rotation matrices:

$$\mathbf{SO}(3) = \left\{ \mathbf{R} \in \mathbb{R}^{3\times3} \;\middle|\; \mathbf{R}^\top \mathbf{R} = \mathbf{I}_n \right\}, \tag{3.1}$$

where $\mathbf{I}_n$ denotes the identity matrix. The special Euclidean group $\mathbf{SE}(3)$ is defined as follows:

$$\mathbf{SE}(3) = \left\{ \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0^\top & 1 \end{bmatrix} \;\middle|\; \mathbf{R} \in \mathbf{SO}(3), \mathbf{t} \in \mathbb{R}^3 \right\}, \tag{3.2}$$

where $\mathbf{t} \in \mathbb{R}^3$ is a 3D translation vector.

3.1.2 *Pinhole Camera Model*

A camera model describes the basic geometric relationship between a 3D scene and its projections onto a 2D image plane. In the pinhole camera model, the camera aperture is modeled as a single point and no lenses are used for focusing the light. The intrinsic camera parameters describe the mapping of the scene in front of the camera (3D points in the local camera coordinate frame) to the 2D pixels in the final image using a projection function.

Pinhole Projection Model: In the basic pinhole camera model, for a given point $\mathbf{p} = [x, y, z]^\top \in \mathbb{R}^3$ in 3D, the general perspective projection function $\pi : \mathbb{R}^3 \to \mathbb{R}^3$ is defined as:

$$\pi(\mathbf{p}) = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{z} \begin{bmatrix} x \\ y \\ z \end{bmatrix} . \tag{3.3}$$

The intrinsic camera calibration matrix $\mathbf{K} \in \mathbb{R}^{3\times3}$ is then defined as:

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} , \tag{3.4}$$

where f_x, f_y corresponds to the focal length, and c_x, c_y to the principal point of the camera. The projected homogeneous 2D pixel position $\tilde{\mathbf{x}}$ of a 3D point $\mathbf{p}$ can then be calculated as:

$$\tilde{\mathbf{x}} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K}\pi(\mathbf{p}). \tag{3.5}$$

For a 2D point with known depth z, the back-projection to 3D is given by the inverse projection $\pi^{-1} : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^3$:

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \pi^{-1}\left(\begin{bmatrix} u \\ v \end{bmatrix}, z\right) = \mathbf{K}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} z. \tag{3.6}$$

3.1.3 *Nonlinear Least-Squares Optimization*

Many optimization problems in computer vision can be formulated such that the objective is to minimize an energy function that takes the form of a sum of squared residual terms. In general, the energy function can be written as:

$$E(\mathbf{x}) = \sum_i r_i^2(\mathbf{x}), \tag{3.7}$$

where r_i is the i-th residual term, and E is the optimization objective.

In most cases, the residual terms are a nonlinear function of the optimization variables, making the energy function E nonlinear as well. In the following, we introduce the Gauss-Newton and the Levenberg-Marquardt method. Both of these are second-order, iterative optimization methods for minimizing least-squares problems.

Gauss-Newton: The Gauss-Newton (GN) algorithm is an iterative method to compute and solve a quadratic approximation to E by linearizing the residual function $\mathbf{r}$, using the first-order Taylor expansion to approximate the objective function linearized around $\mathbf{x}_0$:

$$\mathsf{E}(\mathbf{x}_0 + \boldsymbol{\delta}) = \|\mathbf{r}(\mathbf{x}_0 + \boldsymbol{\delta})\|^2 \tag{3.8}$$

$$\approx \|\mathbf{r}_0 + \mathbf{J}_\mathbf{r}\boldsymbol{\delta}\|^2 \tag{3.9}$$

$$= \mathbf{r}_0^\top \mathbf{r}_0 + 2\boldsymbol{\delta}^\top \mathbf{J}_\mathbf{r}^\top \mathbf{r}_0 + \boldsymbol{\delta}^\top \mathbf{J}_\mathbf{r}^\top \mathbf{J}_\mathbf{r}\boldsymbol{\delta}, \tag{3.10}$$

with

$$\mathbf{J}_\mathbf{r} = \left.\frac{d\mathbf{r}(\mathbf{x})}{d\mathbf{x}}\right|_{\mathbf{x}_0} \qquad \text{and} \qquad \mathbf{r}_0 = \mathbf{r}(\mathbf{x}_0). \tag{3.11}$$

To evaluate the optimal value of $\boldsymbol{\delta}$ that minimizes the energy function, we set its derivative to zero. This leads to the update step of the Gauss-Newton algorithm:

$$\mathbf{J}_\mathbf{r}^\top \mathbf{J}_\mathbf{r}\boldsymbol{\delta} + \mathbf{J}_\mathbf{r}^\top \mathbf{r}_0 = 0. \tag{3.12}$$

Solving for the increment $\boldsymbol{\delta}$ gives:

$$\boldsymbol{\delta} = -(\mathbf{J}_\mathbf{r}^\top \mathbf{J}_\mathbf{r})^{-1}\mathbf{J}_\mathbf{r}^\top \mathbf{r}_0 = -\mathbf{H}^{-1}\mathbf{b}. \tag{3.13}$$

To obtain the final estimate, the increment is added to the evaluation point and repeated until convergence:

$$\mathbf{x} \leftarrow \mathbf{x}_0 + \boldsymbol{\delta}. \tag{3.14}$$

Levenberg-Marquardt: The Levenberg-Marquardt (LM) [73, 83] algorithm extends the Gauss-Newton algorithm by adding a damping factor λ to the update step to better condition the updates and make the optimization more robust. This is done by adaptively switching between gradient descent steps and Gauss-Newton updates. If λ is large, the update step will be close to a gradient descent step, while if λ is small, the update step will be close to a Gauss-Newton step. On the one hand, the Gauss-Newton algorithm shows fast quadratic convergence close to the local minimum, however, it can become unstable for poor initializations. On the other hand, gradient descent shows slow linear convergence, however, it is guaranteed to decrease the function for a sufficiently small step size. The updates can then be computed as:

$$\boldsymbol{\delta} = -\left(\mathbf{H} + \lambda\,\mathrm{diag}(\mathbf{H})\right)^{-1}\mathbf{b}, \tag{3.15}$$

where the diagonal matrix $\mathrm{diag}(\mathbf{H})$ contains the diagonal entries of $\mathbf{H}$.

3.2 DEEP LEARNING

In this section, we introduce the basic concepts of deep learning, which are fundamental to the work presented in this thesis. In particular, we focus on deep neural networks and their use for computer vision tasks. Deep learning architectures such as deep neural networks and convolutional neural networks have made tremendous success in many areas and in particular have revolutionized computer vision research. Deep learning is a subset of machine learning where neural networks are used to learn from huge amounts of data. For a more comprehensive introduction, we recommend the Deep Learning book from Goodfellow et al. [43].

3.2.1 *Types of Learning*

In general, learning problems in machine learning can be separated into three main types: supervised, unsupervised, and reinforcement learning. However, since the boundaries between supervised and unsupervised learning are not straight, we will also introduce some more common hybrid types of learning: semi-supervised learning, and self-supervised learning.
Supervised Learning: The most common type of learning problem in machine learning is supervised learning. Supervised learning can be described as training a model that enables the mapping between input data and target variables. We assume that we have a dataset comprised of observations of the input variable together with corresponding observations of the target variable. The complete dataset is usually split into a training set, validation set, and test set. The model is fit on the training set consisting of n examples $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ and used to make predictions on the test set. To evaluate the accuracy of the learned model $(f : X \mapsto Y)$, the output predictions $\hat{y}_i = f(x_i)$ from the model are compared to the withheld true target variables y_i from the test set. To train the model parameters, we define a scalar-valued loss function $L(f(\mathbf{x}), \mathbf{y})$ that measures the discrepancy between a true target label y_i and the model prediction, i.e. $\hat{y}_i = f(x_i)$ for some $f \in \mathcal{F}$, where $\mathcal{F}$ denotes some particular class of functions.

Thus, our objective in learning is to find $f^* \in \mathcal{F}$ that minimizes the expected loss over the data generating distribution $\hat{p}_{\text{data}}$:

$$f^* = \arg\min_{f \in \mathcal{F}} \mathbb{E}_{(\mathbf{x},\mathbf{y}) \sim \hat{p}_{\text{data}}} L(f(\mathbf{x}), \mathbf{y}), \tag{3.16}$$

where $f(\mathbf{x})$ is the predicted output for input $\mathbf{x}$. However, in practice, we do not have access to all the elements of the data generating distribution $\hat{p}_{\text{data}}$, making the optimization problem intractable. Therefore, we usually approximate the expected loss in Equation (3.16) with

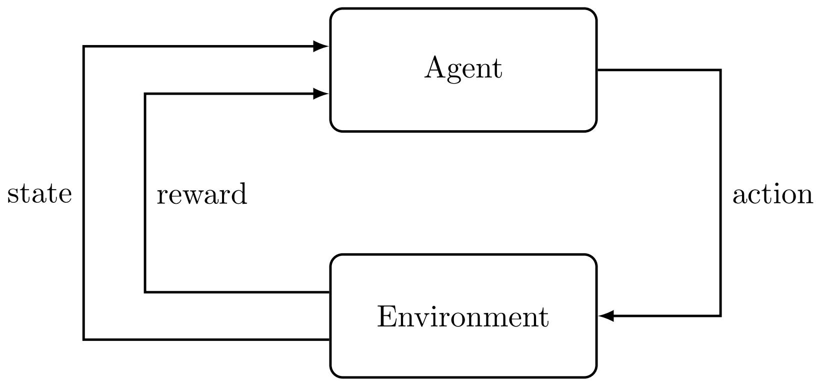


Figure 3.1: **The agent-environment interaction loop in a Markov decision process.** An agent interacts with its environment by taking actions, yielding and receiving rewards, and an updated representation of the state.

sampling by averaging the loss over the examples from the training set:

$$f^* \approx \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} L\left(f(x_i), y_i\right). \tag{3.17}$$

The two most well-known supervised learning problems are regression and classification. Their main difference is in the representation of the target labels. In regression problems, the labels are continuous, whereas, in classification problems the labels are categorical.
Unsupervised Learning: In comparison to supervised learning, unsupervised learning leverages only the input data without having access to target variables. Therefore, unsupervised learning needs to learn meaningful patterns from data without the possibility of being guided through labels.

Despite the existence of many different unsupervised learning problems, the two main common problems that are frequently encountered are clustering and density estimation. A clustering algorithm deals with finding groups or clusters of similar data points in an unlabelled dataset. Density estimation is the process of estimating the underlying probability density function based on observed data points.
Reinforcement Learning: Reinforcement learning is a general learning framework that describes a class of problems where an agent interacts in an environment and learns to operate using rewards. The typical agent-environment interaction loop of a reinforcement learning scenario is illustrated in Figure 3.1. An agent interacts with its environment in discrete time steps. At each time step t, the agent receives the current state s_t and on that basis chooses an action a_t. In the next time step, the agent receives a reward r_{t+1} based on the selected action and ends up in a new state s_{t+1}.

Reinforcement learning problems can be modeled as a Markov decision process (MDP). An MDP is a 4-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, where $\mathcal{S}$ is a set of states called the state space, $\mathcal{A}$ is a set of actions called the action space, $\mathcal{P}$ is the probability that action a in state s at time t will lead to state s_{t+1} at time $t+1$, and $\mathcal{R}$ is the reward (or expected reward) received after transitioning from state s_t to state s_{t+1}. To solve the problem, we introduce the notion of a policy function $\pi : \mathcal{S} \mapsto \mathcal{A}$, that maps from state space $\mathcal{S}$ to action space $\mathcal{A}$. The goal of the agent is to find a policy that maximizes the cumulative reward through repeated interaction with the environment. We, therefore, introduce the notion of a state-value function:

$$v_\pi(s) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s \right], \tag{3.18}$$

where $\gamma \in [0, 1]$ is a discount factor that trades off the importance of immediate vs. long-term rewards. To further describe the impact of a particular action, we can define an action-value function:

$$q_\pi(s, a) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a \right]. \tag{3.19}$$

The optimal state-value function $v^*(s)$ is the maximum state-value function over all policies:

$$v^*(s) = \max_\pi v_\pi(s). \tag{3.20}$$

The optimal action-value function $q^*(s, a)$ is the maximum action-value function over all policies:

$$q^*(s, a) = \max_\pi q_\pi(s, a). \tag{3.21}$$

The optimal value function specifies the best possible performance in the MDP. A policy that achieves the optimal state-value and action-value function is called an optimal policy and is usually denoted by π^* (there can be multiple optimal policies). An optimal policy can be found by maximizing over $q^*(s, a)$:

$$\pi^*(s, a) = \begin{cases} 1 & \text{if } a = \arg\max_{a \in \mathcal{A}} q^*(s, a) \\ 0 & \text{otherwise} \end{cases} \tag{3.22}$$

Semi-Supervised Learning: Semi-supervised learning is a special case of supervised learning where we only have access to labels for a small subset of the training set. The main goal of semi-supervised learning algorithms is to make effective use of all the available data and not just the labeled data as in a supervised setting.
Self-Supervised Learning: Self-supervised learning can be interpreted as an unsupervised learning problem that is framed as a supervised

learning problem such that algorithms from supervised learning could be applied. Therefore, self-supervised learning methods try to obtain supervisory signals from the data itself, without dependence on manual annotations. Thus, the general idea of self-supervised learning is to predict an unobserved or hidden part of the input based on some known part of the input.

In recent years, contrastive self-supervised learning approaches have attracted great attention and surpassed their supervised learning counterparts on various computer vision downstream tasks. The main idea behind contrastive learning is the use of distance-based losses, such that similar samples are grouped closer together and dissimilar samples are far from each other. Furthermore, generative models can also be used to learn representations without explicit supervision. An autoencoder [49] is a type of neural network that is used to learn low-dimensional features, which would allow for an accurate reconstruction of the input. This is achieved via an encoder-decoder architecture that is separated by a bottleneck that represents the internal compact representation of the input. Generative adversarial networks (GANs) [44] are powerful generative models that are used to learn the underlying distribution of the training data to create new data instances. GAN models consist of two separate networks, a generator and a discriminator that are trained in the form of a min-max two-player game. The generator aims to generate realistic-looking *fake* samples, and the discriminator's objective is to estimate the probability that a sample is *real* or *fake*.

3.2.2 *Regularization*

A common problem for deep neural networks is the issue that trained models can not generalize well to unseen data. This is mainly attributed to the following reasons. First, a model with too little capacity cannot fit the data. Second, a model with too much capacity can fit the data too well and overfit the training dataset. Underfitting can be reduced by increasing the model capacity, however, to prevent deep neural networks from overfitting the use of regularization techniques is required. Lastly, a naive solution to Equation (3.17) could be a function f that perfectly maps each sample pair in the training data, however, returns zero elsewhere. This would yield a function that would most likely not generalize to all $(\mathbf{x}, \mathbf{y}) \sim \hat{p}_{\text{data}}$. To overcome these issues, a regularization factor R is added to the loss objective:

$$f^* \approx \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} L\left(f(x_i), y_i\right) + R(f). \tag{3.23}$$

In the following, we present the most commonly used regularizations in deep neural networks, i.e. L_1 and L_2 regularization. The L_1 regularization on the model parameter θ is defined as:

$$R(\theta) = \|\theta\|_1 = \sum_i |\theta_i|. \tag{3.24}$$

L_1 regularization is a good choice when the number of features is high since it provides sparse solutions. Therefore, features with zero coefficients can be ignored and it is acting as a form of feature selection. Moreover, L_1 regularization is robust to outliers.

The L_2 regularization on the model parameter θ is defined as:

$$R(\theta) = \frac{1}{2}\|\theta\|_2^2 = \frac{1}{2}\sum_i \theta_i^2. \tag{3.25}$$

L_2 regularization provides non-sparse solutions and, therefore, does not exhibit an implicit feature selection property. In contrast to L_1 regularization, it is not robust to outliers.

Data Augmentation: In practice, deep learning algorithms require tremendous amounts of annotated training data. Mostly, the labeling process is not only tedious, error-prone, and expensive due to manual interactions, it is sometimes infeasible to obtain labels for every potential object of interest. However, having access to more data is a practical way to get better consistent deep learning models. Unfortunately, in a real-world application, the amount of training data is typically limited. Data augmentation is one way of alleviating this problem. It provides a simple and cheap way to increase the size of the training dataset. Common data augmentations applied to images may include brightness changes, perspective transformations, random rotations, blurring, scaling, etc.

Dropout: Dropout [125] is another popular technique for reducing overfitting in deep neural networks. It is achieved by ignoring randomly selected nodes during training. This effectively forces hidden nodes to become less interdependent and to be ignored in a particular forward or backward pass. Dropout can be easily implemented by randomly selecting nodes to be *dropped out* with a given probability. In practice, dropout is usually only applied at training time and is not used at inference time to preserve a deterministic behavior. Dropout can essentially be interpreted as a way to learn an ensemble of many different networks since, at each training step, randomly chosen nodes are temporarily removed from the network. Figure 3.2 shows a typical neural network where all nodes are connected and activated (before applying dropout) on the left. On the right, after applying dropout, the crossed nodes have been dropped and all the associated connections are removed.

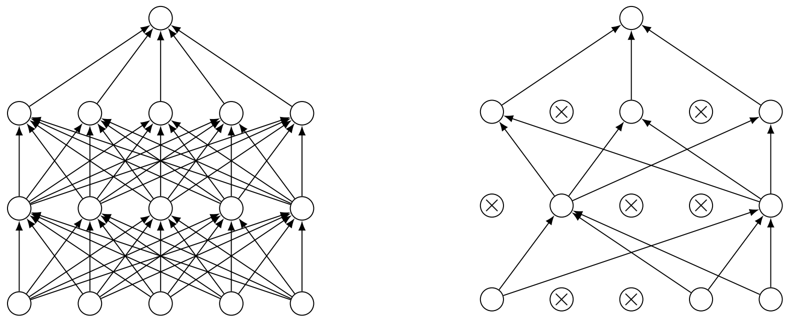

(a) Before applying dropout. (b) After applying dropout.

Figure 3.2: **Dropout regularization in a neural network.** Left: A neural network before applying dropout. Right: The same neural network after applying dropout. The crossed units have been *dropped* out.

3.2.3 *Optimization*

In this section, we introduce the techniques used for training neural networks. In general, the problem of learning the parameters θ of a model for a supervised learning task can be formulated as an optimization problem of the general form:

$$\theta^* = \arg\min_{\theta} g(\theta), \tag{3.26}$$

where $g(\theta) = \frac{1}{n}\sum_{i=1}^{n} L\left(f_\theta(x_i), y_i\right) + R(f_\theta)$. The objective function of neural networks is highly non-convex. Thus using gradient-based learning methods, we are not guaranteed to converge to the global minima. Therefore, in practice, proper weight initialization is important to reach a *good* local minima. Most commonly, the network weights are updated through backpropagation and variants of gradient descent optimization algorithms.

Gradient Descent Optimization: Gradient descent is a commonly used first-order iterative optimization algorithm to find a local minimum of a differentiable objective function. The algorithm iteratively updates the parameters in the negative direction of the loss objective. In practice, for training deep neural networks, we only approximate the gradients using a small minibatch of examples since calculating derivatives on the entire dataset is usually intractable. The parameters are then updated using stochastic gradient descent (SGD):

$$\theta \leftarrow \theta - \eta \nabla_\theta g(\theta), \tag{3.27}$$

where η is called step size or learning rate and ∇_θ denotes the gradient w.r.t. θ.

More recently, advanced iterative optimization methods for obtaining faster convergence have been proposed. Commonly used optimization schemes include: SGD with momentum [129], Adadelta [145], Adagrad [32], RMSProp [134], and Adam [65].

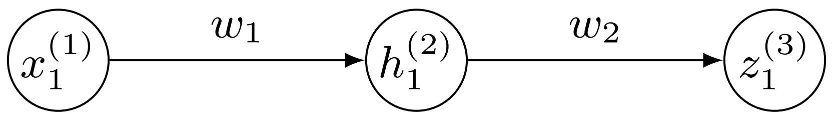


Figure 3.3: **Illustration of backpropagation.** A simple graph for illustrating backpropagation within deep neural networks.

Backpropagation: Backpropagation is a commonly used concept for efficiently computing the gradients of loss functions with respect to their inputs. The algorithm works by decomposing the parameters of the network to a sequence of differentiable functions for computing the gradients one layer at a time using the chain rule. The calculation of gradients is usually computed backward through the network, i.e. starting with the gradient of the final layer and chaining backward to the gradient of the first layer. This is a very efficient method to update the weights in a neural network since the weights are updated layer by layer and gradients do not need to be recomputed. Figure 3.3 shows a simple illustration of a neural network consisting of one input neuron, one hidden layer neuron, and one output neuron to demonstrate how the chain rule works. For simplicity, we denote the loss based on the output $z_1^{(3)}$ and the ground truth value to be $\mathcal{L}$. To update the loss w.r.t. w_1, we apply the chain rule as follows:

$$\frac{\delta\mathcal{L}}{\delta w_1} = \frac{\delta\mathcal{L}}{\delta z_1^{(3)}}\frac{\delta z_1^{(3)}}{\delta h_1^{(2)}}\frac{\delta h_1^{(2)}}{\delta w_1}. \tag{3.28}$$

3.2.4 *Nonlinear Activation Functions*

In a neural network, the nonlinear activation function transforms the summed weighted input from a node into its corresponding output. Nonlinear mapping helps the network to learn complex patterns in the data. In this section, we present the most commonly used nonlinear activation functions for neural networks.
The **Logistic Sigmoid** function

$$\mathrm{Sigmoid}(x) = \sigma(x) = \frac{1}{1+\exp(-x)} \tag{3.29}$$

transforms the input to the range $[0, 1]$. In practice, the sigmoid activation function has two major drawbacks. First, the output is not zero-centered, which may lead to undesirable zig-zagging behavior in the gradient updates of the weights. Second, very small and large input values easily lead to vanishing gradients.
The **Tanh** function

$$\tanh(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)} \tag{3.30}$$

transforms the input to the range $[-1, 1]$ and therefore leads to a zero-centered output and is preferably chosen over the sigmoid non-linearity.
The **Rectified Linear Unit (ReLU) [90]** function

$$\mathrm{ReLU}(x) = (x)^{+} = \max(0, x) \tag{3.31}$$

has become very popular in the last few years. It is a piecewise linear function and the activation is simply thresholded at zero. The function is not differentiable at $x = 0$ and is not zero-centered. However, the convergence of stochastic gradient descent is much faster compared to the sigmoid or tanh functions [67]. Unfortunately, one major drawback of ReLU units is that they can be very fragile during training and can *die*, i.e. only output zero for any input. In order to overcome the *dying ReLU* problem, Leaky ReLUs [81] and Exponential Linear Units (ELUs) [23] are introduced.
The **Softmax** function

$$\mathrm{Softmax}(x_i) = \frac{\exp x_i}{\sum_j \exp(x_j)} \tag{3.32}$$

is a function that ensures that the output is a probability distribution: each element is non-negative and the summation of those values is guaranteed to equal 1. It is most commonly used as the last operation in a neural network to convert logits to probabilities for multi-label classification problems. Please note that the softmax function is a generalization of the sigmoid function. For the task of binary classification (2 classes), softmax reduces to sigmoid.

3.2.5 *Feed-Forward Neural Networks*

Deep neural networks are composed of several layers and are capable of learning complex functions. A neural network can be constructed by connecting different units, also called neurons. At a basic level, a neural network consists of three main components: an input layer, multiple hidden layers, and an output layer. In principle, neural networks can take any arbitrary network structure, however, feed-forward neural networks are one of the first and most well-known architectures. A feed-forward neural network can also be interpreted as a form of a directed acyclic graph. Figure 3.4 illustrates a simple three-layer feed-forward neural network. We denote the neural network as a three-layer network since the input layer is usually not counted. This network is fully connected since each neuron in layer l is connected to all input neurons from layer $l-1$.

The neurons are the smallest building blocks and an elementary component in neural networks. A neuron is used to compute a linear combination of its inputs. In general, given D input neurons, each

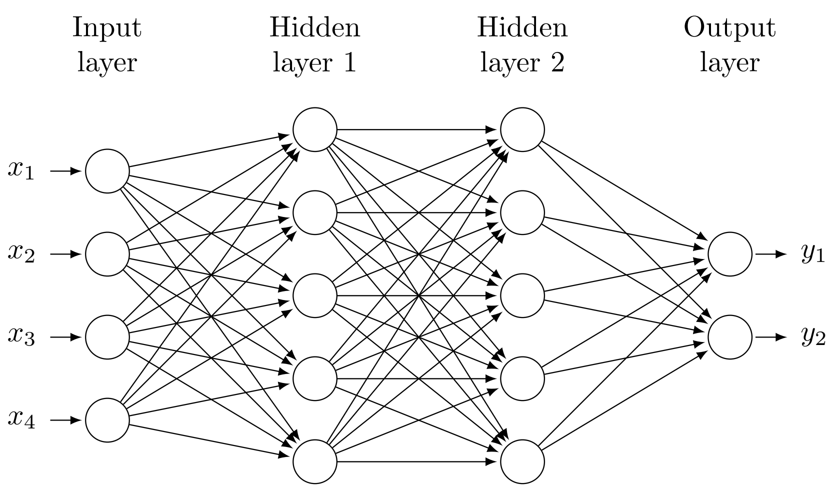


Figure 3.4: **A feed-forward neural network.** Network graph of a three-layer feed-forward neural network architecture with four input units and two output units. The hidden layers contain five hidden units each. Each circle corresponds to a unit that takes several inputs and produces a single output. The network structure is formed by connecting the different units.

input x_j is separately weighted and summed up to produce an output u_k for a given neuron k:

$$u_k = \sum_{j=0}^{D} w_{kj} x_j + b_k, \tag{3.33}$$

where w_{kj} denotes the weights and b_k the bias of neuron k. The output can then be passed through a differentiable, nonlinear activation function ϕ (see Section 3.2.4) yielding a nonlinear mapping:

$$y_k = \phi(u_k). \tag{3.34}$$

Finally, we denote the computation of a layer's output response consisting of M neurons as follows:

$$\mathbf{y} = \phi(\mathbf{W}\mathbf{x} + \mathbf{b}), \tag{3.35}$$

where $\mathbf{x} \in \mathbb{R}^D$ denotes each layer's input, $\mathbf{y} \in \mathbb{R}^M$ denotes each layer's output, and ϕ denotes the nonlinear activation function which is applied elementwise. The learnable weight matrix is denoted by $\mathbf{W} \in \mathbb{R}^{M \times D}$ and the bias vector is $\mathbf{b} \in \mathbb{R}^M$.

The layered structure of deep neural networks allows us to learn highly nonlinear input-output relationships. By stacking many layers on top of each other, each layer can extract a different representation of the input, making deep neural networks universal function approximators.

$$\underbrace{\begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{\text{Input}} * \underbrace{\begin{bmatrix} \boxed{\begin{matrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{matrix}} \end{bmatrix}}_{\text{Kernel}} = \underbrace{\begin{bmatrix} 1 & 4 & 3 & \boxed{4} & 1 \\ 1 & 2 & 4 & 3 & 3 \\ 1 & 2 & 3 & 4 & 1 \\ 1 & 3 & 3 & 1 & 1 \\ 3 & 3 & 1 & 1 & 0 \end{bmatrix}}_{\text{Output}}$$

Figure 3.5: **Illustration of a convolution operation.** Illustration of convolving a 3×3 kernel (or filter) over an input array with stride 1 and without zero-padding. The receptive field of the filter is highlighted in orange, the kernel weights in blue, and the resulting output activation in green. Each element of the output is the result of a dot product between the kernel and the input array.

3.2.6 *Convolutional Neural Networks*

Convolutional neural networks (CNNs, or ConvNets) are neural network architectures most commonly used for data with a grid-like topology since spatial correlations between features can be exploited. In this thesis, we mainly deal with image-based tasks, where the input can be represented as a multi-dimensional array (i.e. a tensor).

In Figure 3.4, we depict a feed-forward multi-layer perceptron, where each neuron is connected to all neurons from the previous layer (fully connected). In contrast, CNNs are typically composed of three types of layers: convolution, pooling, and fully connected layers. In the following, we introduce the basic concepts of convolution and pooling.

Convolution: Convolutional neural networks are based on convolution operations in which the learnable weights of each layer are defined as kernels (or filters). Figure 3.5 illustrates a convolution operation, where a kernel operates on the input array in a sliding window fashion. In general, the output of a convolutional layer is called *feature map* or *activation map* and is defined as follows:

$$O[m, n] = \sum_i \sum_j k[i, j] I[m - i, n - j], \tag{3.36}$$

where $O[m, n]$ denotes the value at location $[m, n]$ in the output feature map, I denotes the input, k denotes the kernel, and i and j define the kernel size.

The main motivation of a convolution operation is that the kernel weights are shared between pixel values such that it significantly reduces the number of learnable weights in contrast to fully connected layers. As a result, neural networks can learn generic features for computer vision tasks. During training, the kernel weights are automatically learned based on the provided dataset. However, convolution

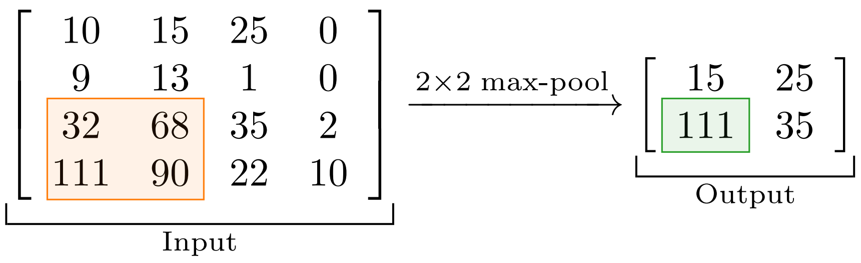


Figure 3.6: **Illustration of a max-pooling operation.** Illustration of a max-pooling operation with a receptive field of size 2×2. The receptive field is shown in orange and the output is shown in green.

layers have a couple of hyperparameters that must be predefined, such as the size of the kernels, number of kernels, padding, and stride. Padding defines the size that extends the input on both sides. The most commonly used padding strategy is zero-padding (adding zeros to all four sides of the input). Stride controls the number of pixels we slide the filter over the input. When the stride is 1, we move the filters one pixel at a time. When the stride is 2, we move the filters two pixels at a time yielding a reduced spatial dimension. Moreover, each convolutional layer can have multiple kernels (also called channels). Each kernel generates one channel in the output feature map. For an input array of size $W_1 \times H_1 \times D_1$, the output feature map size of the convolutional layer is then calculated as follows:

$$W_2 = (W_1 - F + 2P)/S + 1 \tag{3.37}$$
$$H_2 = (H_1 - F + 2P)/S + 1 \tag{3.38}$$
$$D_2 = K, \tag{3.39}$$

where K denotes the number of kernels, F is the kernel size, S defines stride, and P denotes the amount of padding on the borders of the input.

Pooling: Pooling is a technique to perform downsampling of the feature maps. The most popular form of pooling operation is max-pooling, which is illustrated in Figure 3.6. It extracts patches from the input and outputs the maximum value of each patch whilst disregarding all other values. In practice, max-pooling with a filter of size 2×2 and stride of 2 is commonly used to downsample the in-plane dimension of the input by a factor of 2.

Part II

OWN PUBLICATIONS

4 MODULAR VEHICLE CONTROL FOR TRANSFERRING SEMANTIC INFORMATION BETWEEN WEATHER CONDITIONS USING GANS

Authors: Patrick Wenzel*[1,2]
Qadeer Khan*[1,2]
Daniel Cremers[1,2]
Laura Leal-Taixé[1]

* Equal contribution
[1] Technical University of Munich
[2] Artisense

Conference: *Conference on Robot Learning (CoRL), 2018*

Abstract: Even though end-to-end supervised learning has shown promising results for sensorimotor control of self-driving cars, its performance is greatly affected by the weather conditions under which it was trained, showing poor generalization to unseen conditions. In this paper, we show how knowledge can be transferred using semantic maps to new weather conditions without the need to obtain new ground truth data. To this end, we propose to divide the task of vehicle control into two independent modules: a control module which is only trained on one weather condition for which labeled steering data is available, and a perception module which is used as an interface between new weather conditions and the fixed control module. To generate the semantic data needed to train the perception module, we propose to use a generative adversarial network (GAN)-based model to retrieve the semantic information for the new conditions in an unsupervised manner. We introduce a master-servant architecture, where the master model (semantic labels available) trains the servant model (semantic labels not available). We show that our proposed method trained with ground truth data for a single weather condition is capable of achieving similar results on the task of steering angle prediction as an end-to-end model trained with ground truth data of 15 different weather conditions.

Revised layout and minor adaptations. Accepted version of original publication [5] and detailed disclaimer are included in Appendix A.1.

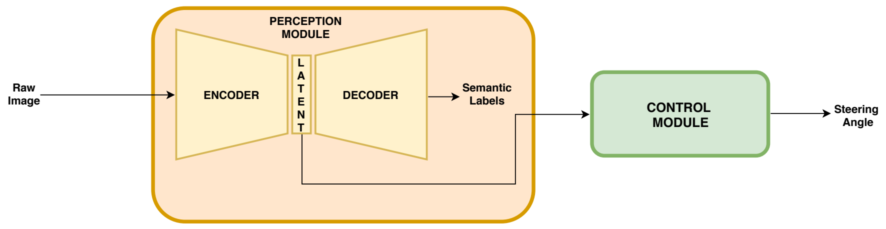


Figure 4.1: The perception module is trained as an encoder-decoder architecture, without any skip connections. The encoder sub-module first embeds the raw image into a lower dimensional latent vector. The decoder sub-module reconstructs the semantic scene from this latent vector. If the low dimensional latent vector contains all the necessary information to reconstruct the semantic scene to a reasonable degree of accuracy, then we directly feed it as an input to the control module instead of the semantic labels.

4.1 INTRODUCTION

One major goal of robotics and artificial intelligence research is to develop self-driving cars which can accurately perceive the environment and interact with the world. To develop an approach for addressing these problems, we have to deal with enormous challenges in perception, control, and localization. In general, the task of building an autonomous driving system can be divided into two parts: 1) path planning, and 2) vehicle control. Path planning provides a global solution for reaching a destination from a given starting position. It uses various information from different sensors such as GPS, IMU, and traffic conditions to infer the most optimized path. Meanwhile, vehicle control is meant to provide a local solution for predicting the immediate steering commands at the current instance in time. It utilizes information from sensors such as RGB cameras, lidar or radar. These sensors allow the self-driving car to sense and understand its current surroundings, such as the status of traffic lights or the presence of a pedestrian or another vehicle in front of the car.

In this paper, we focus our attention only on vehicle control to explain how transfer learning can be utilized to improve the robustness and stability of predicting steering commands even for unseen weather conditions for which no supervised data is available. For this, the task of vehicle control is segregated into perception and control. Figure 4.1 represents two modules, with each performing one of these tasks. The purpose of the perception module is to pre-process the raw input sensor data and extract useful features. In our approach, we use images captured by an RGB camera to extract semantic features of the scene. These extracted features are then fed to the control module which aims to produce the correct steering command for that particular sensor input.

Modular pipeline vs end-to-end learning. In an end-to-end training approach, both the perception and the control module would be trained together [20]. We propose to split the task into separate perception and control, so that each module is trained and optimized independently without affecting each other. The main advantage of the separate modules is that without retraining the control module, we can simply replace the perception module to work on different environmental conditions, whereas in an end-to-end learning system, supervised labels for the new domain would first be needed to be collected and then the control module would also need to be retrained on this additional data.

Our main contributions are the following:

- Ability to control the vehicle in unseen weather conditions without having the need to collect additional data for the steering commands and without requiring to retrain the control module. This is done by simply replacing the perception module additionally trained on the semantics of the new condition.
- We show how knowledge can be transferred from a weather condition for which semantic labels are available to other weather conditions for which no labels exist in an unsupervised manner using GANs.

4.2 RELATED WORK

Supervised learning for self-driving cars. The use of supervised learning methods to train driving policies for self-driving cars is a well-known and common approach. The first step towards using neural networks for the task of road following dates back to ALVINN [97]. This approach uses a very simple shallow network which maps images and a laser range finder as input and produces action predictions. Recently, NVIDIA [15] proposed to use deep convolutional neural networks trained end-to-end for a simple lane following task. This approach was successful in relatively simple real-world scenarios. One major drawback of end-to-end imitation learning is that it cannot generalize well across different domains for which no labeled training data is available. However, most end-to-end learning approaches [71, 121, 147] suffer from this problem.

Transfer learning. Generative adversarial networks provide a framework to tackle this generalization gap [61] by image generation techniques which can be used for domain adaptation. The authors of [144] proposed a network that can convert non-realistic virtual images into realistic ones with similar scene structures. Similarly, Hoffman et al. [50] proposed a novel discriminatively-trained adversarial model which can be used for domain adaptation in unseen environments. They show new state-of-the-art results across multiple adaptation tasks, including digit classification and semantic segmentation of road scenes.

Semantic segmentation. Visual understanding of complex environments is an enabling factor for self-driving cars. The authors of [26] provide a large-scale dataset with semantic abstractions of real-world urban scenes focusing on autonomous driving. By using semantic segmentation, it is possible to decompose the scene into pixel-wise labels we are particularly interested in. This especially helps self-driving cars to discover driveable areas of the scene. It is therefore possible to segment a scene into different classes (e.g. road and not road) [21].

Modular pipeline vs end-to-end learning. The authors of [31] trained both an end-to-end and a modular based model on one set of weather conditions and tested the model on a different set of weather conditions. Based on their experiments they concluded that the modular approach is more vulnerable to failures under complex weather conditions than the end-to-end approach.

Our method also uses a modular approach, but additionally introduces an image translation technique to overcome the generalization gap between the unseen weather conditions. This is done by only retraining the perception module without having to retrain the control module for each and every domain (i.e. weather condition). A useful consequence of this is that we do not have to recollect additional labeled data for the new conditions.

4.3 IMITATION LEARNING ON THE LATENT SEMANTIC VECTOR

Perception module. In this work, we use images captured by an RGB camera placed at the front of the car as inputs to the perception module. The perception module processes these images and produces an output map containing the semantics of the scene, which in turn can be used as an input to the control module. The CARLA [31] simulator yields semantic labels for 13 classes. The advantage of using semantic labels instead of raw RGB data is described below:

- Figure 4.2 shows how two weather conditions have different RGB inputs but the same semantic pixel labels. Hence, the control module does not separately need to learn to predict the correct steering commands for each and every weather condition.
- The semantic labels can precisely localize the pixels of important road landmarks such as traffic lights and signs. The status/information contained on these can then be read off to take appropriate planning and control decisions.
- A high proportion of the pixels have the same label as its neighbors. This redundancy can be utilized to reduce the dimensionality of the semantic scene. Hence, the number of parameters required to train the control module can then also be reduced.

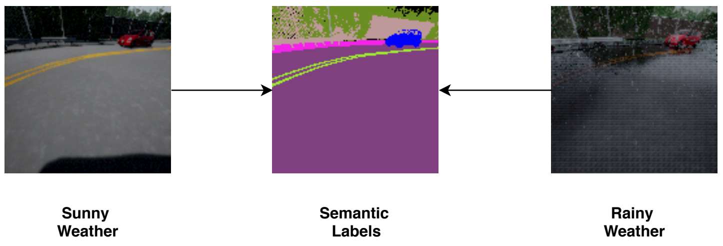


Figure 4.2: For the perception module we take in raw image data as obtained from the car's camera and output the semantic segmentation of the scene. Notice that irrespective of the weather condition the semantics of the scene remain the same. Since the perception module bears the burden of producing the correct semantic labels, the control module would be robust to changes in lighting, weather, and climate conditions.

The perception module, which is used to produce the semantic labels of a scene from the RGB camera is trained as an encoder-decoder architecture. The network architecture which is being used is a modified version of the one proposed by Larsen et al. [69]. The structure and the parameters of the model are shown in the supplementary material. The encoder first encodes the information contained in the input data to a lower dimensional latent vector. The decoder, then takes this latent vector and attempts to reconstruct the semantics of the scene. The output of the decoder is of the same size as the image but having 13 channels with each representing the probability of occurrence of one of the semantic labels. The model is trained by minimizing the weighted sum of the categorical cross-entropy of each pixel in the image. The categorical cross-entropy (negative log-likelihood) between predictions p and targets t is calculated as follows:

$$\mathcal{L}_i = -\sum_j t_{i,j} \log(p_{i,j}) w_j, \tag{4.1}$$

where i denotes the pixel and j denotes the class. The weight w_j of each semantic label is inversely proportional to its frequency of occurrence in the dataset.

Control module. Note that we do not use skip connections between the encoder and decoder of the perception module. Therefore, since the lower dimensional latent vector is capable of reconstructing the semantic labels of the scene, we can directly use this vector as input to the control module instead of the complete scene. Figure 4.1 depicts how the latent semantic embedding vector produced by the encoder of the perception module can be used as an input to the control module.

The control module aims to predict the correct steering angle, from the latent embedding fed as an input to the model. The data used for training the control module is collected in a supervised manner by recording images and their corresponding steering angles. The loss function attempts to minimize the mean squared error (MSE) between

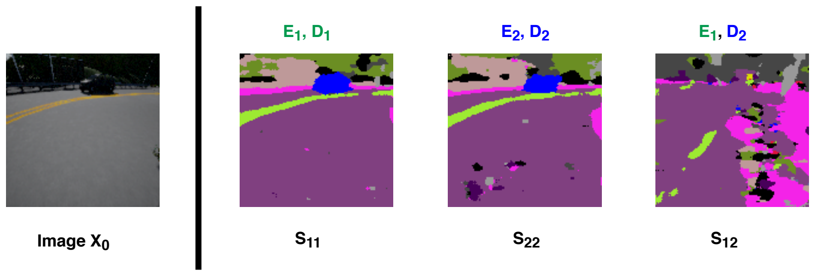


Figure 4.3: This figure shows the segmentation reconstructions S_{11} and S_{22} when image X_0 is passed through two segmentation models M_1 (with Encoder E_1, Decoder D_1) and M_2 (with Encoder E_2, Decoder D_2). Both models are trained independently on the same data. Note that the reconstructions reflect the true semantics of the scene reasonably well. S_{12} shows the reconstruction when the embedding from encoder E_1 is fed to through decoder of D_2. The ambiguity in S_{12} implies that for the same image the two models yield different semantic vectors.

the actual steering angle and the one predicted by the model across all the samples. The architecture of the control model is depicted in the supplementary material.

4.4 MASTER-SERVANT ARCHITECTURE FOR TRANSFER LEARNING

The control module does not perform well if tested in an environment which is completely different from the one on which the perception module was trained on. A naive and yet computational demanding solution could be to retrain the perception module under every other weather condition. However, this is not a viable solution for the following reasons:

- We would need semantic labels for every other weather condition. Obtaining semantic labels of a scene is a painstakingly slow process and prone to errors, since it requires human effort.
- Even if we have access to the semantic labels and retrain the perception module under the new environmental conditions, we would still have to also retrain the control module. This is due to the fact that the semantic latent vector produced by the new perception module might be different from the one produced by the old perception module, despite the same semantics of the scene. Figure 4.3 describes how for the same image, two independently trained segmentation models could yield different semantic vectors, despite being trained on the same data.

Proposed master-servant architecture. Suppose that the perception module P_0 and the control module C_0 are trained under a certain environmental condition. When tested on a very different weather condition P_0 may fail to produce the relevant semantic latent vector for the control module C_0 to take the correct steering decision. We would therefore like to replace P_0 with a different perception module P_1 such that it produces the correct latent vector to allow the same control module C_0 to execute the appropriate steering command even on this very different condition. For this, we propose a master-servant architecture model for training the perception module functioning on images from a domain for which no semantic labels are available. Figure 4.4 demonstrates the necessary steps of the master-servant architecture.

Suppose we have images (from domain X) and their corresponding semantic labels. With this, we can train a segmentation model using the encoder-decoder architecture described previously. We refer to the trained encoder of this model as the **master perception module** P_0. We would also like to obtain the correct semantic embedding of images (from domain Y) for new conditions for which no semantic labels are available. We refer to the perception module for which we would like to furnish the correct semantic embedding for images in domain Y as the **servant perception module** P_1. We use the master module, P_0, to train the servant module, P_1, in the steps described as follows:

1. An image X_0 is arbitrarily selected from domain X. X_0 is fed to through P_0 to obtain the semantic embedding of the scene denoted by z_0. Meanwhile, the generator G translates the image X_0 to generate an image Y_0 from domain Y, such that the semantics of the scene are preserved. If semantics are being preserved, then z_0 should be equal to z_1 (the semantic embedding obtained by feeding Y_0 through P_1).

2. Y_0 is fed through P_1 to get the predicted latent embedding z_1.

3. The mean squared error (MSE) between z_0 and z_1 is used as the loss function to update the weights of P_1 in order to minimize the difference between the two latent embeddings.

Some examples of the images produced by the generator G, segmentation reconstruction when z_0 (semantic embedding of the master) and z_1 (semantic embedding of the servant) is fed through the decoder of the master perception module P_0 are shown in the supplementary material.

Unsupervised transfer of semantics. We observe that with this master-servant architecture we are able to train the servant perception module for obtaining the correct semantic embeddings for images from domain Y for which semantic labels were never available. We can thus replace P_0 with P_1 which would also work on these unseen weather conditions without having to retrain the control module. Moreover, no additional human effort is required for the labeling of semantics.

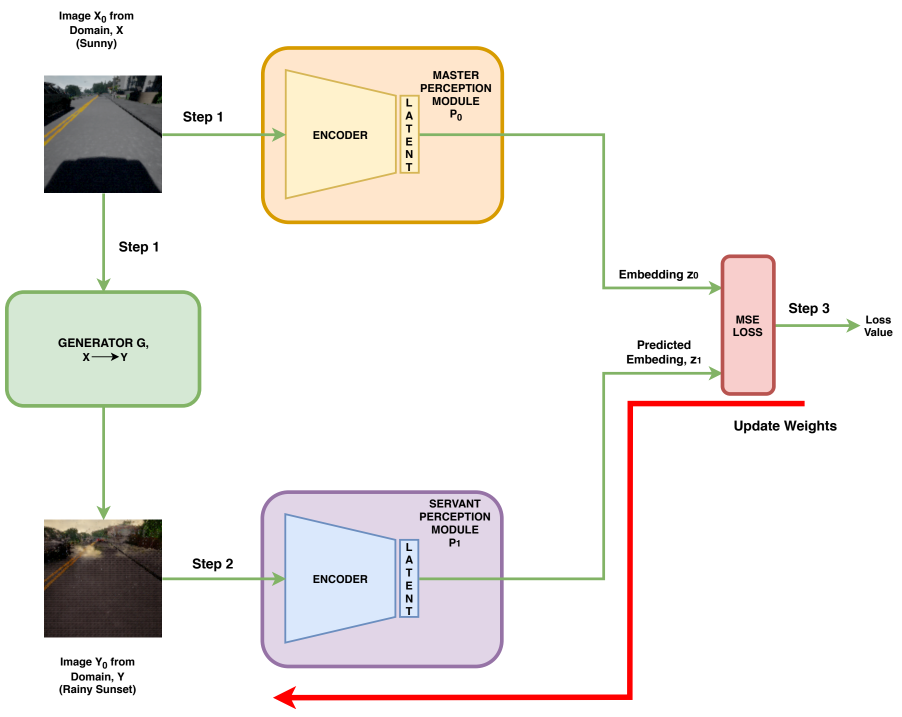


Image X_0 from Domain, X (Sunny)
Step 1
MASTER PERCEPTION MODULE P_0
ENCODER
LATENT
Step 1
Embedding z0
MSE LOSS
Step 3
Loss Value
GENERATOR G, X⟶Y
Predicted Embeding, z1
Update Weights
SERVANT PERCEPTION MODULE P1
Step 2
ENCODER
LATENT
Image Y_0 from Domain, Y (Rainy Sunset)

Figure 4.4: We propose a master-servant architecture to train a servant perception module P_1 for images which do not have semantic labels in an unsupervised manner. Images in domain X have semantic labels and are used to train the perception module P_0, which we refer to as the master perception module. P_0 is pre-trained using the complete encoder-decoder architecture. Images in domain Y do not have semantic labels. The process works as follows. **Step 1:** The generator G is used to convert an image X_0 from domain X to an image Y_0 in domain Y such that the semantic information is preserved. Meanwhile X_0 is also fed to the master perception module P_0 to get the latent embedding z_0. **Step 2:** The image Y_0 is fed to the servant perception module P_1 to get the predicted latent embedding z_1. **Step 3:** Since the semantic labels of X_0 and Y_0 are the same, their latent embeddings should also be the same. We use the mean squared error (MSE) to minimize this difference, wherein the embedding z_0 is used as the true label for training P_1. **Update Weights:** We back-propagate the MSE loss to update the weights of only P_1 so that its embedding matches with that of P_0. The green arrows indicate forward propagation and the red arrow shows back-propagation.

The most critical component which made the functioning of this approach possible is the generator G, which is able to translate images between two different domains, while preserving the semantics. The generator G is pre-trained using the CycleGAN [151] approach. Unlike other image-to-image translation methods such as pix2pix [54], an important feature of CycleGANs is the fact that this approach does not require paired data between two domains. Therefore, the task of collecting (if even possible) images with a one-to-one correspondence between two domains can be eliminated. The procedure for training the generator G using the CycleGAN approach is shown in the supplementary. The architecture used was taken from [151]. The supplementary material shows some examples of paired and unpaired data from two different domains produced by the CARLA simulator.

4.5 EXPERIMENTAL RESULTS

Experimental setup. For evaluating our method, we used the CARLA simulator. The CARLA simulator provides 15 different weather conditions (labeled from 0 to 14). We focus our attention on the car turning around corner scenarios since it is a more complicated maneuver to perform than lane following and it would thus give a better understanding of possible failure conditions. We train 5 different models to predict the steering angle whilst assuming that the car throttle is fixed. For a fair comparison, the approach is evaluated on multiple different turns and we do not consider the presence of pedestrians and cars in the ego vehicle's driving lane. The starting position of the agent is just before the curve and the duration of the turn is fixed to 120 frames since it covers the entire turning maneuver. The turn is considered successful if the car did not crash whilst executing the turn. Furthermore, in order to make a quantitative evaluation of the performance of each of the 5 models, new test data containing the images and the corresponding true steering commands for each of the 15 weather conditions was collected. Figure 4.5 shows a plot of the mean squared error (MSE) between the actual and the predicted steering commands by the 5 different models across all the weather conditions on samples of the test data. Meanwhile, Table 4.1 enumerates the percentage of turns each of the 5 models are successfully able to execute across all the 15 weather conditions.

The supplementary material contains the description and some samples of the 15 weather conditions along with video samples demonstrating the performance of the models on certain weather conditions. The dataset can be downloaded at: `https://git.io/fApfH`. The details of the 5 models are given below:

End-to-end, all weathers. An end-to-end model is trained on all weather conditions. Here we have assumed that we have access to the steering commands across all the conditions. As can be seen

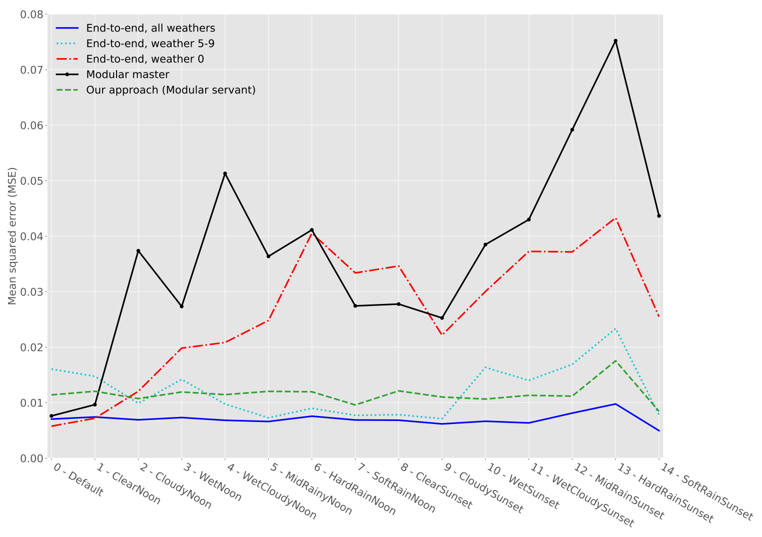


Figure 4.5: Plot of the mean squared error (MSE) between the actual and the predicted steering commands by 5 different models across the weather conditions 0 to 14. The **blue** line is the error plot for a model trained end-to-end, from images and corresponding steering commands for all the 15 weather conditions. The **cyan** error curve corresponds to the end-to-end model trained on images and steering commands for weathers 5-9. The **red** line is for the model trained end-to-end from images and corresponding steering commands for only the default weather condition 0. The **black** line represents the model referred to as the master whose perception and control modules are trained separately. The perception module is trained using the actual semantic labels available for the default weather condition, whereas the control model is trained from the actual steering commands of the same condition. The **green** curve is the model whose control model is the same as the one for the master, but the perception module is trained as a servant from the master perception module from images generated by the CycleGANs for weather conditions 2, 3, 4, 6, 8, 9, 10, 11, 12, and 13, in addition to the default condition 0.

from Figure 4.5, this model gives the lowest error particularly for weathers 1 to 14. Moreover, we observe in Table 4.1 that this model is able to successfully execute a high proportion of the turns across all the weather conditions, since it was trained on all of them. All subsequent models are trained with the steering commands available for a subset of the weather conditions and their performance is compared with this model.

End-to-end, weather 5-9. This model is trained end-to-end on weathers 5, 6, 7, 8, and 9 which were arbitrarily selected just to see how it would perform on unseen weather conditions. As shown in Figure 4.5 it has a relatively low error on these conditions and a higher error elsewhere. Furthermore, the plot shows that this end-to-end approach only seems to work well on the trained conditions for which we have labeled data. Moreover, as can be seen in Table 4.1, the model is capable of maneuvering well on the trained weather conditions and on those which are similar or have good visibility. However, on weather conditions 11-14 the model fails to execute the majority of the turns. This is mainly due to the fact that these weather conditions (11-14) are relatively disparate in terms of appearance and visibility as compared to the trained ones (5-9).

In practice, we do not have the steering commands available for all the possible or even a diverse subset of the weather conditions. Rather, the labeled data would correspond to only the condition of the day/period on which it was collected. Therefore, the 3 successive models that we now consider assume that the steering commands and the corresponding images/semantics are only available for the default weather condition (labeled as 0). From this, we evaluate how end-to-end training would compare to the proposed modular approach across all the remaining weather conditions for which no labeled data is available.

End-to-end, weather 0. This model is trained end-to-end from images and steering commands for the default weather condition. Figure 4.5, shows that this model outperforms all the other models only on weather condition 0 on which it was trained. For all other conditions, it gives high errors.

Modular master. This model is trained on the default weather condition (0) but the task is divided into 2 separate perception and control modules. The perception module P_0 is trained on the semantic labels. We refer to this perception module as the master, since it will later be used to train the servant module for retrieving the semantic information of the unseen weather conditions. The control module is in turn trained with imitation learning to predict the steering angle of the car from the latent embedding generated by the encoder of P_0. The fourth row of Table 4.1 depicts the percentage of turns the model was successfully able to maneuver for each of the 15 conditions. As observed in the table, the model is successful only on the default weather conditions (on which it was trained) and the sunny weather condition (which closely resembles the default condition). Similar to the previous model (trained end-to-end on the default condition), this model also fails on a large proportion when tested on weather conditions that are far off from the default condition in terms of visual appearance. From this, there seems to be no apparent advantage of using a modular approach over the end-to-end training when we have access to the labels for only one weather condition. Nevertheless, the master perception module P_0 obtained through this method will serve as a baseline for training a servant perception module that additionally works for unseen weather conditions. This approach is described in the following.

Our approach (Modular servant). We train one servant perception module to cater for weather conditions on which P_0 failed to perform.

Table 4.1: The table reports the percentage of successfully completed turns by the 5 models for each weather condition. Higher is better.

	Weather condition															
Model	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	average
End-to-end, all weathers	100	100	88	88	100	100	100	100	100	100	100	88	88	88	100	96
End-to-end, weather 5-9	88	88	100	88	100	100	100	100	100	100	100	50	50	25	88	85
End-to-end, weather 0	100	63	38	38	13	13	0	0	0	50	13	0	0	0	0	22
Modular master	100	100	88	50	50	63	50	50	50	63	75	50	0	0	50	56
Our approach (Modular servant)	100	100	100	100	88	100	100	100	100	100	100	88	100	63	100	96

We selected a subset of weather conditions (i.e. 2, 3, 4, 6, 8, 9, 10, 11, 12, and 13) to train the servant module. Using CycleGANs, separate generators were trained between each of these conditions and the default weather condition. The images produced by the CycleGAN generators for each of these conditions were fed as an input in equal proportion along with the default images to train only a single servant perception module P_1. Despite having no access to the steering commands and the semantic labels for weather conditions 1 to 14, Figure 4.5 shows that the error for this model across these 14 weather conditions is significantly lower than the previous 2 models which were also trained only from labels of weather condition 0. Moreover, we see from the last row of Table 4.1, that this model is successfully able to execute a good proportion of the turns for most of the weather conditions. Only on condition 13 (HardRainSunset), the model fails to perform well. The visibility under this condition is low and the images generated by the CycleGAN do not seem to preserve the semantics, hence resulting in the model to perform relatively poorly. Nevertheless, on all the other remaining weather conditions its performance is comparable to the first end-to-end model trained on steering labels for all the weather conditions.

4.6 conclusion

In this paper, we have shown that in order to generalize vehicle control across unseen weather conditions it is worthwhile to divide the task into separate perception and control modules. This separation eliminates the tedious task of recollecting labeled steering command data for each and every new environment the vehicle might come across. Moreover, retraining of the control module for new environments can be avoided by a simple replacement of the perception module. The initial perception module was trained from the semantic labels available only for one of the weather conditions. For environments for which semantic labels are missing, the proposed master-servant architecture can be deployed for transferring semantic knowledge from one domain to another (i.e. between different weather conditions) in an unsupervised manner using CycleGANs which do not require paired data. We believe that the presented approach to making driving policies more robust by training under different weather conditions will prove useful in future research.

5 TOWARDS GENERALIZING SENSORIMOTOR CONTROL ACROSS WEATHER CONDITIONS

Authors: Qadeer Khan*[1,2]
Patrick Wenzel*[1,2]
Daniel Cremers[1,2]
Laura Leal-Taixé[1]

* Equal contribution
[1] Technical University of Munich
[2] Artisense

Conference: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019*
DOI: 10.1109/IROS40897.2019.8968451

Abstract: The ability of deep learning models to generalize well across different scenarios depends primarily on the quality and quantity of annotated data. Labeling large amounts of data for all possible scenarios that a model may encounter would not be feasible; if even possible. We propose a framework to deal with limited labeled training data and demonstrate it on the application of vision-based vehicle control. We show how limited steering angle data available for only one condition can be transferred to multiple different weather scenarios. This is done by leveraging unlabeled images in a teacher-student learning paradigm complemented with an image-to-image translation network. The translation network transfers the images to a new domain, whereas the teacher provides soft supervised targets to train the student on this domain. Furthermore, we demonstrate how utilization of auxiliary networks can reduce the size of a model at inference time, without affecting the accuracy. The experiments show that our approach generalizes well across multiple different weather conditions using only ground truth labels from one domain.

Revised layout and minor adaptations. Accepted version of original publication [2] and detailed disclaimer are included in Appendix A.2.

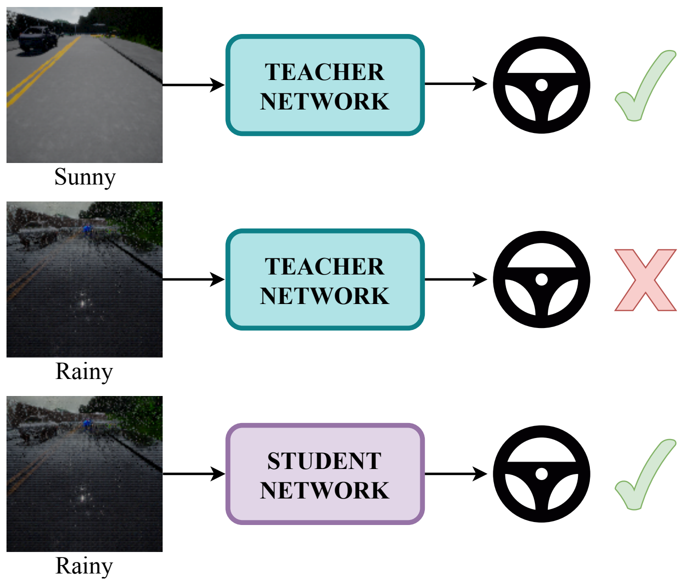


Figure 5.1: Teacher-student training for generalizing sensorimotor control across weather conditions. **Top:** The teacher network, trained on ground truth data collected on sunny weather is capable of predicting the correct steering angle when tested on this condition. **Middle:** However, the teacher fails to predict the correct steering when tested on an input image from a different domain (rainy weather). **Bottom:** With our proposed framework, the student network trained with supervised information from the teacher network is capable of predicting the correct steering for the rainy weather. This is done without any additional ground truth labels or semantic information.

5.1 INTRODUCTION

The ubiquity of a tremendous amount of processing power in contemporary computing units has proliferated the usage of deep learning-based approaches in control applications. In particular, supervised deep learning methods have made great strides in sensorimotor control, whether it be for autonomous driving [15], robot perception [62], or manipulation tasks [74, 89, 148]. However, the performance of such models is heavily dependent on the availability of ground truth labels. To have the best generalization capability, one should annotate data for all possible scenarios. Nonetheless, obtaining labels of high quality is a tedious, time consuming, and error-prone process.

We propose to instead utilize the information available for one domain and transfer it to a different one without human supervision as shown in Figure 5.1. This is particularly helpful for many robotic applications wherein a robotic system trained in one environment should generalize across different environments without human intervention. For example in simultaneous localization and mapping (SLAM), it is very important that the algorithm is robust to different lighting conditions [95]. In the context of autonomous driving, transferring knowledge from simulation to the real world or between different weather conditions is of high relevance. Recently, [86, 5, 144] have attempted to tackle these problems by dividing the task of vehicle control into different modules, where each module specialized in extracting features from a particular domain. In these works, semantic labels are used as an intermediate representation for transferring knowledge between different domains. However, obtaining these semantic labels requires human effort which is time-consuming, expensive, and error-prone [5]. In this work, we instead propose to use a teacher-student learning-based approach to generalize sensorimotor control across weather conditions without the need for extra annotations, e.g. semantic segmentation labels.

To this end, we make the following contributions:

- We demonstrate how knowledge of ground truth data for steering angles can be transferred from one weather scenario to multiple different weather conditions. This is achieved without the additional requirement of having semantic labels. We make use of an image-to-image translation network to transfer the images between different domains while preserving information necessary for taking a driving decision.
- We show how the proposed method can also utilize images without ground truth steering commands to train the models using a teacher-student framework. The teacher provides relevant supervised information regarding the unlabeled images to train the features of the student. Hence, we can eliminate the need for an expert driver for data collection across diverse conditions.
- If the sample data with ground truth labels is limited, then the teacher and student models may tend to overfit. To overcome this, we propose using weighted auxiliary networks connected to the intermediate layers of these models. During inference, the model size can be reduced by eliminating auxiliary layers with low weights without reducing accuracy.

In the following sections, we first review related work. We then present the details of our method, followed by an analysis of our model's performance. Finally, we discuss various parts of our model.

5.2 RELATED WORK

Vision-based autonomous driving approaches have been studied extensively in an academic and industrial setting [55]. A plenty of real world [26, 41, 139] as well as synthetic [38, 102, 103, 107] datasets for autonomous driving research have become available. In recent years, neural network approaches have significantly advanced the state-of-the-art in computer vision tasks. Especially, end-to-end learning for sensorimotor control has recently gained a lot of interest in the vision and robotics community. In this context, different approaches to autonomous driving are studied: modular pipelines [133], imitation learning [97], conditional imitation learning [25], and direct perception [20].

Embodied agent evaluation. Most available datasets [26, 41] cannot be used for evaluating online driving performance due to their static nature. The evaluation of driving models on realistic data is challenging and often not feasible. Therefore, a lot of interest has emerged in building photo-realistic simulators [31, 85, 119] to analyze those models. However, despite having access to simulation engines, there is currently no universally accepted benchmark to evaluate vision-based control agents. Therefore, our experimental setup is a step towards a field where it is still not quite established how to evaluate and measure the performance of the models [10, 24].

Unpaired image-to-image translation networks. Unsupervised image-to-image translation techniques are rapidly making progress in generating high-fidelity images across various domains [53, 75, 76, 151]. Our framework is agnostic to any particular method. Hence, continual improvements in these networks can be easily integrated into our framework by replacing a previous network.

Transfer learning via semantic modularity. Several works used semantic labels of the scene as an intermediate representation for transferring knowledge between domains. In the context of autonomous driving, the authors of [86] proposed to map the driving policy utilizing semantic segmentation to a local trajectory plan to be able to transfer between simulation and real-world data. Furthermore, for making a reinforcement model trained in a virtual environment workable in the real world, the authors of [144] utilize the intermediate semantic representation as well to translate virtual to real images. However, there is still little work on generalizing driving models across weathers. The work by [5] showed how to transfer knowledge between different weather conditions using a semantic map of the scene. In contrast, in this paper, we demonstrate the possibility of transferring the knowledge between weathers even without semantic labels.

Knowledge distillation. Originally, knowledge distillation [48] was used for network compression (student network is smaller than the teacher while maintaining the accuracy). However, the authors of [140] focus on extracting knowledge from a trained (teacher) network and guide another (student) network in an individual training process. Furthermore, [120] used a slightly modified version of knowledge distillation for the task of pedestrian detection. In this work, we use a teacher-student architecture, but rather to leverage unlabeled data for sensorimotor control.

5.3 SENSORIMOTOR CONTROL ACROSS WEATHERS

In this section, we introduce a computational framework for transferring knowledge of ground truth labels from one weather condition to multiple different scenarios without any semantic labels and additional human labeling effort. Figure 5.2 gives a high-level overview of the framework.

5.3.1 *Teacher End-to-End Training*

In this step, the teacher model is trained end-to-end in a supervised manner to predict the steering command of the vehicle from the raw RGB images generated by the camera placed at the front of the ego vehicle. The training data is collected by an expert driver only once for that particular weather scenario. We refer to the images recorded under the weather condition under which this data was collected as belonging to domain D_0. Note that the teacher model is itself divided into a Feature Extraction Module (FEM), F_0 and a control module, C_0. The raw image (belonging to D_0) is passed through F_0 to retrieve a lower-dimensional feature representation. This feature representation is in turn fed to the C_0 which predicts the steering angle. A depiction of the model is shown in Figure 5.3. The FEM, F_0 is a sequential combination of 4 units where each unit comprises a convolutional, pooling, and activation layer. The output of unit 4 is flattened to a size of 800, which is in turn fed as an input to the module, C_0. The control module, C_0 is a series of fully connected layers and outputs the steering command.
Auxiliary network. It might be the case that the amount of images with labels is limited or the model is too large for the task at hand. Hence, the model may tend to overfit. Therefore, during training, to mitigate the effect of overfitting, F_0 additionally uses auxiliary networks connected to its intermediate layers [130]. Each of the auxiliary networks has a control module, C_0 with shared weights. The projection layers, P_1, P_2 and P_3 project the feature maps of the intermediate layers to the dimension of C_0, i.e. 800. The overall output of the teacher model is the weighted sum of the outputs of the auxiliary networks. The loss is also described by a weighted combination of the individual losses of the 4 auxiliary networks. The loss for each of the control modules is the mean squared error (MSE) between the ground truth label provided by the expert and that predicted by C_0. The overall loss is a weighted sum of the losses from each of the 4 control modules.

$$\mathcal{L} = \sum_{i=1}^{4} \alpha_i \mathcal{L}_i, \qquad \text{s.t.} \sum_{i=1}^{4} \alpha_i = 1, \tag{5.1}$$

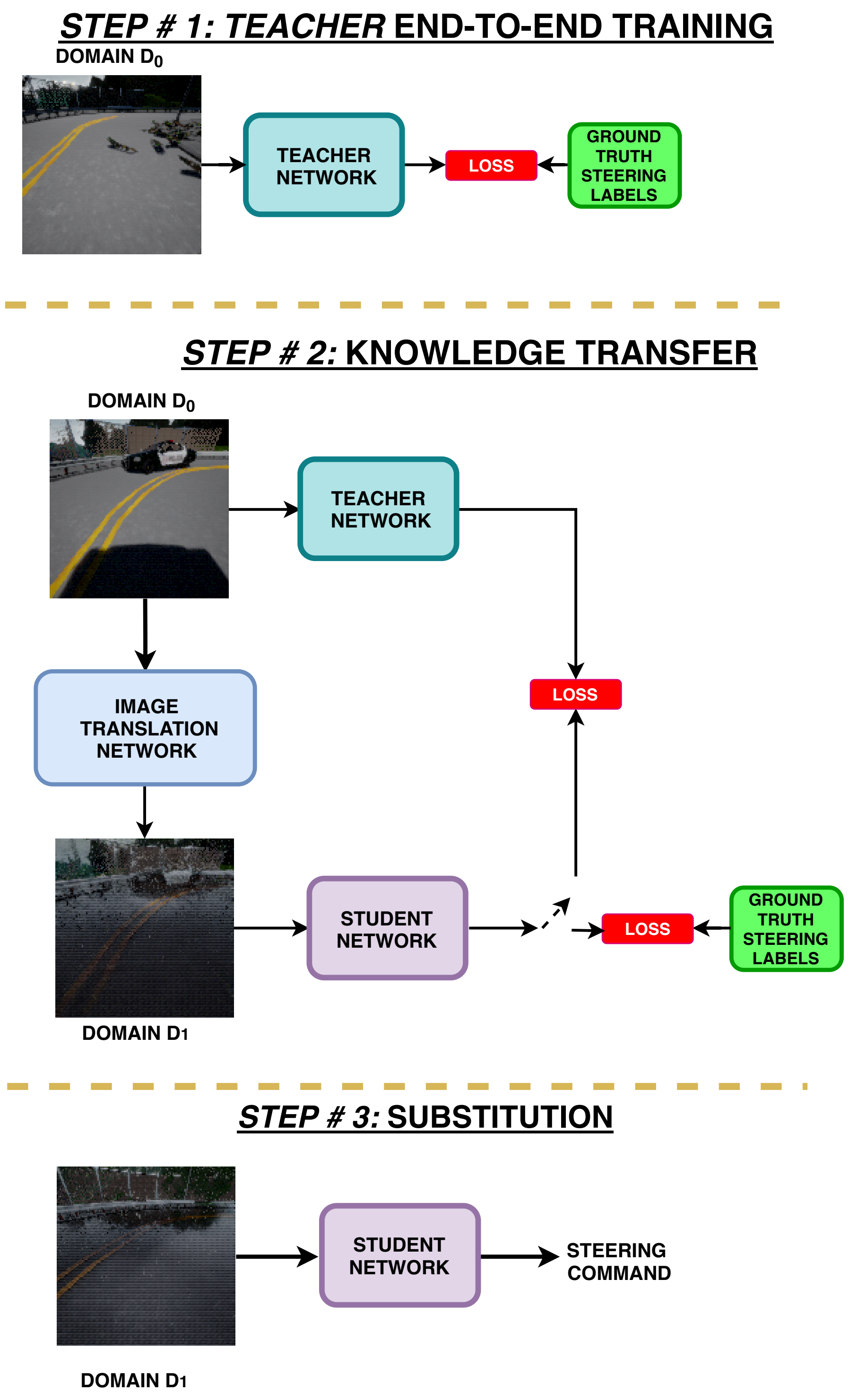


Figure 5.2: This figure gives a high level overview of the 3 steps for transferring knowledge between two domains D_0 and D_1 for the purpose of sensorimotor control. Ground truth steering data for only a limited number of images from domain D_0 is available. Details of the framework are provided in Section 5.3.

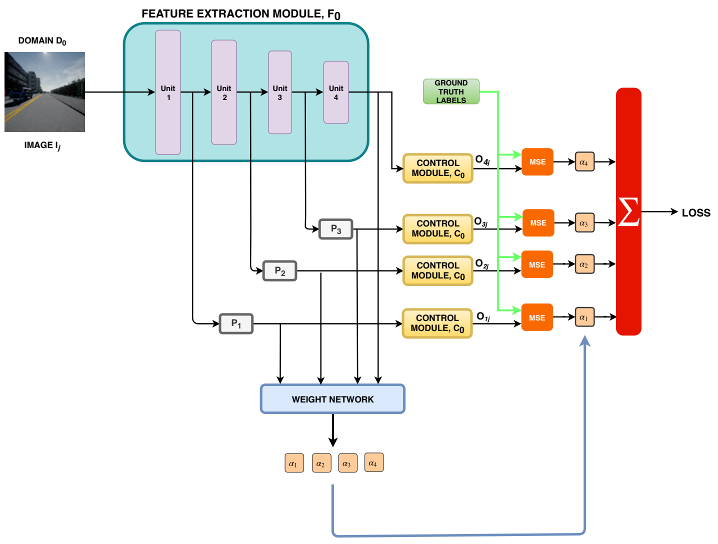


Figure 5.3: The figure depicts the general architecture of the model comprised of the FEM and the auxiliary control modules.

where α_i, and $\mathcal{L}_i$ are the weighting and the error for the auxiliary network, obtained from the intermediate unit i of the FEM F_0. The error functions are calculated as follows:

$$\mathcal{L}_i = \frac{1}{N}\sum_{j=1}^{N}(y_j - O_{ij})^2, \qquad (5.2)$$

where y_j is the ground truth steering angle obtained from the expert driver for a sample j and N denotes the number of total samples. O_{ij} is the output of the control module corresponding to the ith auxiliary network for the jth sample.

The weights α_i are themselves learned by a separate weight network. The auxiliary network that has the greatest contribution toward the overall output would also have the highest relative weight. This is important in case of limited data, wherein not all layers may be essential to train the model. In such a case the weights of the shallower auxiliary networks would be higher in comparison to the deeper auxiliary networks. Hence, a significant contribution towards the overall prediction would come from these shallow layers, thereby making the deeper layers effectively dormant. An extreme case would be when the labeled data is so small that even the first layer is enough to give a correct model prediction. In such a case, only $\alpha_1 = 1$ and all other $\alpha_i = 0$, for $i = 2, 3, 4$.

5.3.2 *Knowledge Transfer*

As described in step 2 of Figure 5.2, knowledge of ground truth labels from domain D_0 is transferred to domain D_1 using a teacher-student architecture. The output of the auxiliary networks acts as the teacher to provide supervised information to train the student.

We use the FEM, F_0 and control module, C_0 (combined, referred to as teacher) trained on images belonging to domain D_0, for which ground-truth steering labels are available, to transfer knowledge to a different combination of FEM, F_1 and control module, C_1 (referred to as student) for domain D_1, for which we have access to only unlabeled images. The subsequent procedure is detailed in the following steps:

1. Image I_0 belonging to domain D_0 is passed through an image-translation-network to generate image I_1 belonging to domain D_1 in a manner that only the semantic information is preserved but the weather condition is modified. [53, 75, 151] describe methods for training a translation network in an unsupervised manner using generative adversarial networks (GANs). We use [151] for our experiments. A positive implication of using these networks is that they preserve the semantics of the scene and hence the steering angle label would also be the same.

2. **Hard loss:** If I_0 happens to have a ground truth (*hard*) label then the weights of the student network are updated with these labels and the loss is referred to as the *hard loss*. **Soft loss:** Otherwise, a forward pass can also be done by passing I_0 through the teacher. Meanwhile, the corresponding image I_1 is passed through the student network. The output of the teacher can then be used as a soft target for updating the weights of the student via the soft loss. The overall loss is the weighted average of the soft and hard losses. The weights indicate the relative importance given to the soft targets in relation to the ground truth labels.

Note that the student network can be fed not only images from domain D_1 but rather multiple domains including domain D_0. Hence, the student network would not only be capable of predicting the steering for multiple domains but would act as a regularizer for better generalization (See P1 in Section 5.5).

5.3.3 *Substitution*

This refers to step 3 described in Figure 5.2. At inference time, the teacher network can be substituted with the student network to predict the correct steering command on images from all domains which the student encountered during training.

5.4 EXPERIMENTS

In this section, we evaluate our approach on the CARLA simulator [31] version 0.8.2. It provides a total of 15 different weather conditions (labeled from 0 to 14) for two towns, *Town1* and *Town2*, respectively.

5.4.1 *Evaluation Metrics*

Finding appropriate evaluation metrics is rather challenging for navigation and driving tasks. There is no unique way to quantify these tasks. The authors of [10] discuss different problem statements for embodied navigation and present based on these discussions evaluation metrics for some standard scenarios. In [24], a more extensive study on evaluation metrics for vision-based driving models is carried out. In particular, they analyzed the difference between online and offline evaluation metrics for driving tasks. The preliminary results showed that driving models can have similar mean squared error (MSE) but drastically different driving performances. As a result of this, it is not straightforward to trivially link offline to online performance due to a low correlation between them. Nevertheless, the authors of [24] found that among offline metrics not requiring additional parameters, the mean absolute error between the driving commands and the predicted ones yields the highest correlation with online driving performance.

In addition to using this offline metric, we evaluate the online performance of the models when executing multiple and diverse turnings around corners, since it is a much more challenging task in comparison with simply moving in a straight line. The online performance is tested on the CARLA simulator across all the 15 weather conditions. For each weather condition, we evaluate the models for multiple different turns. In all experiments, the starting position of the vehicle is just before the curve. The duration of the turn is fixed to 120 frames because it covers the entire curvature of the turn. We report the percentage of time the car remains within the driving lane as a measure of success.

5.4.2 *Dataset*

For collecting ground truth training data, we navigate through the city using the autopilot mode. To demonstrate the superiority of our method, we collect a limited sample size of 6500 images for weather condition 0 of which only 3200 are labeled with ground truth steering commands. Using our proposed method we aim to transfer knowledge to the remaining 14 weather scenarios. Also, note that none of the 6500 images have any semantic labels.

The 3200 sample images with ground truth data are only available for *Town2*, whereas all the offline and online evaluations are performed on *Town1*. To focus the attention on the effectiveness of our approach and preserve compatibility with prior work [15, 24, 139], the models are trained to predict the steering angle of the car while keeping the throttle fixed. The steering angles in CARLA are normalized to values between -1 and 1. The corresponding degrees for these normalized values depend on the vehicle being used. The default vehicle which we use for our experiments has a maximum steering angle of 70°.

5.4.3 *Models*

The offline and online performance of the models described in this section are given in Figure 5.4 and Table 5.1, respectively. Figure 5.4 shows the plot of the mean absolute error between the actual steering command and that predicted by all of the models. Table 5.1 contains the percentage for which the ego-vehicle remains within the driving lane while making turning maneuvers executed by the models across the 15 weather scenarios.

Oracle: Steering labels for all weathers. Here we have assumed that we have access to the ground truth steering commands across all the 15 different weather conditions for *Town1*. Since we are also evaluating the models on *Town1* across all the weather conditions, we find in both the offline and online evaluation metrics that this model achieves the highest accuracy and hence it could serve as an upper bound for evaluating the other models along with our approach.

Model [5]: Steering and semantic labels for weather 0. Here we adopt the approach of [5], wherein the semantic labels of the images are additionally available for the 3200 labeled samples on weather 0. This additional information is used to first train what we refer to as the feature extraction module (FEM) in a supervised manner. The FEM module, in this case, is trained as an encoder-decoder architecture. The encoder encodes the input image into a lower-dimensional latent vector, while the decoder reconstructs the semantic map of the image from the latent vector. The latent vector is then used to train the control module from the ground truth steering labels. The FEM and control modules are hence trained independently and without any auxiliary networks. This FEM trained on the semantics of weather 0 is used as a teacher to train the student which is capable of producing the semantics of all the other 14 weather conditions. The authors of [5] used the method of [151] and provide 10 separate networks for translating from weather 0 to weathers 2, 3, 4, 6, 8, 9, 10, 11, 12, and 13, respectively. The translated images for each of the 10 weather conditions along with weather 0 are fed in equal proportion to train the student. We would particularly like to evaluate our method which does not have access to any semantic labels against this model. In addition to this, we also evaluate the performance of this method on the model provided by the paper, which was trained with more than 30 000 samples from both *Town1* and *Town2*. The performance of this model on *Town1* is far superior since it was trained on much greater data and also had access to ground truth data from *Town1*.

Teacher: Steering angles for weather 0. This model is trained using only the available labeled data for weather 0 in an end-to-end manner. This model has a poor performance for the unseen weather conditions, particularly for conditions 3-14, which are considerably different in visual appearance compared to weather 0. Nevertheless,

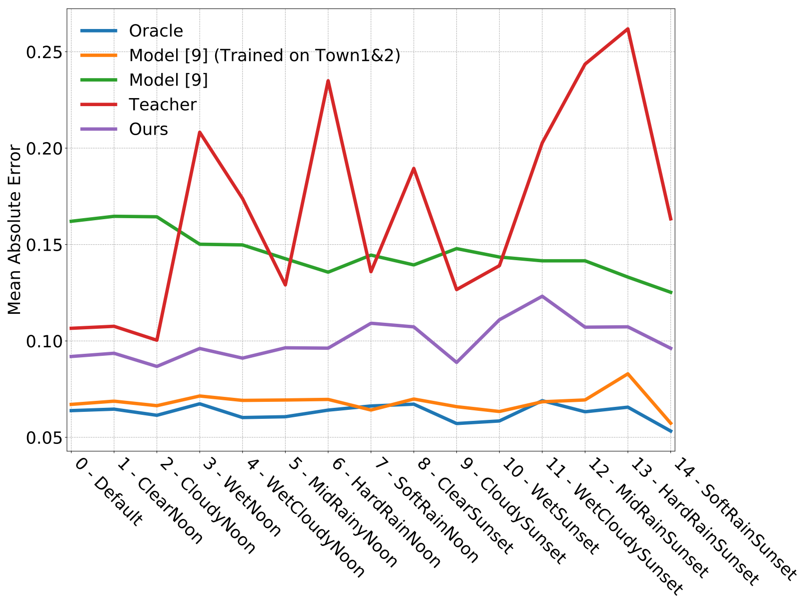


Figure 5.4: This plot shows the mean absolute error between the actual steering angle and that predicted by the 5 different models (see Section 5.4.3) on data collected across the 15 different weather conditions on *Town1*. Lower is better.

despite the poor performance this model can be used as a teacher to train the student for predicting the correct steering angles for weather conditions 1-14 for which no ground truth data exists. This approach is described in the next model. Also, note that the unlabeled data remains unutilized here.

Ours: Steering angles for weather 0. This model is trained using the method described in Section 5.3, wherein knowledge is transferred from the teacher network trained on images and ground truth steering commands from weather 0 to the student network which is capable of handling images from all weathers 0-14. For a fair comparison against the model trained with semantic labels (Model [5], described earlier) we use the same data and generative models to translate even the unlabeled images to weathers 2, 3, 4, 6, 8, 9, 10, 11, 12, and 13, respectively. These generated images can then be fed to the student model for predicting the correct steering angles for all the 15 weather conditions.

5.5 Discussion

In this section, we discuss some critical insights on the experimental observations we obtained while evaluating the models. Here are some points we found worthwhile to provide some commentary based on the results provided in Figure 5.4 and Table 5.1.

P1 - Better regularization: It is interesting to observe that the teacher model, trained only on the available 3200 labeled samples from *Town2*

Table 5.1: This table shows the percentage for which the ego-vehicle remains within the driving lane while executing a turn for the models across the 15 different weather scenarios on *Town1*. Higher is better.

		Weather Conditions															
Method	Trained on	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	overall
Oracle	*Town1*	99.79	99.90	100	97.40	98.96	99.27	98.13	98.85	98.27	99.90	99.27	96.35	93.85	93.96	96.35	98.02
Model [5]	*Town1&2*	99.06	93.44	98.85	98.75	97.92	98.23	97.60	96.56	91.15	96.04	97.29	95.00	94.69	82.08	95.41	95.47
Model [5]	*Town2*	68.33	67.71	50.00	71.77	67.40	64.38	63.85	63.65	61.88	71.35	51.35	67.50	58.33	61.67	66.98	63.74
Teacher	*Town2*	92.19	92.40	82.12	44.38	51.77	73.65	32.50	61.56	49.48	80.10	60.63	48.54	35.20	34.27	50.52	59.29
Ours	*Town2*	93.96	95.21	81.25	99.90	100	94.17	90.42	79.69	77.19	86.77	84.58	65.63	68.54	58.44	80.73	83.77
Ours (Auxiliary network 1)	*Town2*	93.96	93.44	80.73	92.40	100	99.69	90.42	80.10	77.19	87.50	91.98	67.40	66.25	57.29	81.15	83.97

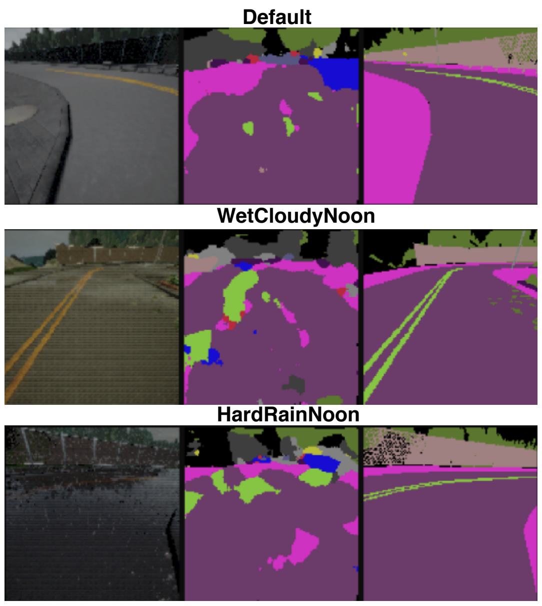


Figure 5.5: This plot shows three sample images (column 1) with the corresponding semantic segmentation output by the model (column 2) for 3 different weathers. The segmentation produced by the model does not reflect the actual semantic characteristics of the scene (column 3).

on weather 0 has a worse offline performance for *Town1* on weather 0 in comparison to our method. This seems to imply that our approach which has been trained on multiple kinds of weather has better generalization capabilities and can even outperform its teacher when evaluated in a different town. Hence, an additional positive consequence of training the student with generated images from multiple diverse domains is that it acts as a regularizer tending to prevent overfitting to one specific domain.

P2 - Semantic inconsistency: Note that Model [5] which in addition to having the same data and labels as our approach has also access to ground truth semantic labels. Yet, its performance is significantly poor. Upon investigation, we found that due to the limited number of semantic labels, the FEM trained as an encoder-decoder architecture seemed to be overfitting to the available data. Hence, when tested on unseen environments, the semantic segmentation output of the module breaks. The latent vector representing these broken semantics is then fed to the control module, which is incapable of predicting the correct steering command. Figure 5.5 shows some sample images with the corresponding semantic segmentation outputs which are considerably different from the true semantics of the scene.

P3 - Modular training constraints: Furthermore, the modular approach of Model [5] wherein the FEM and control module are trained independently as opposed to an end-to-end model served to be a bottleneck in being able to learn the features universally. Also, an assumption to train the control module well is that the FEM would work perfectly well, which is not the case. Hence, the overall error of the modular pipeline would be an accumulation of the errors of the independent FEMs and control modules. We found that if we also shift the training of our approach to a modular one then performance deteriorates. This can be done in our approach by updating only the weights of the FEM of the student from the output features of the FEM of the teacher.

P4 - Auxiliary weights: To prevent overfitting of the models, trained on limited data we used a weighted sum of the outputs of the auxiliary layers. The weights themselves were learned as part of the training. Once training of our student model was complete, we found that more than 97 % of the weight was held by the first auxiliary network. This seemed to imply that only the first unit of the FEM is enough for predicting the steering command. Hence the remaining unit layers are not providing any additional information for the model. So we evaluated our model based on the output of the first auxiliary network rather than on the weighted sum of the 4 auxiliary networks. The online evaluation of this approach is given in Table 5.1 against the row labeled *Ours (Auxiliary network 1)*. It is interesting to note that this approach is comparable in its performance with the original one. Therefore, at test time we can prune the network to a smaller size by making predictions only based on the first auxiliary network and removing the remaining 3 auxiliary networks. This would result in less computation and faster inference.

P5 - Online vs offline evaluation: Figure 5.6 shows an offline evaluation of the two variations of our method described in the previous point across the 15 weather conditions. Note that apart from weather 0, 1, and 2, the two curves are indistinguishable from one another. However, the online evaluation results do not correspond with this observation. For weathers 3, 5, 7, and 9-14 the online performance is different despite having the same offline metric. This confirms the intuition presented in [10] and the problems associated with evaluating embodied agents in offline scenarios. The topic of finding a correlation between offline evaluation metrics and online performance has therefore recently started to receive positive traction. It is therefore important to come up with a universal metric for evaluating various algorithms across the same benchmark. Due to the non-existence of such benchmarks, we created our own for the evaluation of the different approaches.

P6 - Activation maps: To understand the behavior of the model, which also works with only the first auxiliary network, we took the sum of the activation maps of the first unit of the FEM of the student and displayed it as a heatmap as shown in Figure 5.7 for a sample of 2 images. We see that the activation maps are most prominent in regions where there are lane markings, sidewalks, cars, or barriers.

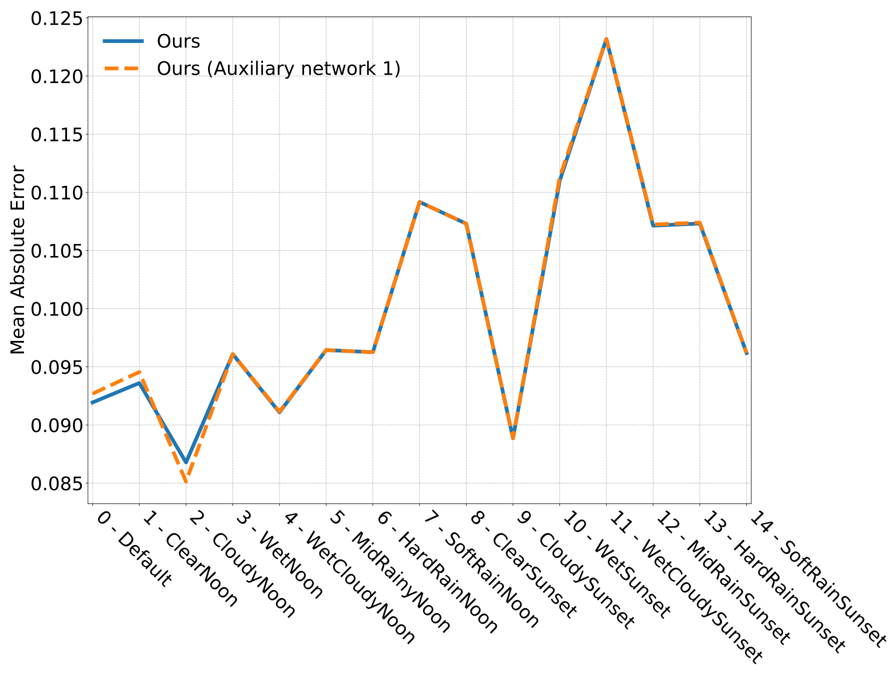


Figure 5.6: This plot shows the mean absolute error between the ground truth steering label and that predicted by the two models. The **blue** curve is the weighted sum of all the 4 auxiliary networks of our model. The **orange** line depicts the output of only the first auxiliary network of our model.

Knowing these cues seems to be enough for the network to take an appropriate driving decision in most of the cases. Therefore, the higher-level features determined by the preliminary layers of the model are already enough to detect these objects of interest.

5.6 CONCLUSION

In this work, we showed how a teacher-student learning-based approach can leverage limited labeled data for transferring knowledge between multiple different domains. Our approach, specifically designed to work for sensorimotor control tasks, learns to accurately predict the steering angle under a wide range of conditions. Experimental results showed the effectiveness of the proposed method, even without having access to semantic labels as an intermediate representation between weather conditions. This framework may be extendable to other application areas for which a certain domain has ground truth data and shares a common characteristic with other domains for which no labels are available.

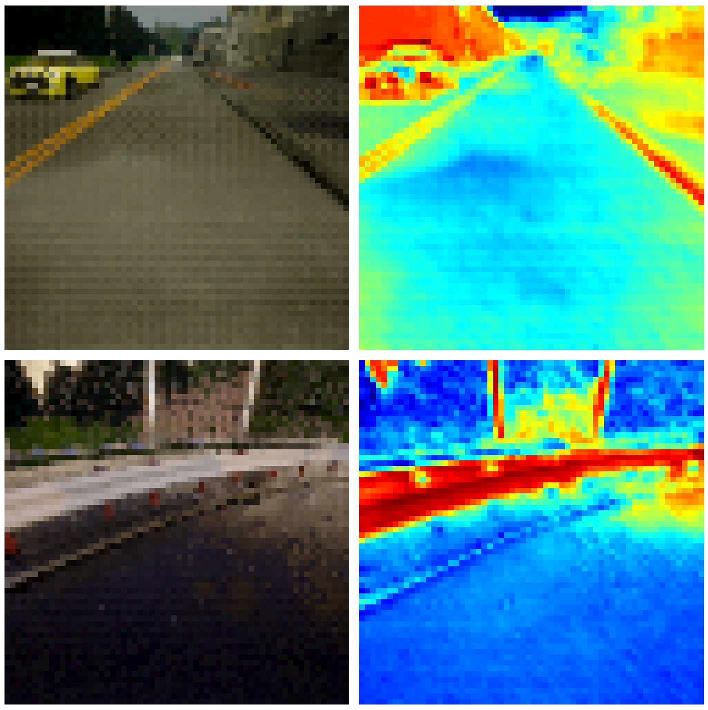


Figure 5.7: This figure shows the sum of the activation maps displayed as a heatmap of the first unit of the FEM of the student model for a sample taken from 2 different weather conditions. The activation maps are more prominent in regions where there are lane markings, sidewalks boundaries, other vehicles, or barriers.

6 GN-NET: THE GAUSS-NEWTON LOSS FOR MULTI-WEATHER RELOCALIZATION

Authors: Lukas von Stumberg*[1,2]
Patrick Wenzel*[1,2]
Qadeer Khan[1,2]
Daniel Cremers[1,2]

* Equal contribution
[1] Technical University of Munich
[2] Artisense

Journal: *IEEE Robotics and Automation Letters (RA-L), 2020*
DOI: 10.1109/LRA.2020.2965031

Abstract: Direct SLAM methods have shown exceptional performance on odometry tasks. However, they are susceptible to dynamic lighting and weather changes while also suffering from a bad initialization on large baselines. To overcome this, we propose GN-Net: a network optimized with the novel Gauss-Newton loss for training weather invariant deep features, tailored for direct image alignment. Our network can be trained with pixel correspondences between images taken from different sequences. Experiments on both simulated and real-world datasets demonstrate that our approach is more robust against bad initialization, variations in day-time, and weather changes thereby outperforming state-of-the-art direct and indirect methods. Furthermore, we release an evaluation benchmark for relocalization tracking against different types of weather. Our benchmark is available at https://vision.in.tum.de/gn-net.

Revised layout and minor adaptations. Accepted version of original publication [3] and detailed disclaimer are included in Appendix A.3.

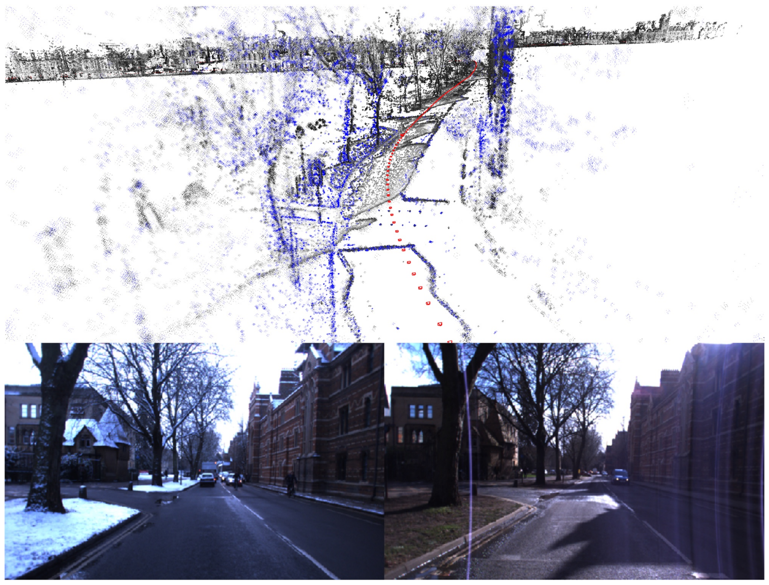


Figure 6.1: We relocalize a snowy sequence from the Oxford RobotCar dataset in a pre-built map created using a sunny weather condition. The points from the prior map (gray) align well with the new points from the current run (blue), indicating that the relocalization is indeed accurate.

6.1 INTRODUCTION

In recent years, very powerful visual SLAM algorithms have been proposed [66, 87]. In particular, direct visual SLAM methods have shown great performance, outperforming indirect methods on most benchmarks [8, 34, 35]. They directly leverage the brightness data of the sensor to estimate localization and 3D maps rather than extracting a heuristically selected sparse subset of feature points. As a result, they exhibit a boost in precision and robustness. Nevertheless, compared to indirect methods, direct methods suffer from two major drawbacks:

1. Direct methods need a good initialization, making them less robust for large baseline tracking or cameras with a low frame rate.

2. Direct methods cannot handle changing lighting/weather conditions. In such situations, their advantage of being able to pick up very subtle brightness variations becomes a disadvantage to the more lighting invariant features.

In the last years, researchers have tackled the multiple-daytime tracking challenge with deep learning approaches that are designed to convert nighttime images to daytime images, e.g. using GANs [54, 76, 98]. While this improves the robustness to changing lighting, one may ask why images should be the best input representation. Could there be better alternate representations?

This paper addresses the problem of adapting direct SLAM methods to challenging lighting and weather conditions. In this work, we show how to convert images into a multi-dimensional feature map which is invariant to lighting/weather changes and has by construction a larger basin of convergence. Thereby we overcome the aforementioned problems simultaneously. The deep features are trained with a novel Gauss-Newton loss formulation in a self-supervised manner. We employ a Siamese network trained with labels obtained either from simulation data or any state-of-the-art SLAM algorithm. This eliminates the additional cost of human labeling that is typically necessary for training a neural network. We exploit the probabilistic interpretation of the commonly used Gauss-Newton algorithm for direct image alignment. For this, we propose the Gauss-Newton loss which is designed to maximize the probability of identifying the correct pixel correspondence. The proposed loss function thereby enforces a feature representation that is designed to admit a large basin of convergence for the subsequent Gauss-Newton optimization. The superiority of our method stems from its ability to generate these multi-channel, weather-invariant deep features that facilitate relocalization across different weathers. Figure 6.1 shows how our method can successfully relocalize a snowy sequence in a pre-built map created using a sunny sequence.

In common benchmarks [115], localizing accurately in a pre-built map has been tackled by finding nearby images (e.g. by using NetVLAD [12]) and tracking the relative pose (6DOF) between them. However, we propose to split this into two separate tasks. In this work, we focus on the second challenge which we refer to as *relocalization tracking*. This way, we can evaluate its performance in isolation. This is formalized to what we refer to as *relocalization tracking*. Since there is no publicly available dataset to evaluate *relocalization tracking* performance across multiple types of weather, we are releasing an evaluation benchmark having the following 3 attributes:

- It contains sequences from multiple different kinds of weather.
- Pixel-wise correspondences between sequences are provided for both simulated and real-world datasets.
- It decouples relocalization tracking from the image retrieval task.

The challenge here in comparison with normal pose estimation datasets [17, 37] is that the images involved are usually captured at different daytimes/seasons and there is no good initialization of the pose. We summarize the main contributions of our paper as:

- We derive the Gauss-Newton loss formulation based on the properties of direct image alignment and demonstrate that it improves the robustness to large baselines and illumination/weather changes.

- Our experimental evaluation shows, that GN-Net outperforms both state-of-the-art direct and indirect SLAM methods on the task of *relocalization tracking*.
- We release a new evaluation benchmark for the task of *relocalization tracking* with ground-truth poses. It is collected under dynamic conditions such as illumination changes, and different weathers. Sequences are taken from the the CARLA [31] simulator as well as from the Oxford RobotCar dataset [82].

6.2 RELATED WORK

We review the following main areas of related work: visual SLAM, visual descriptor learning, deep direct image alignment, and image-based relocalization in SLAM.
Direct versus indirect SLAM methods: Most existing SLAM systems that have used feature descriptors are based on traditional manual feature engineering, such as ORB-SLAM [87], MonoSLAM [29], and PTAM [66].

An alternative to feature-based methods is to skip the pre-processing step of the raw sensor measurements and rather use the pixel intensities directly. Popular direct visual methods are DTAM [91], LSD-SLAM [35], DSO [34], and PhotoBundle [8]. However, the main limitation of direct methods is the *brightness constancy* assumption which is rarely fulfilled in any real-world robotic application [94]. The authors of [9] propose to use binary feature descriptors for direct tracking called Bit-planes. While improving the robustness to bad lighting situations it was also found that Bit-planes have a smaller convergence basin than intensities. This makes their method less robust to bad initialization. In contrast, the features we propose *both* improve robustness to lighting and the convergence basin.
Visual descriptor learning: Feature descriptors play an important role in a variety of computer vision tasks. For example, [22] proposed a novel correspondence contrastive loss which allows for faster training and demonstrates their effectiveness for both geometric and semantic matching across intra-class shape or appearance variations. In [116], a deep neural network is trained using a contrastive loss to produce viewpoint- and lighting-invariant descriptors for single-frame localization. The authors of [105] proposed a CNN-based model that learns local patterns for image matching without a global geometric model. [138] uses convolutional neural networks to compute descriptors which allow for efficient detection of poorly textured objects and estimation of their 3D pose. In [19], the authors propose to train features for optical flow estimation using a Hinge loss based on correspondences. In contrast to our work, their loss function does not have a probabilistic derivation and they do not apply their features to pose estimation. [42] uses deep learning to improve SLAM perfor-

mance in challenging situations. They synthetically create images and choose the one with most gradient information as the ground truth for training. In contrast to them, we do not limit our network to output images similar to the real world. In [28], the authors compare dense descriptors from a standard CNN, SIFT, and normal image intensities for dense Lucas-Kanade tracking. There, it can be seen that grayscale values have a better convergence basin than the other features, which is something we overcome with our approach.

Deep direct image alignment: BA-NET [132] introduces a network architecture to solve the structure from motion (SfM) problem via feature-metric bundle adjustment. Unlike the BA-NET, instead of predicting the depth and the camera motion simultaneously, we propose to only train on correspondences obtained from a direct SLAM system. The advantage is that correspondences are oftentimes easier to obtain than accurate ground-truth poses. Furthermore, we combine our method with a state-of-the-art direct SLAM system and utilize its depth estimation, whereas BA-NET purely relies on deep learning. RegNet [46] is another line of work which tries to replace the handcrafted numerical Jacobian by a learned Jacobian with the help of a depth prediction neural network. However, predicting a dense depth map is often inaccurate and computationally demanding. The authors of [80] propose to use a learning-based inverse compositional algorithm for dense image alignment. The drawback of this approach is that the algorithm is very sensitive to the data distribution and constrained towards selecting the right hyperparameters. In [56] they use high-dimensional features in a direct image alignment framework for monocular VO. In contrast to us, they only use already existing features and do not apply them for relocalization.

Relocalization: An important task of relocalization is to approximate the pose of an image by simply querying the most similar image from a database [27, 60]. However, this has only limited accuracy unless the 6DOF pose between the queried and the current image is estimated in a second step. Typically, this works by matching 2D-3D correspondences between an image and a point cloud and estimating the pose using indirect image alignment [88]. In contrast, we propose to use direct image alignment paired with deep features.

Relocalization benchmarks: The authors of [115] have done sequence alignment on the Oxford RobotCar dataset, however, they have not made the matching correspondences public. The Photo Tourism [123] is another dataset providing images and ground-truth correspondences of popular monuments from different camera angles and across different weather/lighting conditions. However, since the images are not recorded as a sequence, relocalization tracking is not possible. Furthermore, their benchmark only supports the submission of features rather than poses, thereby restricting evaluation to only indirect methods.

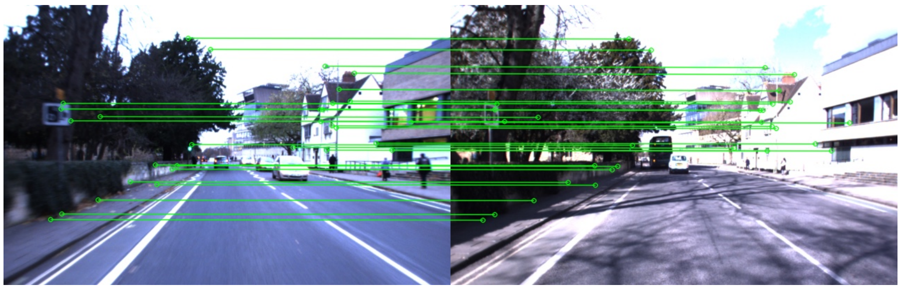


Figure 6.2: This figure shows training correspondences between a pair of images from our benchmark.

6.3 DEEP DIRECT SLAM

In this work, we argue that a network trained to output features which produce better inputs for direct SLAM as opposed to normal images should have the following properties:

- Pixels corresponding to the same 3D point should have similar features.
- Pixels corresponding to different 3D points should have dissimilar features.
- When starting in a vicinity around the correct pixel, the Gauss-Newton algorithm should move toward the correct solution.

For optimizing the last property, we propose the novel Gauss-Newton loss which makes use of the probabilistic background of the Gauss-Newton algorithm for direct image alignment. The final loss is a weighted sum of the pixel-wise contrastive loss and the Gauss-Newton loss.

Architecture: We are interested in learning a non-linear mapping, which maps images, $\mathbb{R}^{W \times H \times C}$ to a dense visual descriptor space, $\mathbb{R}^{W \times H \times D}$, where each pixel is represented by a D-dimensional vector. The training is performed by a Siamese encoder-decoder structured network, where we feed a pair of images, $\mathbf{I}_a$ and $\mathbf{I}_b$, producing multi-scale feature pyramids $\mathbf{F}_a^l$ and $\mathbf{F}_b^l$, where l represents the level of the decoder. For each image pair, we use a certain number of matches, denoted by N_{pos}, and a certain number of non-matches, denoted by N_{neg}. A pixel $\mathbf{u}_a \in \mathbb{R}^2$ from image $\mathbf{I}_a$ is considered to be a positive example if the pixel $\mathbf{u}_b \in \mathbb{R}^2$ from image $\mathbf{I}_b$ corresponds to the same 3D vertex (Figure 6.2). We make use of the inherent multi-scale hierarchy of the U-Net [106] architecture to apply the different loss terms from coarser to finer-scaled pyramid levels. With this approach, our learned features will have a larger convergence radius for visual SLAM methods.

Pixelwise contrastive loss: The pixelwise contrastive loss attempts to minimize the distance between positive pairs, and maximize the distance between negative pairs. It can be computed as follows:

$$\mathcal{L}_{\text{contrastive}}\left(\mathbf{F}_a, \mathbf{F}_b, l\right) = \mathcal{L}_{\text{pos}}\left(\mathbf{F}_a, \mathbf{F}_b, l\right) + \mathcal{L}_{\text{neg}}\left(\mathbf{F}_a, \mathbf{F}_b, l\right) \tag{6.1}$$

$$\mathcal{L}_{\text{pos}}\left(\mathbf{F}_a, \mathbf{F}_b, l\right) = \frac{1}{N_{\text{pos}}} \sum_{N_{\text{pos}}} D_{\text{feat}}^2 \tag{6.2}$$

$$\mathcal{L}_{\text{neg}}\left(\mathbf{F}_a, \mathbf{F}_b, l\right) = \frac{1}{N_{\text{neg}}} \sum_{N_{\text{neg}}} \max\left(0, M - D_{\text{feat}}\right)^2 \tag{6.3}$$

where $D_{\text{feat}}(\cdot)$ is the L_2 distance between the feature embeddings: $D_{\text{feat}} = \left\|\mathbf{F}_a^l(\mathbf{u}_a) - \mathbf{F}_b^l(\mathbf{u}_b)\right\|_2$ and M is the margin and set to 1.

Gauss-Newton algorithm for direct image alignment: Our learned deep features are ultimately applied to pose estimation. This is done using direct image alignment but generalized to a multi-channel feature map $\mathbf{F}$ with D channels. The input to this algorithm is a reference feature map $\mathbf{F}$ with known depths for some pixels in the image, and a target feature map $\mathbf{F}'$. The output is the predicted relative pose ξ. Starting from an initial guess the following steps are performed iteratively:

1. All points $\mathbf{p}_i$ with known depth values are projected from the reference feature map $\mathbf{F}$ into the target feature map $\mathbf{F}'$ yielding the point $\mathbf{p'}_i$. For each of them a residual vector $\mathbf{r} \in \mathbb{R}^D$ is computed, enforcing that the reference pixel and the target pixel should be similar:

$$\mathbf{r}_i\left(\mathbf{p}_i, \mathbf{p'}_i\right) = \mathbf{F}'\left(\mathbf{p'}_i\right) - \mathbf{F}\left(\mathbf{p}_i\right) \tag{6.4}$$

2. For each residual the derivative with respect to the relative pose is:

$$\mathbf{J}_i = \frac{d\mathbf{r}_i}{d\xi} = \frac{d\mathbf{F}'\left(\mathbf{p'}_i\right)}{d\mathbf{p'}_i} \frac{d\mathbf{p'}_i}{d\xi} \tag{6.5}$$

   Notice that the reference point $\mathbf{p}_i$ does not change for different solutions ξ, therefore it does not appear in the derivative.

3. Using the stacked residual vector $\mathbf{r}$, the stacked Jacobian $\mathbf{J}$, and a diagonal weight matrix $\mathbf{W}$, the Gaussian system and the step $\boldsymbol{\delta}$ is computed as follows:

$$\mathbf{H} = \mathbf{J}^\top \mathbf{W} \mathbf{J} \text{ and } \mathbf{b} = -\mathbf{J}^\top \mathbf{W} \mathbf{r} \text{ and } \boldsymbol{\delta} = \mathbf{H}^{-1} \mathbf{b} \tag{6.6}$$

Note that this derivation is equivalent to normal direct image alignment (as done in the frame-to-frame tracking from DSO) when replacing $\mathbf{F}$ with the image $\mathbf{I}$. In the computation of the Jacobian, the numerical derivative of the features $\frac{d\mathbf{F}'(\mathbf{p'}_i)}{d\mathbf{p'}_i}$ is used. As typical images are extremely non-convex this derivative is only valid in a small vicinity (usually 1 to 2 pixels) around the current solution which is the main reason why direct image alignment needs a good initialization. To partially overcome this, a pyramid scheme is often used. Usually tracking on multiple channels instead of one can decrease the convergence radius ([9, 28]). However, in our case, we train the feature maps to in fact have a larger convergence basin than images by enforcing smoothness in the vicinity of the correct correspondence.

Gauss-Newton on individual pixels: Instead of running the Gauss-Newton algorithm on the 6DOF pose we can instead use it on each point $\mathbf{p}_i$ individually (which is similar to the Lucas-Kanade algorithm [79]). Compared to direct image alignment, this optimization problem has the same residual, but the parameter being optimized is the point position instead of the relative pose. In this case, the Hessian will be a 2×2 matrix and the step $\boldsymbol{\delta}$ can simply be added to the current pixel position (we leave out $\mathbf{W}$ for simplicity):

$$\mathbf{J'}_i = \frac{d\mathbf{F}'(\mathbf{p'}_i)}{d\mathbf{p'}_i} \text{ and } \mathbf{H}'_i = \mathbf{J'}_i^\top \mathbf{J}_i \text{ and } \mathbf{b'}_i = \mathbf{J'}_i^\top \mathbf{r}_i \tag{6.7}$$

These individual Gauss-Newton systems can be combined with the system for 6DOF pose estimation (Equation (6.6)) using:

$$\mathbf{H} = \sum_i \left(\frac{d\mathbf{p'}_i}{d\xi}\right)^\top \mathbf{H'}_i \left(\frac{d\mathbf{p'}_i}{d\xi}\right) \text{ and } \mathbf{b} = \sum_i \left(\frac{d\mathbf{p'}_i}{d\xi}\right)^\top \mathbf{b'}_i. \tag{6.8}$$

The difference between our simplified systems and the one for pose estimation is only the derivative with respect to the pose, which is much smoother than the image derivative [34].

This means that if the Gauss-Newton algorithm performs well on individual pixels it will also work well on estimating the full pose. Therefore, we propose to train a neural network on correspondences which are easy to obtain, e.g. using a SLAM method, and then later apply it for pose estimation.

We argue that training on these individual points is superior to training on the 6DOF pose. The estimated pose can be correct even if some points contribute very wrong estimates. This increases robustness at runtime but when training we want to improve the information each point provides. Also, when training on the 6DOF pose we only have one supervision signal for each image pair, whereas when training on correspondences we have over a thousand signals. Hence, our method exhibits exceptional generalization capabilities as shown in the results section.

The probabilistic Gauss-Newton loss: The linear system described in Equation (6.7) defines a 2-dimensional Gaussian probability distribution. The reason is that the Gauss-Newton algorithm tries to find the solution with maximum probability in a least squares fashion.

This can be derived using the negative log-likelihood of the Gaussian distribution:

$$
\begin{aligned}
E(\mathbf{x}) &= -\log f_X(\mathbf{x}) && (6.9)\\
&= \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}) + \log\left(2\pi\sqrt{|\boldsymbol{\Sigma}|}\right) && (6.10)\\
&= \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \mathbf{H}(\mathbf{x}-\boldsymbol{\mu}) + \log(2\pi) - \frac{1}{2}\log(|\mathbf{H}|) && (6.11)
\end{aligned}
$$

where $\mathbf{x}$ is a pixel position and $\boldsymbol{\mu}$ is the mean.

In the Gauss-Newton algorithm the mean (which also corresponds to the point with maximum probability) is computed with $\boldsymbol{\mu} = \mathbf{x}_s + \boldsymbol{\delta}$, where the $\boldsymbol{\delta}$ comes from Equation (6.6) and $\mathbf{x}_s$ denotes the start point. To derive this, only the first term is used (because the latter parts are constant for all solution $\mathbf{x}$). In our case, however, the second term is very relevant, because the network can influence both $\boldsymbol{\mu}$ and $\mathbf{H}$.

This derivation shows, that $\mathbf{H}, \mathbf{b}$ as computed in the GN-algorithm, also define a Gaussian probability distribution with mean $\mathbf{x}_s + \mathbf{H}^{-1}\mathbf{b}$ and covariance $\mathbf{H}^{-1}$.

When starting with an initial solution $\mathbf{x}_s$ the network should assign maximal probability to the pixel that marks the correct correspondence. With $\mathbf{x}$ being the correct correspondence, we therefore use $E(\mathbf{x})$ = Equation (6.11) as our loss function which we call the *Gauss-Newton loss* (see Algorithm 1).

Algorithm 1 Compute Gauss-Newton loss

```
$\mathbf{F}_a \leftarrow \text{network}(\mathbf{I}_a)$
$\mathbf{F}_b \leftarrow \text{network}(\mathbf{I}_b)$
$e \leftarrow 0$                                              ▷ Total error
for all correspondences $\mathbf{u}_a, \mathbf{u}_b$ do
    $\mathbf{f}_t \leftarrow \mathbf{F}_a(\mathbf{u}_a)$                              ▷ Target feature
    $\mathbf{x}_s \leftarrow \mathbf{u}_b + \text{rand}(\text{vicinity})$             ▷ Compute start point
    $\mathbf{f}_s \leftarrow \mathbf{F}_b(\mathbf{x}_s)$
    $\mathbf{r} \leftarrow \mathbf{f}_s - \mathbf{f}_t$                               ▷ Residual
    $\mathbf{J} \leftarrow \frac{d\mathbf{F}_b}{d\mathbf{x}_s}$                       ▷ Numerical derivative
    $\mathbf{H} \leftarrow \mathbf{J}^\top\mathbf{J} + \epsilon\mathbf{I}_n$          ▷ Added epsilon for invertibility
    $\mathbf{b} \leftarrow \mathbf{J}^\top r$
    $\boldsymbol{\mu} \leftarrow \mathbf{x}_s - \mathbf{H}^{-1}\mathbf{b}$
    $e_1 \leftarrow \frac{1}{2}(\mathbf{u}_b - \boldsymbol{\mu})^\top\mathbf{H}(\mathbf{u}_b - \boldsymbol{\mu})$   ▷ First error term
    $e_2 \leftarrow \log(2\pi) - \frac{1}{2}\log(|\mathbf{H}|)$                      ▷ Second error term
    $e \leftarrow e + e_1 + e_2$
end for
```

In the algorithm, a small number ϵ is added to the diagonal of the Hessian, to ensure it is invertible.

Analysis of the Gauss-Newton loss: By minimizing Equation (6.11) the network has to maximize the probability density of the correct solution. As the integral over the probability densities always has to be

1, the network has the choice to either focus all the density on a small set of solutions (with more risk of being penalized if this solution is wrong), or to distribute the density to more solutions which in turn will have a lower individual density. By maximizing the probability of the correct solution, the network is incentivized to improve the estimated solution and its certainty.

This is also reflected in the two parts of the loss. The first term $e_1 = \frac{1}{2}\left(\mathbf{u}_b - \mu\right)^\top \mathbf{H}\left(\mathbf{u}_b - \mu\right)$ penalizes deviations between the estimated and the correct solution, scaled with the Hessian $\mathbf{H}$. The second term $e_2 = \log\left(2\pi\right) - \frac{1}{2}\log\left(|\mathbf{H}|\right)$ is large if the network does not output enough certainty for its solution. This means that the network can reduce the first error term e_1 by making $\mathbf{H}$ smaller. As a consequence, the second error term will be increased, as this will also reduce the determinant of $\mathbf{H}$. Notice also that this can be done in both dimensions independently. The network has the ability to output a large uncertainty in one direction, but a small uncertainty in the other direction. This is one of the traditional advantages of direct methods which are naturally able to utilize also lines instead of just feature points.

From Equation (6.11), it can be observed that the predicted uncertainty depends only on the numerical derivative of the target image at the start position. The higher the gradients the higher the predicted certainty. In DSO this is an unwanted effect that is counteracted by the gradient-dependent weighting applied to the cost-function [34, Equation (7)]. In our case, however, it gives the network the possibility to express its certainty and incentivizes it to output discriminative features.

Upon training the network with our loss formulation, we observe that the features are very similar despite being generated from images taken from sequences with different lighting/weather conditions, as shown in Figure 6.3.

6.4 RELOCALIZATION TRACKING BENCHMARK

Previous tasks for localization/odometry can primarily be divided into two categories:

- Odometry datasets [17, 37], where there is a continuous stream of images (sometimes combined with additional sensor data like IMUs).
- Image collections where individual images are usually further apart from each other in space/time [63, 115].

We argue that for several applications a combination of these two tasks which we refer to as *relocalization tracking* is a more realistic scenario. The idea is that the algorithm has two inputs:

1. An image sequence (like a normal odometry dataset).

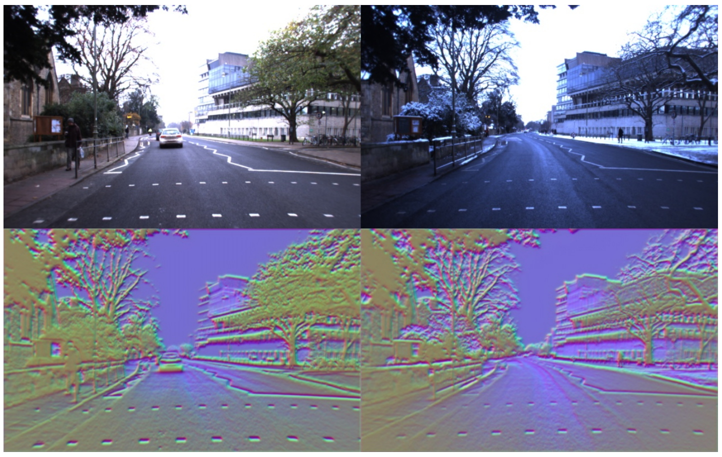


Figure 6.3: This figure shows images and their corresponding feature maps predicted by our GN-Net for the Oxford RobotCar dataset. Each column depicts the image and feature map for a sample taken from 2 different sequences. Despite lighting and weather changes, the feature maps are robust to these variations. The visualization of the features shows the high-dimensional descriptors reduced to 3D through PCA.

2. A collection of individual images (possibly with different weathers/times), each of which shall be tracked against one specific image from point 1.

The algorithm is supposed to track the normal sequential image sequence and at the same time perform tracking of the images in point 2. The advantage of this task is that the used algorithm can utilize the temporally continuous sequence from point 1 to compute accurate depth values for a part of the image (using a standard visual odometry method), which can then be used to improve the tracking of the individual images of point 2.

This task is very realistic as it comes up when tracking an image sequence and at the same time trying to relocalize this sequence in a prior map. A similar challenge occurs by trying to merge multiple maps from different times. In both cases, one has more information than just a random collection of images. It is important to reiterate here that the task of finding relocalization candidates is not considered but rather tracking them with maximum accuracy/robustness is the focus. This is because our benchmark decouples image retrieval from tracking.

We have created a benchmark for *relocalization tracking* using the CARLA simulator and the Oxford RobotCar dataset. Our benchmark includes ground-truth poses between different sequences for training, validation, and testing. We focus on the use case of relocalization in the context of autonomous driving. Therefore, our datasets contain limited point-of-view changes but strong lighting and weather changes.

CARLA: For synthetic evaluations, we use CARLA version 0.8.2. We collect data for 3 different weather conditions representing *WetNoon*, *SoftRainNoon*, and *WetCloudySunset*. We recorded the images at a fixed framerate of 10 frames per second (FPS). At each time step, we record images and its corresponding dense depth map from 6 different cameras with different poses rendered from the simulation engine, which means that the poses in the benchmark are not limited to just 2DOF. The images and the dense depth maps are of size 512×512. For each weather condition, we collected 3 different sequences comprising 500-time steps with an average distance of $1.6\,\mathrm{m}$. This is done for training, validation, and testing, meaning there are 27 sequences, containing 6 cameras each. Training, validation, and test sequences were all recorded in different parts of the CARLA town. We have generated the test sequences after all hyperparameter tuning of our method was finished, meaning we had no access to the test data when developing the method. In accordance, we shall withhold the ground truth for the test sequences.
Oxford RobotCar: Creating a multi-weather benchmark for this dataset imposes various challenges because the GPS-based ground truth is very inaccurate. To find the relative poses between images from different sequences we have used the following approach. For pairs of images from two different sequences, we accumulate the point cloud captured by the 2D lidar for 60 meters using the visual odometry result provided by the Oxford dataset. The resulting two point clouds are aligned with the global registration followed by ICP alignment using the implementation of Open3D [150]. We provide the first pair of images manually and the following pairs are found using the previous solution. We have performed this alignment for the following sequences: *2014-12-02-15-30-08 (overcast)* and *2015-03-24-13-47-33 (sunny)* for training. For testing, we use the reference sequence *2015-02-24-12-32-19 (sunny)* and align it with the sequences *2015-03-17-11-08-44 (overcast)*, *2014-12-05-11-09-10 (rainy)*, and *2015-02-03-08-45-10 (snow)*. The average relocalization distance across all sequences is $0.84\,\mathrm{m}$.

6.5 EXPERIMENTAL EVALUATION

We perform our experiments on the *relocalization tracking* benchmark described in Section 6.4. We demonstrate the multi-weather relocalization performance on both the CARLA and the Oxford RobotCar dataset. For the latter, we show that our method even generalizes well to unseen weather conditions like rain or snow while being trained only on the sunny and overcast conditions. Furthermore, a qualitative relocalization demo[1] on the Oxford RobotCar dataset is provided, where we demonstrate that our GN-Net can facilitate precise relocalization between weather conditions.

1 https://vision.in.tum.de/gn-net.

We train our method using sparse depths created by running Stereo DSO on the training sequences. We use intra-sequence correspondences calculated using the DSO depths and the DSO pose. Meanwhile, inter-sequence correspondences are obtained using DSO depths and the ground-truth poses provided by our benchmark. The ground truth poses are obtained via Lidar alignment for Oxford and directly from the simulation engine for CARLA as explained in Section 6.4. Training is done from scratch with randomly initialized weights and an Adam optimizer with a learning rate of 10^{-6}. The image pair fed to the Siamese network is randomly selected from any of the training sequences while ensuring that the images in the pair do not differ by more than 5 keyframes. Each branch of the Siamese network is a modified U-Net architecture with shared weights. Further details of the architecture and training can be found in the supplementary material. Note that at inference time, only one image is needed to extract the deep visual descriptors, used as input to the SLAM algorithm. While in principle, our approach can be deployed in conjunction with any direct method, we have coupled our deep features with Direct Sparse Odometry (DSO).

We compare to state-of-the-art **direct** methods:

Stereo Direct Sparse Odometry (DSO) [136]: Whenever there is a relocalization candidate for a frame we ensure that the system creates the corresponding keyframe. This candidate is tracked using the *coarse tracker*, performing direct image alignment in a pyramid scheme. We use the identity as initialization without any other random guesses for the pose.

GN-Net (Ours): Same as with DSO, however, for relocalization tracking, we replace the grayscale images with features created by our GN-Net on all levels of the feature pyramid. The network is trained with the Gauss-Newton loss formulation described in Section 6.3.

We also compare to state-of-the-art **indirect** methods:

ORB-SLAM [88]: For relocalization tracking, we use the standard feature-based 2-frame pose optimization also used for frame-to-keyframe tracking. We have also tried the RANSAC scheme implemented in ORB-SLAM for relocalization, however, it yielded worse results overall. Thus we will report only the default results.

D2-Net [33], SuperPoint [30]: For both methods we use the models provided by the authors. The relative pose is estimated using the OpenCV implementation of the PnP algorithm in a RANSAC scheme.

We also evaluated the Deeper Inverse Compositional Algorithm [80] on the *relocalization tracking* benchmark. However, the original implementation didn't converge despite multiple training trials with different hyperparameters.

For all our quantitative experiments we plot a cumulative distribution of the relocalization error, which is the norm of the translation between the estimated and the correct solution in meters. For each relocalization error between 0 and 1 meter, it plots the percentage of relocalization candidates that have been tracked with at least this accuracy.

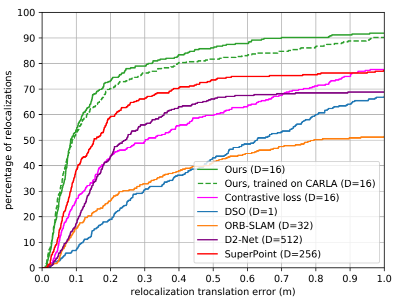
6.5.1 *Quantitative Multi-Weather Evaluation*

We demonstrate the relocalization tracking accuracy on our new benchmark across different weathers. For these experiments, tracking is performed only across sequences with a different weather condition.
CARLA: For this experiment, we train on the training sequences provided by our benchmark. For all learning-based approaches, the best epoch is selected using the relocalization tracking performance on the validation set. The results on the test data are shown in the supplementary.
Oxford RobotCar: We train on the sunny and overcast condition correspondences provided by our *relocalization tracking* benchmark for the Oxford dataset. For the learning-based methods, we select the best epoch based on the relocalization tracking performance on the training set. We use the same hyperparameters that were found using the CARLA validation set. We show the results on the test data in Figure 6.4. Our method significantly outperforms the baselines. The Gauss-Newton loss has a large impact as compared to the model trained with only the contrastive loss.

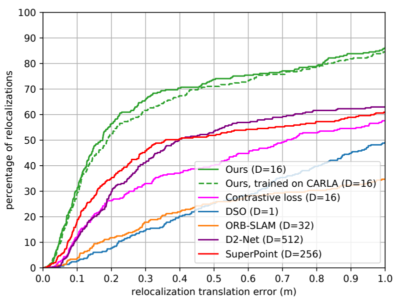
Figures 6.4b to 6.4f show how well our model generalizes to unseen weather conditions. Despite being trained only on two sequences with overcast and sunny conditions the results for tracking against a rainy and a snowy sequence are almost the same. Interestingly our model which was trained only on the CARLA benchmark outperforms all baselines significantly.

6.5.2 *Qualitative Multi-Weather Evaluation*

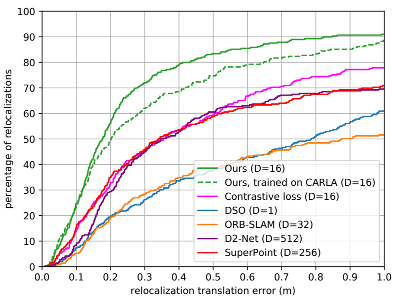
Finally, we show a relocalization demo comparing our GN-Net to DSO. For this, we load a point cloud from a sequence recorded in the sunny condition and relocalize against sequences from rainy and snowy conditions. For each keyframe, we try to track it against the nearest keyframe in the map according to the currently estimated transformation between the trajectory and the map. Figure 6.6 shows that the point clouds from the different sequences align nicely, despite belonging to different weather conditions. This experiment shows that our method can perform the desired operations successfully on a real-world application, including relocalization against unseen weather conditions. Figure 6.7 demonstrates the difference between our Gauss-Newton loss and the contrastive loss. This shows that the quantitative improvement has a visible effect on the application of



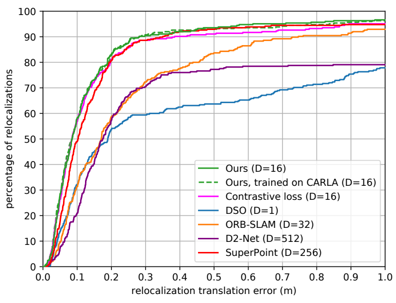
(a) Relocalization sunny and overcast.



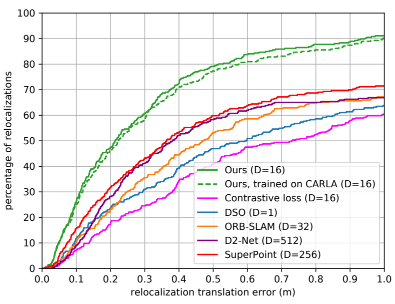
(b) Relocalization sunny and rainy.



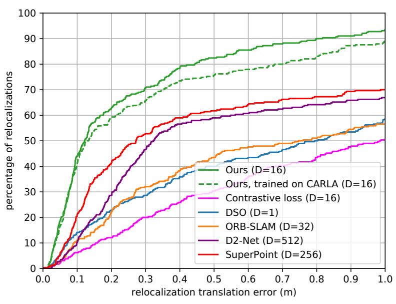
(c) Relocalization sunny and snowy.



(d) Relocalization overcast and rainy.



(e) Relocalization overcast and snowy.



(f) Relocalization rainy and snowy.

Figure 6.4: This figure shows the cumulative relocalization accuracy on the Oxford RobotCar dataset for different sequences. D denotes the dimension of the feature descriptor. Our method achieves the highest accuracy across all sequences. It is interesting to observe that despite being trained only on two sequences in overcast and sunny condition, our model still generalizes very well to even *unseen* rainy and snowy conditions. Even the model trained only on the synthetic CARLA benchmark outperforms all baselines, showing exceptional generalization capabilities.

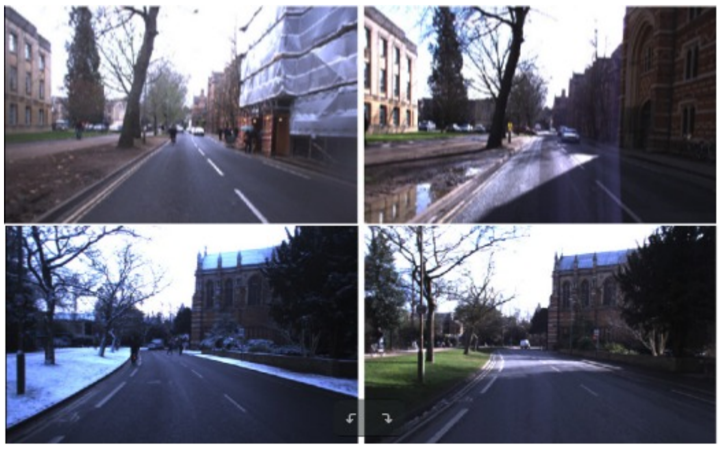


Figure 6.5: This figure shows image pairs used in the qualitative relocalizations. Left: rainy (top row) and snowy (bottom row) images relocalized against the sunny reference images (right).

relocalization. Figure 6.5 shows sample images used in the qualitative relocalizations.

6.5.3 *Additional Experiments on EuRoC and CARLA*

In the supplementary, we provide more evaluations on datasets with and without brightness variations. This includes relocalization tracking on the CARLA benchmark and visual odometry on the EuRoC [17] dataset. We show that also in these situations our deep features significantly outperform DSO and ORB-SLAM because of their robustness to large-baselines. On the EuRoC dataset, we improve the DSO performance by almost a factor of 2 for low-framerates.

6.6 CONCLUSION & FUTURE WORK

With the advent of deep learning, we can devise feature space encodings that are designed to be optimally suited for the subsequent visual SLAM algorithms. More specifically, we propose to exploit the probabilistic interpretation of the commonly used Gauss-Newton al-

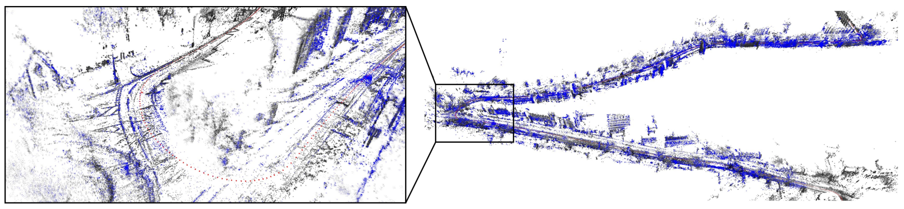


Figure 6.6: This figure shows a point cloud result of our GN-Net. We relocalize a rainy sequence (blue) against a reference map created from the sunny sequence (gray).

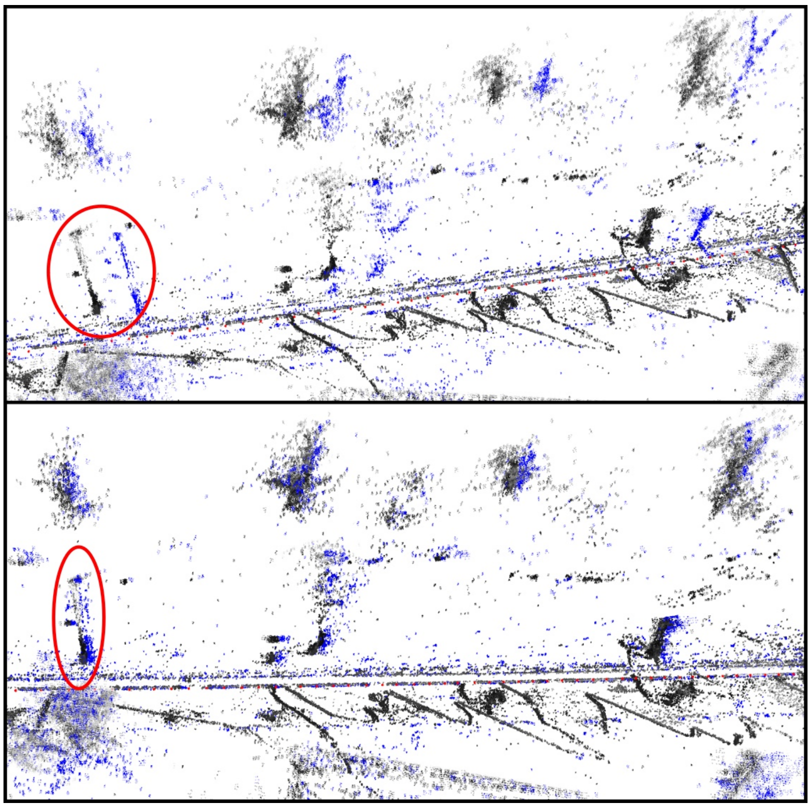


Figure 6.7: Top: relocalization using the model trained with only the contrastive loss. Bottom: relocalization using the model trained with our loss formulation. This visually demonstrates the influence of the Gauss-Newton loss.

gorithm to devise a novel loss function for feature space encoding that we call the Gauss-Newton loss. It is designed to promote robustness to strong lighting and weather changes while enforcing a maximal basin of convergence for the respective SLAM algorithm. Quantitative experiments on synthetic and real-world data demonstrate that the Gauss-Newton loss allows us to significantly expand the realm of applicability of direct visual SLAM methods, enabling relocalization and map merging across drastic variations in weather and illumination.

7

4SEASONS: A CROSS-SEASON DATASET FOR MULTI-WEATHER SLAM IN AUTONOMOUS DRIVING

Authors:	Patrick Wenzel[1,2] Rui Wang[1,2] Nan Yang[1,2] Qing Cheng[2] Qadeer Khan[1,2] Lukas von Stumberg[1,2] Niclas Zeller[2] Daniel Cremers[1,2]
	[1] Technical University of Munich [2] Artisense

Conference:	*German Conference on Pattern Recognition (GCPR), 2020*
DOI:	10.1007/978-3-030-71278-5_29

Abstract: We present a novel dataset covering seasonal and challenging perceptual conditions for autonomous driving. Among others, it enables research on visual odometry, global place recognition, and map-based re-localization tracking. The data was collected in different scenarios and under a wide variety of weather conditions and illuminations, including day and night. This resulted in more than 350 km of recordings in nine different environments ranging from multi-level parking garage over urban (including tunnels) to countryside and highway. We provide globally consistent reference poses with up-to centimeter accuracy obtained from the fusion of direct stereo visual-inertial odometry with RTK-GNSS. The full dataset is available at `https://www.4seasons-dataset.com`.

Revised layout and minor adaptations. Accepted version of original publication [7] and detailed disclaimer are included in Appendix A.4.

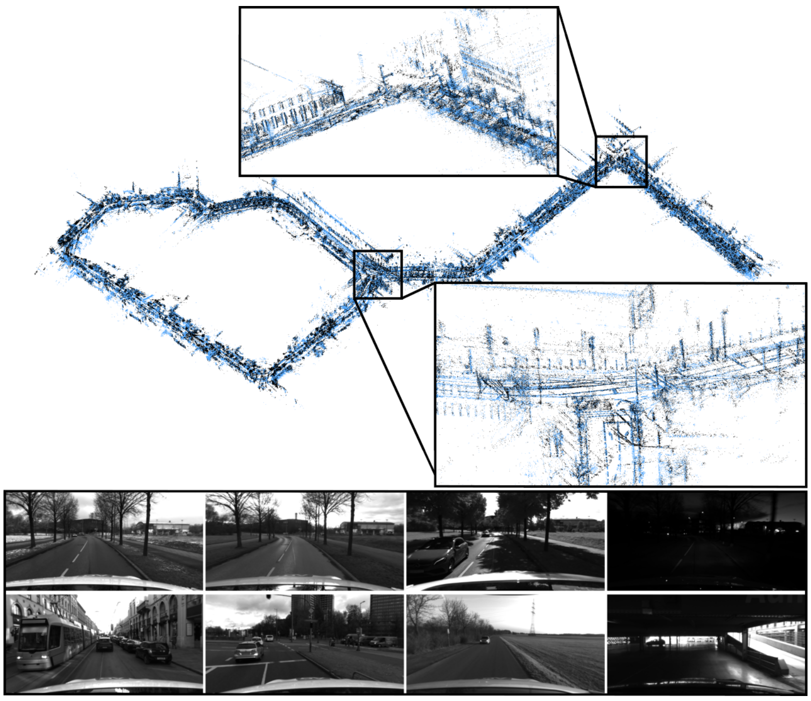


Figure 7.1: **Dataset overview.** Top: overlaid maps recorded at different times and environmental conditions. The points from the reference map (black) align well with the points from the query map (blue), indicating that the reference poses are indeed accurate. Bottom: sample images demonstrating the diversity of our dataset. The first row shows a collection from the same scene across different weather and lighting conditions: snowy, overcast, sunny, and night. The second row depicts the variety of scenarios within the dataset: inner city, suburban, countryside, and a parking garage.

7.1 INTRODUCTION

During the last decade, research on visual odometry (VO) and simultaneous localization and mapping (SLAM) has made tremendous strides [34, 35, 87, 91] particularly in the context of autonomous driving (AD) [36, 88, 136, 143]. One reason for this progress has been the publication of large-scale datasets [18, 26, 40] tailored for benchmarking these methods. Naturally, the next logical step towards progressing research in the direction of visual SLAM has been to make it robust under dynamically changing and challenging conditions. This includes VO, e.g. at night or rain, as well as long-term place recognition and re-localization against a pre-built map. In this regard, the advent of deep learning has exhibited itself to have promising potential in complementing the performance of visual SLAM [33, 56, 58, 3]. Therefore, it has become all the more important to have datasets that are commensurate with handling the challenges of any real-world environment while also being capable of discerning the performance of state-of-the-art approaches.

To accommodate this demand, we present in this paper a versatile cross-season and multi-weather dataset on a large-scale focusing on long-term localization for autonomous driving. By traversing the same stretch under different conditions and over a long-term time horizon, we capture variety in illumination and weather as well as in the appearance of the scenes. Figure 7.1 visualizes two overlaid 3D maps recorded at different times as well as sample images of the dataset.

In detail this work adds the following contributions to the state-of-the-art:

- A cross-season/multi-weather dataset for long-term visual SLAM in automotive applications containing more than 350 km of recordings.
- Sequences covering nine different kinds of environments ranging from multi-level parking garage over urban (including tunnels) to countryside and highway.
- Global six degrees of freedom (6DoF) reference poses with up-to centimeter accuracy obtained from the fusion of direct stereo visual-inertial odometry (VIO) with RTK-GNSS.
- Accurate cross-seasonal pixel-wise correspondences to train dense feature representations.

7.2 RELATED WORK

There exists a variety of benchmarks and datasets focusing on VO and SLAM for AD. Here, we divide these datasets into the ones which focus only on the task of VO as well as those covering different weather conditions and therefore aiming towards long-term SLAM.

7.2.1 *Visual Odometry*

The most popular benchmark for AD certainly is KITTI [40]. This multi-sensor dataset covers a wide range of tasks including not only VO, but also 3D object detection, and tracking, scene flow estimation as well as semantic scene understanding. The dataset contains diverse scenarios ranging from urban over countryside to highway. Nevertheless, all scenarios are only recorded once and under similar weather conditions. Ground truth is obtained based on a high-end inertial navigation system (INS).

Another dataset containing LiDAR, inertial measurement unit (IMU), and image data at a large-scale is the Málaga Urban dataset [14]. However, in contrast to KITTI, no accurate 6DoF ground truth is provided and therefore it does not allow for a quantitative evaluation based on this dataset.

Other popular datasets for the evaluation of VO and VIO algorithms not related to AD include [127] (handheld RGB-D), [17] (UAV stereo-inertial), [37] (handheld mono), and [118] (handheld stereo-inertial).

7.2.2 *Long-Term SLAM*

More related to our work are datasets containing multiple traversals of the same environment over a long period of time. With respect to SLAM for autonomous driving, the Oxford RobotCar Dataset [82] represents a kind of pioneer work. This dataset consists of large-scale sequences recorded multiple times for the same environment over a period of one year. Hence, it covers large variations in the appearance and structure of the scene. However, the diversity of the scenarios is only limited to an urban environment. Also, the ground truth provided for the dataset is not accurate up-to centimeter-level and therefore, requires additional manual effort to establish accurate cross-sequence correspondences.

The work [115] represents a kind of extension to [82]. This benchmark is based on subsequences from [82] as well as other datasets. The ground truth of the RobotCar Seasons [115] dataset is obtained based on structure from motion (SfM) and LiDAR point cloud alignment. However, due to inaccurate GNSS measurements [82], a globally consistent ground truth up-to centimeter-level can not be guaranteed. Furthermore, this dataset only provides one reference traversal in the overcast condition. In contrast, we provide globally consistent reference models for all traversals covering a wide variety of conditions. Hence, every traversal can be used as a reference model that allows further research, e.g. on analyzing suitable reference-query pairs for long-term localization and mapping.

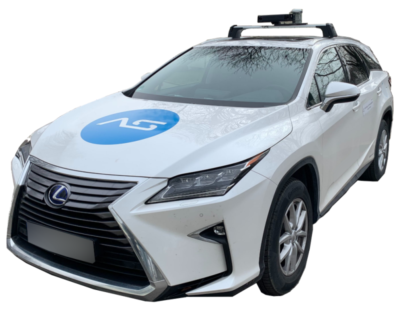
7.2.3 *Other Datasets*

Examples of further multi-purpose AD datasets which also can be used for VO are [18, 26, 52, 137].

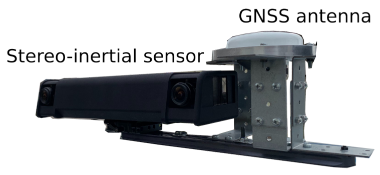
As stated in Section 7.1, our proposed dataset differentiates from previous related work in terms of being both large-scale (similar to [40]) as well as having high variations in appearance and conditions (similar to [82]). Furthermore, we are providing accurate reference poses based on the fusion of direct stereo VIO and RTK-GNSS.

7.3 SYSTEM OVERVIEW

This section presents the sensor setup which is used for data recording (Section 7.3.1). Furthermore, we describe the calibration of the entire sensor suite (Section 7.3.2) as well as our approach to obtain up-to centimeter-accurate global 6DoF reference poses (Section 7.3.3).



(a) Test vehicle.



(b) Sensor system.

Figure 7.2: **Recording setup.** Test vehicle and sensor system used for dataset recording. The sensor system consists of a custom stereo-inertial sensor with a stereo baseline of 30 cm and a high-end RTK-GNSS receiver from Septentrio.

7.3.1 *Sensor Setup*

The hardware setup consists of a custom stereo-inertial sensor for 6DoF pose estimation as well as a high-end RTK-GNSS receiver for global positioning and global pose refinement. Figure 7.2 shows our test vehicle equipped with the sensor system used for data recording.

7.3.1.1 *Stereo-Inertial Sensor*

The core of the sensor system is our custom stereo-inertial sensor. This sensor consists of a pair of monochrome industrial-grade global shutter cameras (Basler acA2040-35gm) and lenses with a fixed focal length of $f = 3.5\,\mathrm{mm}$ (Stemmer Imaging CVO GMTHR23514MCN). The cameras are mounted on a highly-rigid aluminum rail with a stereo baseline of $30\,\mathrm{cm}$. On the same rail, an IMU (Analog Devices ADIS16465) is mounted. All sensors, cameras, and IMU are triggered over an external clock generated by a field-programmable gate array (FPGA). Here, the trigger accounts for exposure compensations, meaning that the time between the centers of the exposure interval for two consecutive images is always kept constant (1/[frame rate]) independent of the exposure time itself.

Furthermore, based on the FPGA, the IMU is properly synchronized with the cameras. In the dataset, we record stereo sequences with a frame rate of 30 fps. We perform pixel binning with a factor of two and crop the image to a resolution of 800×400. This results in a field of view of approximately $77°$ horizontally and $43°$ vertically. The IMU is recorded at a frequency of $2000\,\mathrm{Hz}$. During recording, we run our custom auto-exposure algorithm, which guarantees equal exposure times for all stereo image pairs as well as a smooth exposure transition in highly dynamic lighting conditions, as it is required for visual SLAM. We provide those exposure times for each frame.

7.3.1.2 *GNSS Receiver*

For global positioning and to compensate drift in the VIO system we utilize an RTK-GNSS receiver (mosaic-X5) from Septentrio in combination with an Antcom Active G8 GNSS antenna. The GNSS receiver provides a horizontal position accuracy of up-to 6 mm by utilizing RTK corrections. While the high-end GNSS receiver is used for accurate positioning, we use a second receiver connected to the time-synchronization FPGA to achieve synchronization between the GNSS receiver and the stereo-inertial sensor.

7.3.2 *Calibration*

7.3.2.1 *Aperture and Focus Adjustment*

The lenses used in the stereo system have both adjustable aperture and focus. Therefore, before performing the geometric calibration of all sensors, we manually adjust both cameras for a matching average brightness and a minimum focus blur [51], across a structured planar target in 10 m distance.

7.3.2.2 *Stereo Camera and IMU*

For the intrinsic and extrinsic calibration of the stereo cameras as well as the extrinsic calibration and time-synchronization of the IMU, we use a slightly customized version of *Kalibr*[1] [100]. The stereo cameras are modeled using the Kannala-Brandt model [59], which is a generic camera model consisting of in total eight parameters. To guarantee an accurate calibration over a long-term period, we perform a feature-based epipolar-line consistency check for each sequence recorded in the dataset and re-calibrate before a recording session if necessary.

7.3.2.3 *GNSS Antenna*

Since the GNSS antenna does not have any orientation but has an isotropic reception pattern, only the 3D translation vector between one of the cameras and the antenna within the camera frame has to be known. This vector was measured manually for our sensor setup.

7.3.3 *Ground Truth Generation*

Reference poses (i.e. ground truth) for VO and SLAM should provide high accuracy in both local relative 6DoF transformations and global positioning. To fulfill the first requirement, we extend the state-of-the-art stereo direct sparse VO [136] by integrating IMU measurements [126], achieving a stereo-inertial SLAM system offering

1 https://github.com/ethz-asl/kalibr

average tracking drift around 0.6 % of the traveled distance. To fulfill the second requirement, the poses estimated by our stereo-inertial system are integrated into a global pose graph, each with an additional constraint from the corresponding RTK-GNSS measurement. Our adopted RTK-GNSS system can provide global positioning with up-to centimeter accuracy. The pose graph is optimized globally using the Gauss-Newton method, ending up with 6DoF camera poses with superior accuracy both locally and globally. For the optimization, we make use of the g2o library [68].

One crucial aspect for the dataset is that the reference poses which we provide are actually accurate enough, even though some of the recorded sequences partially contain challenging conditions in GNSS-denied environments. Despite the fact that the stereo-inertial sensor system has an average drift around 0.6 %, this cannot be guaranteed for all cases. Hence, for the reference poses in our dataset, we report whether a pose can be considered to be reliable by measuring the distance to the corresponding RTK-GNSS measurement. Only RTK-GNSS measurements with a reported standard deviation of less than $0.01\ \mathrm{m}$ are considered as accurate. For all poses, without corresponding RTK-GNSS measurement we do not guarantee a certain accuracy. Nevertheless, due to the highly accurate stereo-inertial odometry system, these poses still can be considered to be accurate in most cases even in GNSS-denied environments, e.g. tunnels or areas with tall buildings.

7.4 SCENARIOS

This section describes the different scenarios we have collected for the dataset. The scenarios involve different sequences – ranging from urban driving to parking garage and rural areas. We provide complex trajectories, which include partially overlapping routes, and multiple loops within a sequence. For each scenario, we have collected multiple traversals covering a large range of variations in environmental appearance and structure due to weather, illumination, dynamic objects, and seasonal effects. In total, our dataset consists of nine different scenarios, i.e. industrial area, highway, local neighborhood, ring road, countryside, suburban, inner city, monumental site, and multi-level parking garage.

We provide reference poses and 3D models generated by our ground truth generation pipeline (c.f. Figure 7.3) along with the corresponding raw image frames and raw IMU measurements. Figure 7.4 shows another example of the optimized trajectory, which depicts the accuracy of the provided reference poses.

The dataset will challenge current approaches on long-term localization and mapping since it contains data from various seasons and

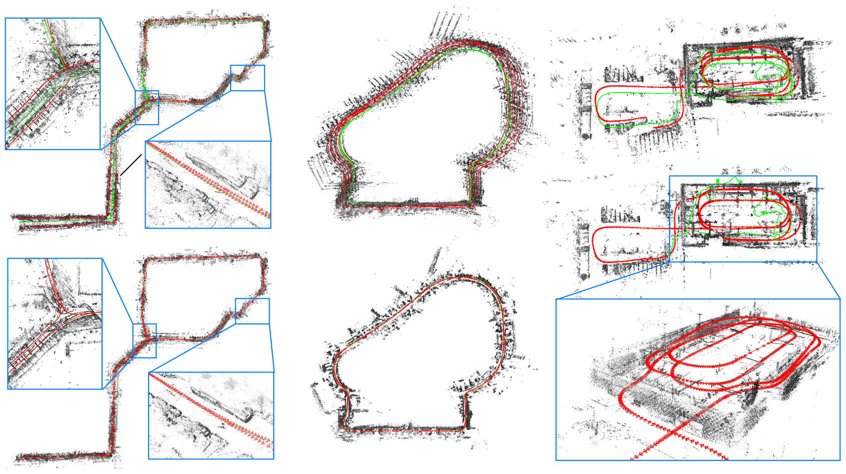


Figure 7.3: **3D models of different scenarios contained in the dataset**. The figure shows a loop around an industrial area (left), multiple loops around an area with high buildings (middle), and a stretch recorded in a multi-level parking garage (right). The green lines encode the GNSS trajectories, and the red lines encode the VIO trajectories. Top: shows the trajectories before the fusion using pose graph optimization. Bottom: shows the result after the pose graph optimization. Note that after the pose graph optimization the reference trajectory is well aligned.

weather conditions as well as from different times of the day as shown in the bottom part of Figure 7.1.

7.4.1 *Ground Truth Validation*

The top part of Figure 7.1 shows two overlaid point clouds from different runs across the same scene. Note that despite the weather and seasonal differences the point clouds align very well. This shows that our reference poses are indeed very accurate. Furthermore, a qualitative assessment of the point-to-point correspondences is shown in Figure 7.5. The figure shows a subset of very accurate pixel-wise correspondences across different seasons (*autumn*/*winter*) in the top and different illumination conditions (*sunny*/*night*) in the bottom. These point-to-point correspondences are a result of our up-to centimeter-accurate global reference poses and are obtained in a completely self-supervised manner. This makes them suitable as training pairs for learning-based algorithms. Recently, there has been an increasing demand for pixel-wise cross-season correspondences which are needed to learn dense feature descriptors [33, 101, 124]. However, there is still a lack of datasets to satisfy this demand. The KITTI [40] dataset does not provide cross-season data. The Oxford RobotCar Dataset [82] provides cross-seasons data, however, since the ground truth is not accurate enough, the paper does not recommend benchmarking localization and mapping approaches.

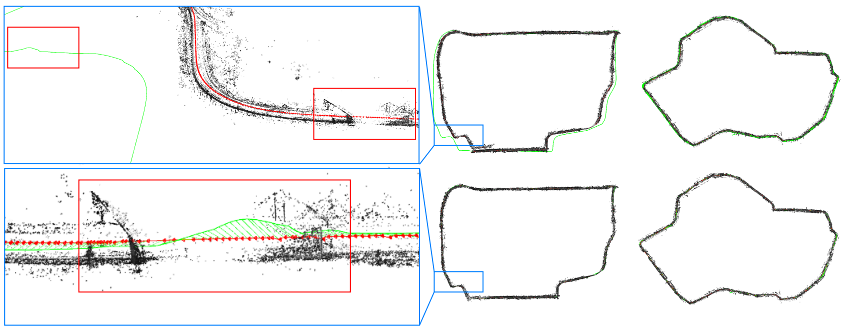


Figure 7.4: **Reference poses validation.** This figure shows two additional 3D models of the scenarios collected. Note that these two sequences are quite large (more than 10 km and 6 km, respectively). Top: before the fusion using pose graph optimization. Bottom: results after optimization. The green lines encode the GNSS trajectories, the red lines show the VIO trajectories (before fusion) and the fused trajectories (after fusion). The left part of the figure shows a zoomed-in view of a tunnel, where the GNSS signal becomes very noisy as highlighted in the red boxes. Besides, due to the large size of the sequence, the accumulated tracking error leads to a significant deviation of the VIO trajectory from the GNSS recordings. Our pose graph optimization, by depending globally on GNSS positions and locally on VIO relative poses, successfully eliminates global VIO drifts and local GNSS positioning flaws.

Recently, RobotCar Seasons [115] was proposed to overcome the inaccuracy of the provided ground truth. However, similar to the authors of [124], we found that it is still challenging to obtain accurate cross-seasonal pixel-wise matches due to pose inconsistencies. Furthermore, this dataset only provides images captured from three synchronized cameras mounted on a car, pointing to the rear-left, rear, and rear-right, respectively. Moreover, the size of the dataset is quite small and a significant portion of it suffers from strong motion blur and low image quality.

To the best of our knowledge, our dataset is the first that exhibits accurate cross-season reference poses for the AD domain.

7.5 TASKS

This section describes the different tasks of the dataset. The provided globally consistent 6DoF reference poses for diverse conditions will be valuable to develop and improve the state-of-the-art for different SLAM-related tasks. Here the major tasks are robust VO, global place recognition, and map-based re-localization tracking.

In the following, we will present the different subtasks for our dataset.

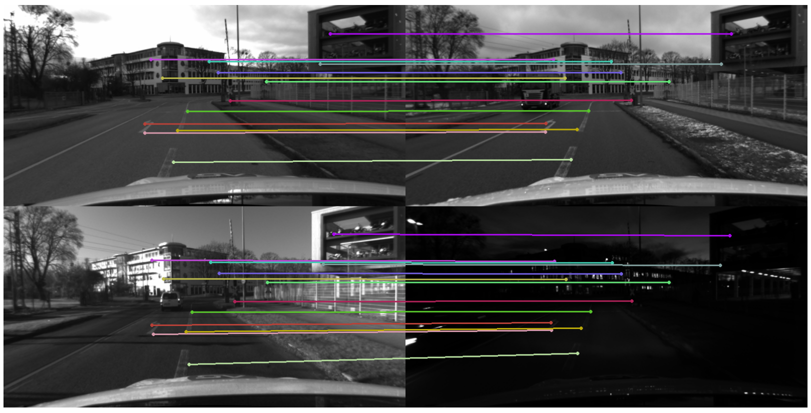


Figure 7.5: **Accurate pixel-wise correspondences, making cross-seasonal training possible.** Qualitative assessment of the accuracy of our data collection and geometric reconstruction method for a sample of four different conditions (from top left in clockwise order: *overcast*, *snowy*, *night*, *sunny*) across the same scene. Each same colored point in the four images corresponds to the same geometric point in the world. The cameras corresponding to these images have different poses in the global frame of reference. Please note that the points are not matched but rather a result of our accurate reference poses and geometric reconstruction. This way we are capable of obtaining sub-pixel level accuracy. On average we get more than 1000 of those correspondences per image pair.

7.5.1 *Visual Odometry in Different Weather Conditions*

VO aims to accurately estimate the 6DoF pose for every frame relative to a starting position. To benchmark the task of VO there already exist various datasets [37, 41, 127]. All of these existing datasets consist of sequences recorded at rather homogeneous conditions (indoors, or sunny/overcast outdoor conditions). However, especially methods developed for AD use cases must perform robustly under almost any condition. We believe that the proposed dataset will contribute to improving the performance of VO under diverse weather and lighting conditions in an automotive environment. Therefore, instead of replacing existing benchmarks and datasets, we aim to provide an extension that is more focused on challenging conditions in AD. As we provide frame-wise accurate poses for large portions of the sequences, metrics well known from other benchmarks like absolute trajectory error (ATE) or relative pose error (RPE) [41, 127] are also applicable to our data.

7.5.2 *Global Place Recognition*

Global place recognition refers to the task of retrieving the most similar database image given a query image [78]. In order to improve the searching efficiency and the robustness against different weather

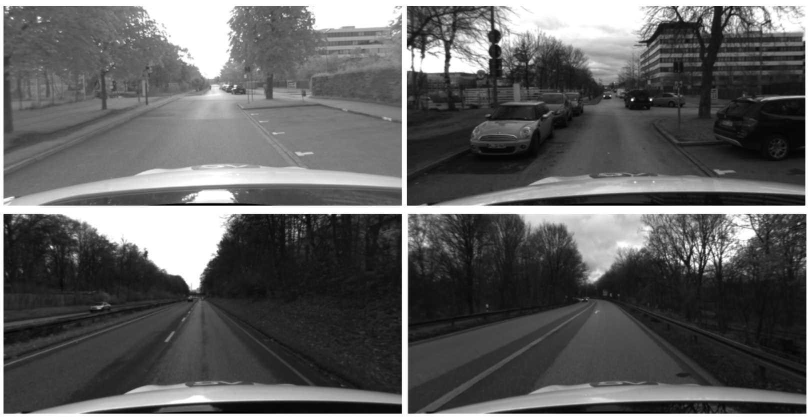


Figure 7.6: **Challenging scenes for global place recognition.** Top: two pictures share the same location with different appearances. Bottom: two pictures have similar appearances but are taken at different locations.

conditions, tremendous progress on global descriptors [11, 13, 39, 57] has been seen. For the re-localization pipeline, visual place recognition serves as the initialization step to the downstream local pose refinement by providing the most similar database images as well as the corresponding global poses. Due to the advent of deep neural networks [47, 67, 122, 130], methods aggregating deep image features are proposed and have shown advantages over classical methods [12, 45, 99, 135].

The proposed dataset is challenging for global place recognition since it contains not only cross-season images that have different appearances but share a similar geographical location but also intra-season images which share similar appearances but with different locations. Figure 7.6 depicts example pairs of these scenarios. We suggest following the standard metric widely used for global place recognition [12, 13, 45, 114].

7.5.3 *Map-Based Re-Localization Tracking*

Map-based re-localization tracking [3] refers to the task of locally refining the 6DoF pose between reference images from a pre-built reference map and images from a query sequence. In contrast to wide-baseline stereo matching, for re-localization tracking, it is also possible to utilize the sequential information of the sequence. This allows us to estimate depth values by running a standard VO method. Those depth estimates can then be used to improve the tracking of the individual re-localization candidates.

In this task, we assume to know the mapping between reference and query samples. This allows us to evaluate the performance of local feature descriptor methods in isolation. In practice, this mapping can be found using image retrieval techniques like NetVLAD [12] as described in Section 7.5.2 or by aligning the point clouds from the reference and query sequences [115], respectively.

Accurately re-localizing in a pre-built map is a challenging problem, especially if the visual appearance of the query sequence significantly differs from the base map. This makes it extremely difficult especially for vision-based systems since the localization accuracy is often limited by the discriminative power of feature descriptors. Our proposed dataset allows us to evaluate re-localization tracking across multiple types of weather conditions and diverse scenes, ranging from urban to countryside driving. Furthermore, our up-to centimeter-accurate ground truth allows us to create diverse and challenging re-localization tracking candidates with an increased level of difficulty. By being able to precisely change the re-localization distances and the camera orientation between the reference and query samples, we can generate more challenging scenarios. This allows us to determine the limitations and robustness of current state-of-the-art methods.

7.6 CONCLUSION

We have presented a cross-season dataset for the purpose of multi-weather SLAM, global visual localization, and local map-based re-localization tracking for AD applications. Compared to other datasets, like KITTI [40] or Oxford RobotCar [82], the presented dataset provides diversity in both multiplicities of scenarios and environmental conditions. Furthermore, based on the fusion of direct stereo VIO and RTK-GNSS we are able to provide up-to centimeter-accurate reference poses as well as highly accurate cross-sequence correspondences. One drawback of the dataset is that the accuracy of the reference poses can only be guaranteed in environments with good GNSS receptions. However, due to the low drift of the stereo VIO system, the obtained reference poses are also very accurate in GNSS-denied environments, e.g. tunnels, garages, or urban canyons.

We believe that this dataset will help the research community to further understand the limitations and challenges of long-term visual SLAM in changing conditions and environments and will contribute to advancing the state-of-the-art. To the best of our knowledge, ours is the first large-scale dataset for AD providing cross-seasonal accurate pixel-wise correspondences for diverse scenarios. This will help to vastly increase robustness against environmental changes for deep learning methods. The dataset is made publicly available to facilitate further research.

8 LM-RELOC: LEVENBERG-MARQUARDT BASED DIRECT VISUAL RELOCALIZATION

Authors: Lukas von Stumberg*[1,2]
Patrick Wenzel*[1,2]
Nan Yang[1,2]
Daniel Cremers[1,2]

* Equal contribution
[1] Technical University of Munich
[2] Artisense

Conference: *International Conference on 3D Vision (3DV), 2020*
DOI: 10.1109/3DV50981.2020.00107

Abstract: We present LM-Reloc – a novel approach for visual relocalization based on direct image alignment. In contrast to prior works that tackle the problem with a feature-based formulation, the proposed method does not rely on feature matching and RANSAC. Hence, the method can utilize not only corners but any region of the image with gradients. In particular, we propose a loss formulation inspired by the classical Levenberg-Marquardt algorithm to train LM-Net. The learned features significantly improve the robustness of direct image alignment, especially for relocalization across different conditions. To further improve the robustness of LM-Net against large image baselines, we propose a pose estimation network, CorrPoseNet, which regresses the relative pose to bootstrap the direct image alignment. Evaluations on the CARLA and Oxford RobotCar relocalization tracking benchmark show that our approach delivers more accurate results than previous state-of-the-art methods while being comparable in terms of robustness.

Revised layout and minor adaptations. Accepted version of original publication [4] and detailed disclaimer are included in Appendix A.5.

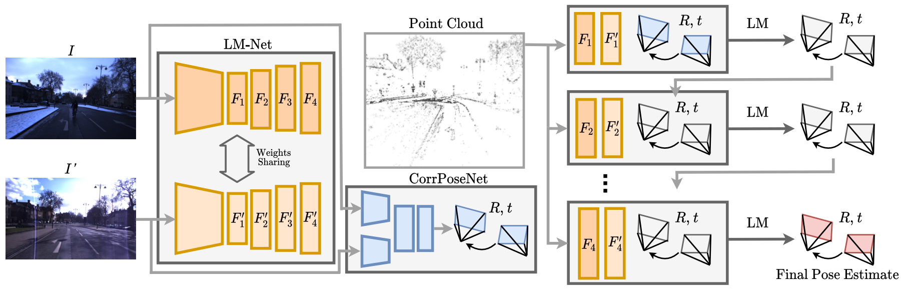


Figure 8.1: We propose LM-Reloc – a novel approach for visual relocalization based on direct image alignment. It consists of two deep neural networks: LM-Net, an encoder-decoder network for learning dense visual descriptors and a CorrPoseNet to bootstrap the direct image alignment. The final 6DoF relative pose estimate between image I and I′ is obtained in a coarse-to-fine pyramid scheme leveraging the learned feature maps. The initialization for the direct image alignment is obtained by the CorrPoseNet.

8.1 INTRODUCTION

Map-based relocalization, that is, to localize a camera within a prebuilt reference map, is becoming more and more important for robotics [27], autonomous driving [16, 92] and AR/VR [109]. Sequential-based approaches, which leverage the temporal structure of the scene provide more stable pose estimations and also deliver the positions in global coordinates compared to single image-based localization methods. The map is usually generated by either using LiDAR or visual Simultaneous Localization and Mapping (vSLAM) solutions. In this paper, we consider vSLAM maps due to the lower-cost visual sensors and the richer semantic information from the images. Feature-based methods [29, 66, 87, 88] and direct methods [8, 34, 35, 64] are two main lines of research for vSLAM.

Once a map is available, the problem of relocalizing within this map at any later point in time requires to deal with long-term changes in the environment. This makes a centimeter-accurate global localization challenging, especially in the presence of drastic lighting and appearance changes in the scene. For this task, feature-based methods are the most commonly used approaches to estimate the ego pose and its orientation. This is mainly due to the advantage that features are more robust against changes in lighting/illumination in the scene.

However, feature-based methods can only utilize keypoints that have to be matched across the images before the pose estimation begins. Thus they ignore large parts of the available information. Direct methods, in contrast, can take advantage of all image regions with sufficient gradients and as a result, are known to be more accurate on visual odometry benchmarks [34, 142, 149].

In this paper, we propose LM-Reloc, which applies direct techniques to the task of relocalization. LM-Reloc consists of LM-Net, CorrPoseNet, and a non-linear optimizer, which work seamlessly together to deliver reliable pose estimation without RANSAC and feature matching. In particular, we derive a loss formulation, which is specifically designed to work well with the Levenberg-Marquardt (LM) algorithm [73, 83]. We use a deep neural network, LM-Net, to train descriptors that are being fed to the direct image alignment algorithm. Using these features results in better robustness against bad initializations, large baselines, and against illumination changes.

While the robustness improvements gained with our loss formulation are sufficient in many cases, for very large baselines or strong rotations, some initialization can still be necessary. To this end, we propose a pose estimation network. Based on two images it directly regresses the 6DoF pose, which we utilize as initialization for LM-Net. The CorrPoseNet contains a correlation layer as proposed in [104], which ensures that the network can handle large displacements. The proposed CorrPoseNet displays a lot of synergies with LM-Net. Despite being quite robust, the predictions of the CorrPoseNet are not very accurate. Thus it is best used in conjunction with our LM-Net, resulting in very robust and accurate pose estimates.

We evaluate our approach on the relocalization tracking benchmark from [3], which contains scenes simulated using CARLA [31], as well as sequences from the Oxford RobotCar dataset [82]. Our LM-Net shows superior accuracy especially in terms of rotation while being competitive in terms of robustness.

We summarize our main contributions:

- LM-Reloc, a novel pipeline for visual relocalization based on direct image alignment, which consists of LM-Net, CorrPoseNet, and a non-linear optimizer.
- A novel loss formulation together with a point sampling strategy that is used to train LM-Net such that the resulting feature descriptors are optimally suited to work with the LM algorithm.
- Extensive evaluations on the CARLA and Oxford RobotCar relocalization tracking benchmark which show that the proposed approach achieves state-of-the-art relocalization accuracy without relying on feature matching or RANSAC.

8.2 RELATED WORK

In this section, we review the main topics that are closely related to our work, including direct methods for visual localization and feature-based visual localization methods.

Direct methods for visual localization. In recent years, direct methods [34, 35, 64] for SLAM and visual odometry have seen a great progress. Unlike feature-based methods [29, 66, 87, 88] which firstly extracts keypoints as well as the corresponding descriptors, and then minimize the geometric errors, direct methods minimize the energy function based on the photometric constancy assumption without performing feature matching or RANSAC. By utilizing more points from the images, direct methods show higher accuracy than feature-based methods [142]. However, classical direct methods show lower robustness than feature-based methods when the photometric constancy assumption is violated due to, e.g. the lighting and weather changes which are typical for long-term localization [115]. In [9] and [95], the authors propose to use the handcrafted features to improve the robustness of direct methods against low light or global appearance changes. Some recent works [19, 80, 3] address the issue by using learned features from deep neural networks [70]. In [19] they train deep features using a Hinge-Loss based on the Lucas-Kanade method, however, in contrast to us, they estimate the optical flow instead of applying the features to the task of relocalization. The most related work to ours is GN-Net [3] which proposes a Gauss-Newton loss to learn deep features. By performing direct image alignment on the learned features, GN-Net can deliver reliable pose estimation between the images taken from different weather or season conditions. The proposed LM-Net further derives the loss formulation based on Levenberg-Marquardt to improve the robustness against bad initialization compared to the Gauss-Newton method. Inspired by D3VO [141], LM-Reloc also proposes a relative pose estimation network with a correlation layer [104] to regress a pose estimate which is used as the initialization for the optimization.

Feature-based visual localization. Most approaches for relocalization utilize feature detectors and descriptors, which can either be handcrafted, such as SIFT [77] or ORB [108], or especially in the context of drastic lighting and appearance changes can be learned. Recently, many descriptor learning methods have been proposed which follow a *detect-and-describe* paradigm, e.g. SuperPoint [30], D2-Net [33], or R2D2 [101]. Moreover, SuperGlue [112], a learning-based alternative to the matching step of feature-based methods has been proposed and yields significant performance improvements. For a complete relocalization pipeline, the local pose refinement part has to be preceded by finding the closest image in a database given a query [12]. While some approaches [110, 111, 131] address the joint problem, in this work, we decouple these two tasks and only focus on the pose refinement part.

8.3 METHOD

In this work, we address the problem of computing the 6DoF pose $\xi \in \mathbf{SE}(3)$ between two given images I and I'. Furthermore, we assume that depths for a sparse set of points P are available, e.g. by running a direct visual SLAM system such as DSO [34].

The overall pipeline of our approach is shown in Figure 8.1. It is composed of LM-Net, CorrPoseNet, and a non-linear optimizer using the LM algorithm. LM-Net is trained with a novel loss formulation designed to learn feature descriptors optimally suited for the LM algorithm. The encoder-decoder architecture takes as input a reference image I as well as a target image I'. The network is trained end-to-end and will produce multi-scale feature maps F_l and F'_l, where $l = 1, 2, 3, 4$ denotes the different levels of the feature pyramid. In order to obtain an initial pose estimate for the non-linear optimization, we propose CorrPoseNet, which takes I and I' as the inputs and regresses their relative pose. Finally, the multi-scale feature maps together with the depths obtained from DSO [34] form the non-linear energy function which is minimized using LM algorithm in a coarse-to-fine manner to obtain the final relative pose estimate. In the following, we will describe the individual components of our approach in more detail.

8.3.1 *Direct Image Alignment with Levenberg-Marquardt*

In order to optimize the pose $\boldsymbol{\xi}$ (consisting of rotation matrix $\mathbf{R}$ and translation $\mathbf{t}$), we minimize the feature-metric error:

$$E(\boldsymbol{\xi}) = \sum_{\mathbf{p} \in P} \left\| F'_l(\mathbf{p}') - F_l(\mathbf{p}) \right\|_\gamma, \tag{8.1}$$

where $\|\cdot\|_\gamma$ is the Huber norm and $\mathbf{p}'$ is the point projected onto the target image I' using the depths and the pose:

$$\mathbf{p}' = \Pi\left(\mathbf{R}\Pi^{-1}(\mathbf{p}, d_{\mathbf{p}}) + \mathbf{t}\right). \tag{8.2}$$

This energy function is first minimized on the coarsest pyramid level 1, whose feature maps F_1 have a size of $(w/8, h/8)$, yielding a rough pose estimate. The estimate is refined by further minimizing the energy function on the subsequent pyramid levels 2, 3, and 4, where F_4 has the size of the original image (w, h). In the following, we provide details of the minimization performed in every level and for simplicity we will denote F_l as F from now on.

Minimization is performed using the Levenberg-Marquardt algorithm. In each iteration we compute the update $\boldsymbol{\delta} \in \mathbb{R}^6$ in the Lie algebra $\mathfrak{se}(3)$ as follows: Using the residual vector $\mathbf{r} \in \mathbb{R}^n$, the Huber weight matrix $\mathbf{W} \in \mathbb{R}^{n \times n}$, and the Jacobian of the residual vector with respect to the pose $\mathbf{J} \in \mathbb{R}^{n \times 6}$, we compute the Gauss-Newton system:

$$\mathbf{H} = \mathbf{J}^\top \mathbf{W} \mathbf{J} \text{ and } \mathbf{b} = -\mathbf{J}^\top \mathbf{W} \mathbf{r}. \tag{8.3}$$

The damped system can be obtained with either Levenberg's formula [73]:

$$\mathbf{H}' = \mathbf{H} + \lambda \mathbf{I}_n \tag{8.4}$$

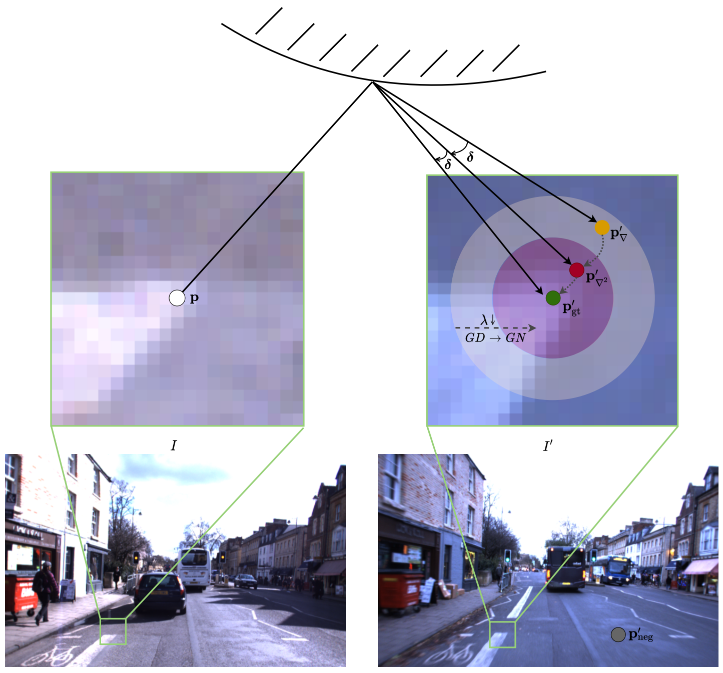


Figure 8.2: Visualization of the typical behavior of direct image alignment with Levenberg-Marquardt. Initially, the projected point position (orange point, $\mathbf{p}'_{\nabla}$) is far away from the correct solution (green point, $\mathbf{p}'_{\text{gt}}$), and λ is large, yielding an update step similar to gradient descent. After some iterations the projected point position gets closer to the optimum (red point, $\mathbf{p}'_{\nabla^2}$) and at the same time λ will get smaller, leading to an update step similar to the Gauss-Newton algorithm. This is the intuition behind our point sampling strategy, where we utilize the ground-truth correspondence $\mathbf{p}'_{\text{gt}}$ for Equation (8.7), a negative $\mathbf{p}'_{\text{neg}}$ sampled across the whole image for Equation (8.8), a negative $\mathbf{p}'_{\nabla}$ sampled in a far vicinity for Equation (8.12), and a negative $\mathbf{p}'_{\nabla^2}$ sampled in a close vicinity for Equation (8.14).

or the Marquardt's formula [83]:

$$\mathbf{H}' = \mathbf{H} + \lambda\,\text{diag}(\mathbf{H}) \tag{8.5}$$

depending on the specific application.

The update $\boldsymbol{\delta}$ and the pose $\boldsymbol{\xi}^i$ in the iteration i are computed as:

$$\boldsymbol{\delta} = \mathbf{H}'^{-1}\mathbf{b} \text{ and } \boldsymbol{\xi}^i = \boldsymbol{\delta} \boxplus \boldsymbol{\xi}^{i-1}, \tag{8.6}$$

where $\boxplus : \mathfrak{se}(3) \times \mathbf{SE}(3) \rightarrow \mathbf{SE}(3)$ is defined as in [34].

The parameter λ can be seen as an interpolation factor between gradient descent and the Gauss-Newton algorithm. When λ is high the method behaves like gradient descent with a small step size, and when it is low it is equivalent to the Gauss-Newton algorithm. In practice, we start with a relatively large λ and multiply it by 0.5 after a successful iteration, and by 4 after a failed iteration [34].

Figure 8.2 shows the typical behavior of the algorithm. In the beginning, the initial pose is inaccurate, resulting in projected point positions, which are a couple of pixels away from the correct location. λ will be high meaning that the algorithm will behave similarly to gradient descent. After a couple of iterations, the pose got more accurate, and the projected points are in a closer vicinity to the correct location. By now, λ has probably decreased, so the algorithm will behave more similarly to the Gauss-Newton algorithm. Now we expect the algorithm to converge quickly.

8.3.2 *Loss Formulation for Levenberg-Marquardt*

The key contribution of this work is LM-Net which provides feature maps F that improve the convergence behavior of the LM algorithm and, in the meantime, are invariant to different conditions. We train our network in a Siamese fashion based on ground-truth pixel correspondences.

In this section, $\mathbf{p}$ denotes a reference point (located on image I) and the ground-truth correspondence (located on image I') is $\mathbf{p}'_{gt}$. For the loss functions explained below we further categorize $\mathbf{p}'$ into $\mathbf{p}'_{neg}$, $\mathbf{p}'_{\nabla}$, and $\mathbf{p}'_{\nabla^2}$, which is realized by using different negative correspondence sampling. Our loss formulation is inspired by the typical behavior of the Levenberg-Marquardt algorithm explained in the previous section (see Figure 8.2). For a point, we distinguish four cases which can happen during the optimization:

1. The point is at the correct location ($\mathbf{p}'_{gt}$).
2. The point is an outlier ($\mathbf{p}'_{neg}$).
3. The point is relatively far from the correct solution ($\mathbf{p}'_{\nabla}$).
4. The point is very close to the correct solution ($\mathbf{p}'_{\nabla^2}$).

In the following we will derive a loss function for each of the 4 cases:

1. The point is already at the correct location. In this case, we would like the residual to be as small as possible, in the best case 0.

$$E_{pos} = \left\| F'(\mathbf{p}'_{gt}) - F(\mathbf{p}) \right\|^2 \tag{8.7}$$

2. The point is an outlier or the pose estimate is completely wrong. In this case, the projected point position can be at a completely different location than the correct correspondence. In this scenario, we would like the residual of this pixel to be very large to reflect this and potentially reject a wrong update. To enforce this property we sample a negative correspondences $\mathbf{p}'_{\text{neg}}$ uniformly across the whole image, and compute

$$E_{\text{neg}} = \max\left(M - \left\| F'(\mathbf{p}'_{\text{neg}}) - F(\mathbf{p}) \right\|^2, 0\right) \tag{8.8}$$

where M is the margin of how large we would like the energy of a wrong correspondence to be. In practice, we set it to 1.
3. The predicted pose is relatively far away from the optimum, meaning that the projected point position will be a couple of pixels away from the correct location. As this typically happens during the beginning of the optimization we assume that λ will be relatively large and the algorithm behaves similarly to gradient descent. In this case, we want that the gradient of this point is oriented in the direction of the correct solution so that the point has a positive influence on the update step.

For computing a loss function to enforce this property we sample a random negative correspondence $\mathbf{p}'_{\nabla}$ in a relatively large vicinity around the correct solution (in our experiments we use 5 pixels distance). Starting from this negative correspondence $\mathbf{p}'_{\nabla}$ we first compute the 2×2 Gauss-Newton system for this individual point, similarly to how it is done for optical flow estimation using Lucas-Kanade:

$$\mathbf{r}_{\mathbf{p}}(\mathbf{p}, \mathbf{p}'_{\nabla}) = \mathbf{F}'(\mathbf{p}'_{\nabla}) - \mathbf{F}(\mathbf{p}) \tag{8.9}$$

$$\mathbf{J}_{\mathbf{p}} = \frac{d\mathbf{F}'(\mathbf{p}'_{\nabla})}{d\mathbf{p}'_{\nabla}} \text{ and } \mathbf{H}_{\mathbf{p}} = \mathbf{J}_{\mathbf{p}}^{\top}\mathbf{J}_{\mathbf{p}} \text{ and } \mathbf{b}_{\mathbf{p}} = \mathbf{J}_{\mathbf{p}}^{\top}\mathbf{r}_{\mathbf{p}} \tag{8.10}$$

We compute the damped system using a relatively large fixed λ_f, as well as the optical flow step[1]:

$$\mathbf{H}'_{\mathbf{p}} = \mathbf{H}_{\mathbf{p}} + \lambda_f \mathbf{I}_n \text{ and } \mathbf{p}'_{\text{after}} = \mathbf{p}'_{\nabla} + \mathbf{H}'^{-1}_{\mathbf{p}}\mathbf{b}_{\mathbf{p}}. \tag{8.11}$$

In order for this point to have a useful contribution to the direct image alignment, this update step should move in the correct direction by at least δ. We enforce this using a Gradient-Descent loss function which is small only if the distance to the correct correspondence *after* the update is smaller than before the update:

$$E_{\text{GD}} = \max\left(\left\|\mathbf{p}'_{\text{after}} - \mathbf{p}'_{\text{gt}}\right\|^2 - \left\|\mathbf{p}'_{\nabla} - \mathbf{p}'_{\text{gt}}\right\|^2 + \delta, 0\right) \tag{8.12}$$

In practice, we choose $\lambda_f = 2.0$ and $\delta = 0.1$.
4. The predicted pose is very close to the optimum, yielding a projected point position in very close proximity of the correct correspondence, and typically λ will be very small, so the update will mostly be a Gauss-Newton step. In this case we would like the algorithm to converge as quickly as possible, with subpixel accuracy. We enforce

1 Here we use Equation (8.4) instead of Equation (8.5) since we find it more stable for training LM-Net.

this using the Gauss-Newton loss [3]. To compute it we first sample a random negative correspondence $\mathbf{p}'_{\nabla^2}$ in a 1-pixel vicinity around the correct location. Then we use Equations (8.9) and (8.10), replacing $\mathbf{p}'_{\nabla}$ with $\mathbf{p}'_{\nabla^2}$ to obtain the Gauss-Newton system formed by $\mathbf{H}_\mathbf{p}$ and $\mathbf{b}_\mathbf{p}$. We compute the updated pixel location:

$$\mathbf{p}'_{\text{after}} = \mathbf{p}'_{\nabla^2} + (\mathbf{H}_\mathbf{p} + \epsilon \mathbf{I}_n)^{-1} \mathbf{b}_\mathbf{p} \tag{8.13}$$

Note that in contrast to the computation of the LM-Loss (Equation (8.12)), in this case, ϵ is just added to ensure invertibility and therefore ϵ is much smaller than the λ_f used above. The Gauss-Newton loss is computed with:

$$\begin{aligned} E_{GN} = \frac{1}{2} \left(\mathbf{p}'_{\text{after}} - \mathbf{p}'_{\text{gt}}\right)^\top \mathbf{H}_\mathbf{p} \left(\mathbf{p}'_{\text{after}} - \mathbf{p}'_{\text{gt}}\right) \\ + \log(2\pi) - \frac{1}{2} \log\left(|\mathbf{H}_\mathbf{p}|\right) \end{aligned} \tag{8.14}$$

Note how all our 4 loss components use a different way to sample the involved points, depicted also in Figure 8.2. With the derivation above we argue that each loss component is important to achieve optimal performance and we demonstrate this in the results section. Note that the Gauss-Newton systems computed for the GD-Loss and the GN-Loss are very relevant for the application of direct image alignment. In fact, the full Gauss-Newton system containing all points (Equation (8.3)), can be computed from these individual Gauss-Newton systems (Equation (8.10)) by simply summing them up and multiplying them with the derivative with respect to the pose [3].

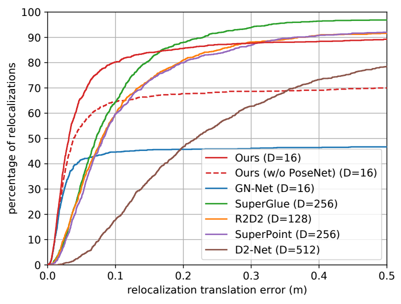
8.3.3 *CorrPoseNet*

In order to deal with the large baselines between the images, we propose CorrPoseNet to regress the relative pose between two images I and I', which serves as the initialization of LM optimization. As our network shall work even in cases of large baselines and strong rotations, we utilize the correlation layer proposed in [104] which is known to boost the performance of affine image transformation and optical flow [84] estimation for large displacements, but has not been applied to pose estimation before.

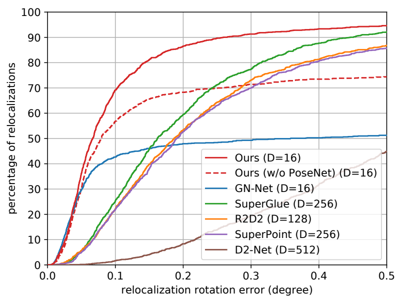
Our network first computes deep features $\mathbf{f}_{\text{corr}}, \mathbf{f}'_{\text{corr}} \in \mathbb{R}^{h \times w \times c}$ from both images individually using multiple strided convolutions with ReLU activations in between. Then the correlation layer correlates each pixel from the normalized source features with each pixel from the normalized target features yielding the correlation map $\mathbf{c} \in \mathbb{R}^{h \times w \times (h \times w)}$:

$$\mathbf{c}\left(i, j, \left(i', j'\right)\right) = \mathbf{f}_{\text{corr}}\left(i, j\right)^\top \mathbf{f}'_{\text{corr}}\left(i', j'\right) \tag{8.15}$$

The correlation map is then normalized in the channel dimension and fed into 2 convolutional layers each followed by batch norm and ReLU.



(a) Translation error.



(b) Rotation error.

Figure 8.3: Results on the CARLA relocalization tracking benchmark test data [3]. For each error threshold, we show the percentage of relocalizations (cumulative error plot) for LM-Reloc (ours) and other state-of-the-art methods. Compared to the indirect methods our approach exhibits significantly better accuracy in both translation and rotation, while having a similar robustness. Compared to GN-Net, the novel loss formulation (see red dashed line), and the CorrPoseNet (see red line) both boost the robustness. D is the feature dimensionality.

Finally, we regress the Euler angle $\mathbf{r}^{\text{euler}}$ and translation $\mathbf{t}$ using a fully connected layer. More details on the architecture are shown in the supplementary material.

We train CorrPoseNet from scratch with image pairs and ground truth poses $\mathbf{r}_{\text{gt}}^{\text{euler}}, \mathbf{t}_{\text{gt}}$. We utilize an L2-loss working directly on Euler angles and translation:

$$E = \left\| \mathbf{t} - \mathbf{t}_{\text{gt}} \right\|_2 + \lambda \left\| \mathbf{r}^{\text{euler}} - \mathbf{r}_{\text{gt}}^{\text{euler}} \right\|_2, \tag{8.16}$$

where λ is the weight, which we set to 10 in practice.

As the distribution of ground truth poses in the Oxford training data is limited we apply the following data augmentation. We first generate dense depths for all training images using a state-of-the-art dense stereo matching algorithm [146]. The resulting depths are then used to warp the images to a different pose sampled from a uniform distribution. In detail, we first warp the depth image to the random target pose, then inpaint the depth image using the OpenCV implementation of Navier Stokes, and finally warp our image to the target pose using this depth map. Note that the dense depths are only necessary for training, not for evaluation. We show an ablation study on the usage of correlation layers and the proposed data augmentation in the supplementary material.

8.4 EXPERIMENTS

We evaluate our method on the relocalization tracking benchmark proposed in [3], which contains images created with the CARLA simulator [31], and scenes from the Oxford RobotCar dataset [82].

We train our method on the respective datasets from scratch. LM-Net is trained using the Adam optimizer with a learning rate of 10^{-6} and for CorrPoseNet we use a learning rate of 10^{-4}. For both networks, we choose hyperparameters and epochs based on the results on the validation data. Our networks use the same hyperparameters for all experiments except where stated otherwise; the direct image alignment code is slightly adapted for Oxford RobotCar, mainly to improve performance when the ego-vehicle is standing still.

As the original relocalization tracking benchmark [3] does not include validation data on Oxford RobotCar we have manually aligned two new sequences, namely *2015-04-17-09-06-25* and *2015-05-19-14-06-38*, and extend the benchmark with these sequences as validation data.

Evaluation metrics: We evaluate the predicted translation $\mathbf{t}_{\text{est}}$ and rotation $\mathbf{R}_{\text{est}}$ against the ground-truth $\mathbf{t}_{\text{gt}}$ and $\mathbf{R}_{\text{gt}}$ according to Equations (8.17) and (8.18).

$$t_\Delta = \left\| \mathbf{t}_{\text{est}} - \mathbf{t}_{\text{gt}} \right\|_2 \tag{8.17}$$

$$R_\Delta = \arccos\left(\frac{\operatorname{trace}\left(\mathbf{R}_{\text{est}}^{-1} \mathbf{R}_{\text{gt}}\right) - 1}{2} \right) \tag{8.18}$$

In this section, we plot the cumulative translation and rotation error until 0.5 m and 0.5°, respectively. For quantitative results, we compute the area under the curve (AUC) of these cumulative curves in percent, which we denote as t_{AUC} for translation and R_{AUC} for rotation from now on.

We evaluate the following direct methods:

Ours: The full LM-Reloc approach consists of CorrPoseNet, LM-Net features and direct image alignment based on Levenberg-Marquardt. The depths used for the image alignment are estimated with the stereo version [136] of DSO [34].

Ours (w/o CorrPoseNet): For a more fair comparison to GN-Net we use identity as initialization for the direct image alignment instead of CorrPoseNet. This enables a direct comparison between the two loss formulations.

GN-Net [3]: In this work, we have also improved the parameters of the direct image alignment pipeline based on DSO [34]. Thus we have re-evaluated GN-Net with this improved pipeline to make the comparison as fair as possible. These re-evaluated results are better than the results computed in the original GN-Net paper.

Baseline methods: Additionally, we evaluate against current state-of-the-art indirect methods, namely SuperGlue [112], R2D2 [101], SuperPoint [30], and D2-Net [33]. For these methods, we estimate the relative pose using the models provided by the authors and the OpenCV implementation of solvePnPRansac. We have tuned the parameters of RANSAC on the validation data and used 1000 iterations and a reprojection error threshold of 3 for all methods. For estimating depth

Table 8.1: This table shows the AUC until 0.5 meters / 0.5 degrees for the relocalization error on the CARLA relocalization tracking benchmark test data. Powered by our novel loss formulation and the combination with CorrPoseNet, LM-Reloc achieves lower rotation and translation errors compared to the state-of-the-art.

Method	t_{AUC}	R_{AUC}
Ours	**80.65**	**77.83**
SuperGlue [112]	78.99	59.31
R2D2 [101]	73.47	54.42
SuperPoint [30]	72.76	53.38
D2-Net [33]	47.62	16.47
Ours (w/o CorrPoseNet)	63.88	61.90
GN-Net [3]	43.72	44.08

values at keypoint locations we use OpenCV stereo matching. It would be possible to achieve a higher accuracy by using SfM and MVS solutions such as COLMAP [117]. However, one important disadvantage of these approaches is, that building a map is rather time-consuming and computationally expensive, whereas all other approaches evaluated on the benchmark [3] are able to create the map close to real-time, enabling applications like long-term loop-closure and map-merging.

8.4.1 *CARLA Relocalization Benchmark*

Figure 8.3 depicts the results on the test data of the CARLA benchmark. For all methods, we show the cumulative error plot for translation in meters and rotation in degrees. It can be seen that our method is more accurate than the state-of-the-art while performing similarly in terms of robustness. We also show the AUC for the two Figures in Table 8.1. Compared to GN-Net it can be seen that our new loss formulation significantly improves the results, even when used without the CorrPoseNet as initialization. The figure conveys that the direct methods (Ours, GN-Net) are more accurate than the evaluated indirect methods.

8.4.2 *Oxford RobotCar Relocalization Benchmark*

We compare to the state-of-the-art indirect methods on the 6 test sequence pairs consisting of the sequences *2015-02-24-12-32-19* (sunny),

Table 8.2: Results on the Oxford RobotCar relocalization tracking benchmark [3]. We compare LM-Net (Ours) against other state-of-the-art methods (SuperGlue, R2D2, SuperPoint, and D2-Net). As can be seen from the results, our method almost consistently outperforms other SOTA approaches in terms of rotation AUC whilst achieving comparable results on translation AUC.

Sequence	Ours		SuperGlue [112]		R2D2 [101]		SuperPoint [30]		D2-Net [33]	
	t_{AUC}	R_{AUC}	t_{AUC}	R_{AUC}	t_{AUC}	R_{AUC}	t_{AUC}	R_{AUC}	t_{AUC}	R_{AUC}
Sunny-Overcast	79.83	**55.48**	**81.01**	52.83	80.86	53.57	78.95	50.03	71.93	39.0
Sunny-Rainy	71.54	**43.7**	**75.58**	40.59	74.84	41.23	69.76	37.12	65.63	27.5
Sunny-Snowy	59.69	**44.06**	**63.57**	43.64	62.92	41.78	60.85	40.02	55.65	30.86
Overcast-Rainy	80.54	**63.7**	79.99	61.64	**81.29**	61.23	80.36	61.56	75.66	51.06
Overcast-Snowy	55.38	47.88	57.67	47.16	**57.68**	**48.41**	55.39	44.96	51.17	34.54
Rainy-Snowy	68.57	**41.67**	69.91	39.87	**71.79**	39.86	67.7	38.05	61.91	27.74

Table 8.3: This table shows the results on the Oxford RobotCar relocalization tracking benchmark test data against GN-Net. Thanks to our LM-based loss formulation we consistently outperform GN-Net on all sequences.

Sequence	Ours (w/o CorrPoseNet)		GN-Net [3]	
	t_{AUC}	R_{AUC}	t_{AUC}	R_{AUC}
Sunny-Overcast	**79.61**	**55.45**	73.53	49.31
Sunny-Rainy	**70.46**	**42.86**	64.58	37.27
Sunny-Snowy	**59.7**	**44.17**	55.27	41.36
Overcast-Rainy	**79.67**	**63.08**	75.72	60.13
Overcast-Snowy	**54.94**	**47.19**	51.34	42.91
Rainy-Snowy	**66.23**	**39.93**	62.63	36.2

2015-03-17-11-08-44 (overcast), *2014-12-05-11-09-10* (rainy), and *2015-02-03-08-45-10* (snowy). In Table 8.2, we show the area under the curve until 0.5 meters / 0.5 degrees for all methods. It can be seen that our method clearly outperforms the state-of-the-art in terms of rotation accuracy, while being competitive in terms of translation error. It should be noted that the ground truth for these sequences was generated using ICP alignment of the 2D-LiDAR data accumulated for 60 meters. We have computed that the average root mean square error of the ICP alignment is 16 centimeters. Therefore, especially the ground-truth translations have limited accuracy. As can be seen from Figure 8.3, the accuracy improvements our method provides are especially visible in the range below 0.15 meters which is hard to measure on this dataset. The rotation error of LiDAR alignment is lower than the translational one, which is why we clearly observe the improvements of our method on the rotations.

In Table 8.3, we compare LM-Net without the CorrPoseNet to GN-Net. Due to our novel loss formulation, LM-Net outperforms the competitor on all sequences significantly.

8.4.3 *Ablation Studies*

We evaluate LM-Net on the CARLA validation data with and without the various losses (Figure 8.4). Compared to a normal contrastive loss, the given loss formulation is a large improvement. As expected, E_{GD} (green line) mainly improves the robustness, whereas E_{GN} (blue line) improves the accuracy. Only when used together (our method), we

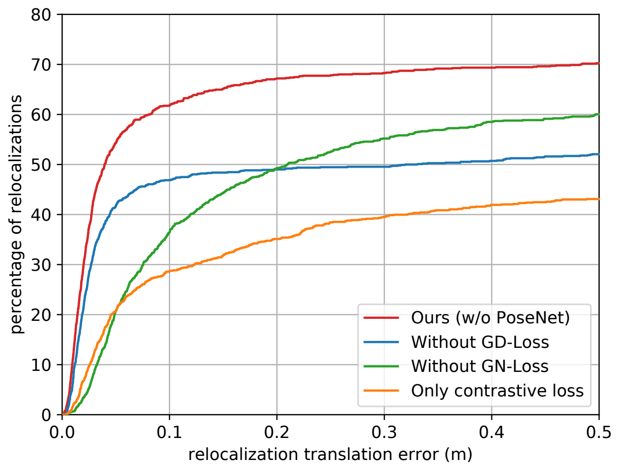


Figure 8.4: This plot shows our ablation study for removing different loss parts on the CARLA relocalization tracking benchmark. Without the GD-loss the achieved robustness is reduced, whereas removing the GN-loss leads to decreased accuracy. Using our full-loss formulation yields a large improvement.

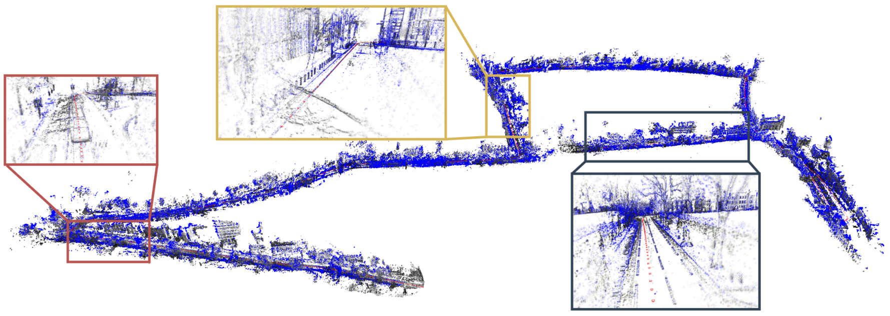


Figure 8.5: This figure shows a point cloud from a sunny reference map (grey points) overlayed with the point cloud from a relocalized snowy sequence (blue points). The well-aligned point clouds demonstrate the high relocalization accuracy of LM-Reloc.

achieve large robustness and large accuracy, confirming our theoretical derivation in Section 8.3.

8.4.4 *Qualitative Results*

To demonstrate the accuracy of our approach in practice, we show qualitative results on the Oxford RobotCar dataset. We track the snowy test sequence *2015-02-03-08-45-10* using Stereo DSO [136] and at the same time perform relocalization against the sunny reference map *2015-02-24-12-32-19*. Relocalization between the current keyframe and the closest map image is performed using LM-Net. Initially, we give the algorithm the first corresponding map image (which would in practice be provided by an image retrieval approach such as NetVLAD [12]). Afterward, we find the closest map image for each keyframe using

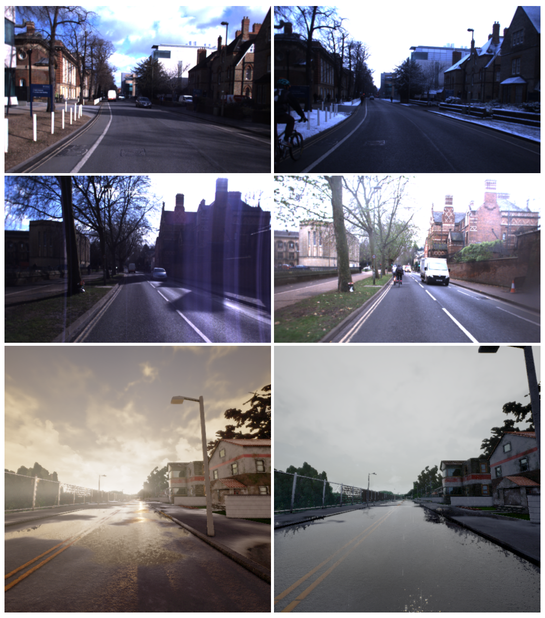


Figure 8.6: Example image pairs from the relocalization tracking benchmark which have been successfully relocalized by LM-Reloc (with an accuracy of better than $10\,\mathrm{cm}$). Top row: Oxford sunny against snowy condition, middle row: Oxford sunny against rainy condition, bottom row: CARLA benchmark.

the previous solution for the transformation between the map and the current SLAM world $\mathbf{T}_{w\_m}$. We visualize the current point cloud (blue) and the point cloud from the map (grey) overlayed using the smoothed $\mathbf{T}_{w\_m}$ (Figure 8.5). The point clouds will align only if the relocalization is accurate. As can be seen in Figure 8.5, the lane markings, poles, and buildings between the reference and query map align well, hence qualitatively showing the high relocalization accuracy of our method. We recommend watching the video at `https://vision.in.tum.de/lm-reloc`. In Figure 8.6 we show example images from the benchmark.

8.5 CONCLUSION

We have presented LM-Reloc as a novel approach for direct visual localization. In order to estimate the relative 6DoF pose between two images from different conditions, our approach performs direct image alignment on the trained features from LM-Net without relying on feature matching or RANSAC. In particular, with the loss function designed seamlessly for the Levenberg-Marquart algorithm, LM-Net provides deep feature maps that coin the characteristics of direct image alignment and are also invariant to changes in lighting and appearance of the scene. The experiments on the CARLA and Oxford RobotCar relocalization tracking benchmark exhibit the state-of-the-art performance of our approach. In addition, the ablation studies also show the effectiveness of the different components of LM-Reloc.

Part III

CONCLUSION AND OUTLOOK

9 SUMMARY

In this thesis, we investigated a very important research topic: vehicle localization and control under challenging conditions. We presented simple, powerful, and effective deep learning-based approaches to get one step further toward robust and reliable computer vision algorithms that work under challenging perceptual conditions and require less labeled data. Leveraging the recent progress in diverse fields (computer vision, deep learning, and robotics), the ideas presented in this thesis enable challenging previously inaccessible tasks.

In the following, we present concluding remarks on the key contributions of this dissertation.

TOWARD ROBUST SENSORIMOTOR CONTROL Sensorimotor control approaches usually need tons of labeled data to reflect flexible navigation in complex and dynamic environments. When performing driving strategies, a key factor is the collection of annotated training data. However, obtaining sufficient and high-quality labeled data is a tedious, time-consuming, and error-prone process. Moreover, to generalize well across changing perceptual conditions, one would need to collect data samples for all invariances present in the real world. This is hardly or not at all possible. To this end, we proposed learning-based frameworks that can use limited labeled training data to transfer knowledge between multiple different domains. We thereby remove the need to have access to annotated training data for all conditions a sensorimotor control approach may encounter. This is achieved by leveraging ground truth data available for one domain and transferring the knowledge across other domains. We show that our approach can achieve equivalent performance with a small subset of the labels as if we had access to labeled data for all domains.

DIRECT VISUAL LOCALIZATION The problem of visual localization is a long-standing challenge in computer vision research and is a core component of technologies such as autonomous driving and augmented or virtual reality. It has been predominantly tackled in an indirect setting that relies on matching sparse keypoints via visual descriptors between a query image and a reference point cloud. Motivated by the exceptional performance of direct SLAM on visual odometry tasks, we advocated a paradigm shift in enabling localization capabilities for direct approaches that are robust under challenging perceptual conditions. To this end, we proposed novel loss formulations based on the properties of direct image alignment to train dense features. These dense feature descriptors are predicted by deep neural networks in a self-supervised fashion and are used to estimate the camera pose by alignment via differentiable optimization. Our approach expands the scope of applicability of direct SLAM methods by enabling re-localization and map merging in changing conditions. We thus believe that our contributions will have a great impact on future visual localization research using direct objectives for dense image alignment.

4SEASONS DATASET In recent years, research on visual SLAM and long-term localization has made tremendous strides. An important reason is the availability of real-world datasets tailored for benchmarking these methods. However, most algorithms are only evaluated under a particular environmental condition or in a specific scenario. This might lead to the development of methods that work well in a particular setting and do not necessarily generalize to all invariances present in the real world. It is, therefore, crucial to develop methods that are robust under all environmental conditions. To this end, we proposed a large-scale real-world benchmark dataset to evaluate the performance of visual SLAM and long-term localization in changing conditions and environments. We believe that our dataset enables the research community to further understand the limitations of current approaches and will thereby help to advance the state-of-the-art.

To sum up, in this thesis, we explored techniques for vehicle localization and control under challenging conditions and made new research advances for several challenging computer vision tasks. While this dissertation only provides a small step toward robust vision systems in all environmental conditions, we believe that future research will benefit from the presented contributions.

10 FUTURE RESEARCH

The techniques presented in this thesis contribute to the vision of robust perception for all conditions and enable challenging, previously inaccessible tasks. While the work in this dissertation is a promising step toward a more powerful embodied vision, there are still many open challenges. In the following, we describe several future research avenues to help achieve this goal.

LEARNING WITHOUT HUMAN-LABELED DATA While the proposed techniques are providing solutions that require less annotated data, we are still far from having equally good-performing algorithms that can learn entirely without human-labeled data. This suggests that further research is needed in acquiring data that needs less manual human interaction for the labeling process. Moreover, the development of learning strategies that can learn in an unsupervised or self-supervised setting needs to be accelerated. Especially in the context of sensorimotor control, it is almost impossible to collect large-scale real-world datasets with labels exhibiting all possible scenarios.

SIM2REAL TRANSFER Most of the deep learning-based methods presented in this thesis require high volumes of training data to achieve good accuracy. However, real-world data acquisition is not only time-consuming and expensive, but it also raises privacy concerns if the data has to be collected in crowded public environments. A promising approach to overcome the dependency on real-world datasets is the *Sim2Real* transfer concept, i.e. to train models in simulation or with synthetic data and deploy them on real sensors. Modern simulation environments provide the possibility to generate large-scale automatically labeled datasets in minutes, compared to the complex process of collecting and labeling data with real sensors. We believe that the research communities need to make faster progress in the development of algorithms that can transfer capabilities learned in simulation to reality.

LIFELONG MACHINE LEARNING To enable long-term dynamic scene understanding for robots, we will need intelligent systems that allow these robots to accurately and robustly perceive and understand the world around them. As humans, we can perceive and understand complex visual scenes from an early age. This enables us to learn and reason about complex situations, ideally allowing us to build and update a *world model*, i.e. a spatial and temporal representation of the environment, from early on. Unfortunately, building and updating such a model is still an incredibly hard challenge for today's artificial intelligence systems. We believe that the concept of lifelong learning is key to true human intelligence, and we are excited to see what future research will bring us.

FINAL REMARKS Although our contributions in this dissertation are mainly from the field of computer vision, we believe that researchers from computer vision, graphics, machine learning, and robotics need to collaborate more closely to develop the next generation of artificial intelligence systems. Improvements in every field will help advance artificial intelligence and will enable us to get one step closer to building robots with human-like capabilities. Solving all these problems is an incredibly exciting challenge that will keep researchers busy for decades.

Part IV

APPENDIX

A ORIGINAL PUBLICATIONS

This chapter includes the original publications of the peer-reviewed research papers [2–5, 7] this cumulative dissertation is based on. The works in Chapters 4, 5, 6, 7, and 8 have revised layouts as well as minor content adaptations compared to the original publications included in this appendix. Additionally, a detailed disclaimer for each paper, indicating a copyright notice, the publication abstract, and the specific individual contributions of the author of this thesis is provided.

A.1 MODULAR VEHICLE CONTROL FOR TRANSFERRING SEMANTIC INFORMATION BETWEEN WEATHER CONDITIONS USING GANS

Copyright

Abstract

Even though end-to-end supervised learning has shown promising results for sensorimotor control of self-driving cars, its performance is greatly affected by the weather conditions under which it was trained, showing poor generalization to unseen conditions. In this paper, we show how knowledge can be transferred using semantic maps to new weather conditions without the need to obtain new ground truth data. To this end, we propose to divide the task of vehicle control into two independent modules: a control module which is only trained on one weather condition for which labeled steering data is available, and a perception module which is used as an interface between new weather conditions and the fixed control module. To generate the semantic data needed to train the perception module, we propose to use a generative adversarial network (GAN)-based model to retrieve the semantic information for the new conditions in an unsupervised manner. We introduce a master-servant architecture, where the master model (semantic labels available) trains the servant model (semantic labels not available). We show that our proposed method trained with ground truth data for a single weather condition is capable of achieving similar results on the task of steering angle prediction as an end-to-end model trained with ground truth data of 15 different weather conditions.

Individual Contributions

Problem definition	*significantly contributed*
Literature survey	*significantly contributed*
Implementation	*significantly contributed*
Experimental evaluation	*contributed*
Preparation of the manuscript	*significantly contributed*

Modular Vehicle Control for Transferring Semantic Information Between Weather Conditions Using GANs

Patrick Wenzel[1,2*], Qadeer Khan[1,2*], Daniel Cremers[1,2], and Laura Leal-Taixé[1]

[1]Technical University of Munich
[2]Artisense

Abstract: Even though end-to-end supervised learning has shown promising results for sensorimotor control of self-driving cars, its performance is greatly affected by the weather conditions under which it was trained, showing poor generalization to unseen conditions. In this paper, we show how knowledge can be transferred using semantic maps to new weather conditions without the need to obtain new ground truth data. To this end, we propose to divide the task of vehicle control into two independent modules: a control module which is only trained on one weather condition for which labeled steering data is available, and a perception module which is used as an interface between new weather conditions and the fixed control module. To generate the semantic data needed to train the perception module, we propose to use a generative adversarial network (GAN)-based model to retrieve the semantic information for the new conditions in an unsupervised manner. We introduce a master-servant architecture, where the master model (semantic labels available) trains the servant model (semantic labels not available). We show that our proposed method trained with ground truth data for a single weather condition is capable of achieving similar results on the task of steering angle prediction as an end-to-end model trained with ground truth data of 15 different weather conditions.

Keywords: Imitation learning, transfer learning, modular vehicle control

1 Introduction

One major goal of robotics and artificial intelligence research is to develop self-driving cars which can accurately perceive the environment and interact with the world. To develop an approach for addressing these problems, we have to deal with enormous challenges in perception, control, and localization. In general, the task of building an autonomous driving system can be divided into two parts: 1) path planning, and 2) vehicle control. Path planning provides a global solution for reaching a destination from a given starting position. It uses various information from different sensors such as GPS, IMU, and traffic conditions to infer the most optimized path. Meanwhile, vehicle control is meant to provide a local solution for predicting the immediate steering commands at the current instance in time. It utilizes information from sensors such as RGB cameras, lidar or radar. These sensors allow the self-driving car to sense and understand its current surroundings, such as the status of traffic lights or the presence of a pedestrian or another vehicle in front of the car.

In this paper, we focus our attention only on vehicle control to explain how transfer learning can be utilized to improve the robustness and stability of predicting steering commands even for unseen weather conditions for which no supervised data is available. For this, the task of vehicle control is segregated into perception and control. Figure 1 represents two modules, with each performing one of these tasks. The purpose of the perception module is to pre-process the raw input sensor data

[*]The majority of this work was done while at Audi Electronics Venture with equal contribution. Correspondence to: Patrick Wenzel <wenzel@cs.tum.edu>.

and extract useful features. In our approach, we use images captured by an RGB camera to extract semantic features of the scene. These extracted features are then fed to the control module which aims to produce the correct steering command for that particular sensor input.

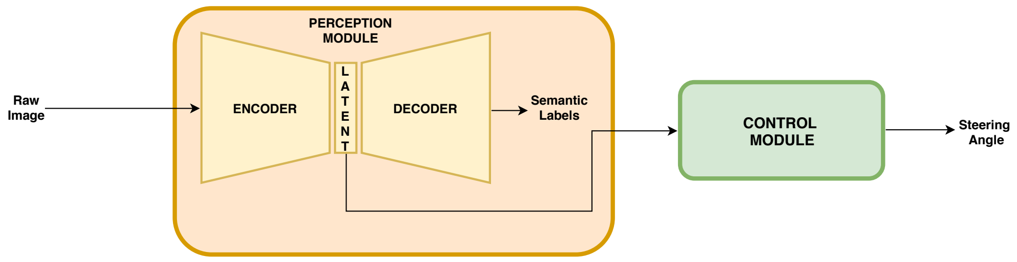


Figure 1: The perception module is trained as an encoder-decoder architecture, without any skip connections. The encoder sub-module first embeds the raw image into a lower dimensional latent vector. The decoder sub-module reconstructs the semantic scene from this latent vector. If the low dimensional latent vector contains all the necessary information to reconstruct the semantic scene to a reasonable degree of accuracy, then we directly feed it as an input to the control module instead of the semantic labels.

Modular pipeline vs end-to-end learning. In an end-to-end training approach, both the perception and the control module would be trained together [1]. We propose to split the task into separate perception and control, so that each module is trained and optimized independently without affecting each other. The main advantage of the separate modules is that without retraining the control module, we can simply replace the perception module to work on different environmental conditions, whereas in an end-to-end learning system, supervised labels for the new domain would first be needed to be collected and then the control module would also need to be retrained on this additional data.

Our main contributions are the following:

- Ability to control the vehicle in unseen weather conditions without having the need to collect additional data for the steering commands and without requiring to retrain the control module. This is done by simply replacing the perception module additionally trained on the semantics of the new condition.
- We show how knowledge can be transferred from a weather condition for which semantic labels are available to other weather conditions for which no labels exist in an unsupervised manner using GANs.

2 Related Work

Supervised learning for self-driving cars. The use of supervised learning methods to train driving policies for self-driving cars is a well-known and common approach. The first step towards using neural networks for the task of road following dates back to ALVINN [2]. This approach uses a very simple shallow network which maps images and a laser range finder as input and produces action predictions. Recently, NVIDIA [3] proposed to use deep convolutional neural networks trained end-to-end for a simple lane following task. This approach was successful in relatively simple real-world scenarios. One major drawback of end-to-end imitation learning is that it cannot generalize well across different domains for which no labeled training data is available. However, most end-to-end learning approaches [4, 5, 6] suffer from this problem.

Transfer learning. Generative adversarial networks provide a framework to tackle this generalization gap [7] by image generation techniques which can be used for domain adaptation. The authors of [8] proposed a network that can convert non-realistic virtual images into realistic ones with similar scene structure. Similarly, Hoffman et al. [9] proposed a novel discriminatively-trained adversarial model which can be used for domain adaptation in unseen environments. They show new state-of-the-art results across multiple adaptation tasks, including digit classification and semantic segmentation of road scenes.

Semantic segmentation. Visual understanding of complex environments is an enabling factor for self-driving cars. The authors of [10] provide a large-scale dataset with semantic abstractions of real-world urban scenes focusing on autonomous driving. By using semantic segmentation, it is possible to decompose the scene into pixel-wise labels we are particularly interested in. This especially helps self-driving cars to discover driveable areas of the scene. It is therefore possible to segment a scene into different classes (e. g. road and not road) [11].

Modular pipeline vs end-to-end learning. The authors of [12] trained both an end-to-end and a modular based model on one set of weather conditions and tested the model on a different set of weather conditions. Based on their experiments they concluded that the modular approach is more vulnerable to failures under complex weather conditions than the end-to-end approach.

Our method also uses a modular approach, but additionally introduces an image translation technique to overcome the generalization gap between the unseen weather conditions. This is done by only retraining the perception module without having to retrain the control module for each and every domain (i. e. weather condition). A useful consequence of this is that we do not have to recollect additional labeled data for the new conditions.

3 Imitation Learning on the Latent Semantic Vector

Perception module. In this work, we use images captured by an RGB camera placed at the front of the car as inputs to the perception module. The perception module processes these images and produces an output map containing the semantics of the scene, which in turn can be used as an input to the control module. The CARLA [12] simulator yields semantic labels for 13 classes. The advantage of using semantic labels instead of raw RGB data is described below:

- Figure 2 shows how two weather conditions have different RGB inputs but the same semantic pixel labels. Hence, the control module does not separately need to learn to predict the correct steering commands for each and every weather condition.
- The semantic labels can precisely localize the pixels of important road landmarks such as traffic lights and signs. The status/information contained on these can then be read off to take appropriate planning and control decisions.
- A high proportion of the pixels have the same label as its neighbors. This redundancy can be utilized to reduce the dimensionality of the semantic scene. Hence, the number of parameters required to train the control module can then also be reduced.

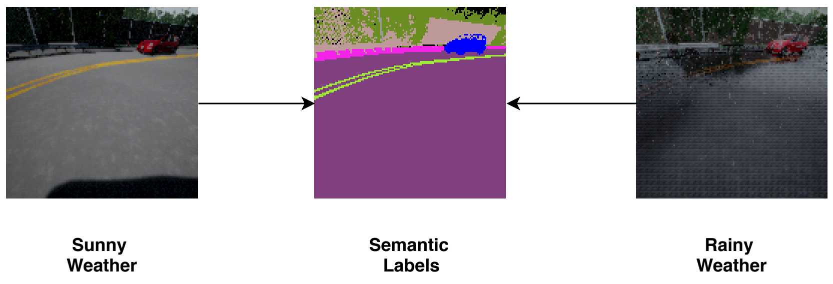


Figure 2: For the perception module we take in raw image data as obtained from the car's camera and output the semantic segmentation of the scene. Notice that irrespective of the weather condition the semantics of the scene remain the same. Since the perception module bears the burden of producing the correct semantic labels, the control module would be robust to changes in lighting, weather, and climate conditions.

The perception module, which is used to produce the semantic labels of a scene from the RGB camera is trained as an encoder-decoder architecture. The network architecture which is being used is a modified version of the one proposed by Larsen et al. [13]. The structure and the parameters of the model are shown in the supplementary material. The encoder first encodes the information contained in the input data to a lower dimensional latent vector. The decoder, then takes this latent vector and attempts to reconstruct the semantics of the scene. The output of the decoder is of the same size as the image but having 13 channels with each representing the probability of occurrence of one of the semantic labels. The model is trained by minimizing the weighted sum of the categorical cross-entropy of each pixel in the image. The categorical cross-entropy (negative log-likelihood) between predictions p and targets t is calculated as follows:

$$\mathcal{L}_i = -\sum_j t_{i,j} \log(p_{i,j}) w_j,$$

where i denotes the pixel and j denotes the class. The weight w_j of each semantic label is inversely proportional to its frequency of occurrence in the dataset.

Control module. Note that we do not use skip connections between the encoder and decoder of the perception module. Therefore, since the lower dimensional latent vector is capable of reconstructing the semantic labels of the scene, we can directly use this vector as input to the control module instead of the complete scene. Figure 1 depicts how the latent semantic embedding vector produced by the encoder of the perception module can be used as an input to the control module.

The control module aims to predict the correct steering angle, from the latent embedding fed as an input to the model. The data used for training the control module is collected in a supervised manner by recording images and their corresponding steering angles. The loss function attempts to minimize the mean squared error (MSE) between the actual steering angle and the one predicted by the model across all the samples. The architecture of the control model is depicted in the supplementary material.

4 Master-Servant Architecture for Transfer Learning

The control module does not perform well if tested in an environment which is completely different from the one on which the perception module was trained on. A naive and yet computational demanding solution could be to retrain the perception module under every other weather condition. However, this is not a viable solution for the following reasons:

- We would need semantic labels for every other weather condition. Obtaining semantic labels of a scene is a painstakingly slow process and prone to errors, since it requires human effort.
- Even if we have access to the semantic labels and retrain the perception module under the new environmental conditions, we would still have to also retrain the control module. This is due to the fact that the semantic latent vector produced by the new perception module might be different from the one produced by the old perception module, despite the same semantics of the scene. Figure 3 describes how for the same image, two independently trained segmentation models could yield different semantic vectors, despite being trained on the same data.

Proposed master-servant architecture. Suppose that the perception module P_0 and the control module C_0 are trained under a certain environmental condition. When tested on a very different weather condition P_0 may fail to produce the relevant semantic latent vector for the control module C_0 to take the correct steering decision. We would therefore like to replace P_0 with a different perception module P_1 such that it produces the correct latent vector to allow the same control module C_0 to execute the appropriate steering command even on this very different condition. For this, we propose a master-servant architecture model for training the perception module functioning on images from a domain for which no semantic labels are available. Figure 4 demonstrates the necessary steps of the master-servant architecture.

Suppose we have images (from domain X) and their corresponding semantic labels. With this, we can train a segmentation model using the encoder-decoder architecture described previously. We refer to the trained encoder of this model as the **master perception module** P_0. We would also like to obtain the correct semantic embedding of images (from domain Y) for new conditions for which no semantic labels are available. We refer to the perception module for which we would like to furnish the correct semantic embedding for images in domain Y as the **servant perception module** P_1. We use the master module, P_0, to train the servant module, P_1, in the steps described as follows:

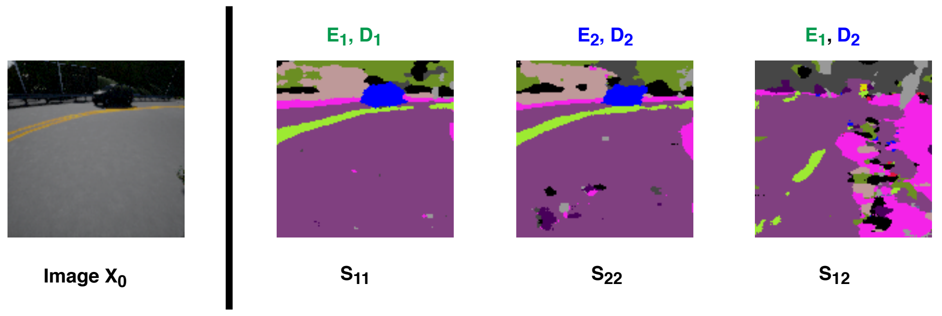


Figure 3: This figure shows the segmentation reconstructions S_{11} and S_{22} when image X_0 is passed through two segmentation models M_1 (with Encoder E_1, Decoder D_1) and M_2 (with Encoder E_2, Decoder D_2). Both models are trained independently on the same data. Note that the reconstructions reflect the true semantics of the scene reasonably well. S_{12} shows the reconstruction when the embedding from encoder E_1 is fed to through decoder of D_2. The ambiguity in S_{12} implies that for the same image the two models yield different semantic vectors.

1. An image X_0 is arbitrarily selected from domain X. X_0 is fed to through P_0 to obtain the semantic embedding of the scene denoted by z_0. Meanwhile, the generator G translates the image X_0 to generate an image Y_0 from domain Y, such that the semantics of the scene are preserved. If semantics are being preserved, then z_0 should be equal to z_1 (the semantic embedding obtained by feeding Y_0 through P_1).
2. Y_0 is fed through P_1 to get the predicted latent embedding z_1.
3. The mean squared error (MSE) between z_0 and z_1 is used as the loss function to update the weights of P_1 in order to minimize the difference between the two latent embeddings.

Some examples of the images produced by the generator G, segmentation reconstruction when z_0 (semantic embedding of the master) and z_1 (semantic embedding of the servant) is fed through the decoder of the master perception module P_0 are shown in the supplementary material.

Unsupervised transfer of semantics. We observe that with this master-servant architecture we are able to train the servant perception module for obtaining the correct semantic embeddings for images from domain Y for which semantic labels were never available. We can thus replace P_0 with P_1 which would also work on these unseen weather conditions without having to retrain the control module. Moreover, no additional human effort is required for the labeling of semantics.

The most critical component which made the functioning of this approach possible is the generator G, which is able to translate images between two different domains, while preserving the semantics. The generator G is pre-trained using the CycleGAN [14] approach. Unlike other image-to-image translation methods such as pix2pix [15], an important feature of CycleGANs is the fact that this approach does not require paired data between two domains. Therefore, the task of collecting (if even possible) images with a one-to-one correspondence between two domains can be eliminated. The procedure for training the generator G using the CycleGAN approach is shown in the supplementary. The architecture used was taken from [14]. The supplementary material shows some examples of paired and unpaired data from two different domains produced by the CARLA simulator.

5 Experimental Results

Experimental setup. For evaluating our method, we used the CARLA simulator. The CARLA simulator provides 15 different weather conditions (labeled from 0 to 14). We focus our attention on the car turning around corner scenarios since it is a more complicated maneuver to perform than lane following and it would thus give a better understanding of possible failure conditions. We train 5 different models to predict the steering angle whilst assuming that the car throttle is fixed. For a fair comparison, the approach is evaluated on multiple different turns and we do not consider the presence of pedestrians and cars in the ego vehicle's driving lane. The starting position of the agent is just before the curve and the duration of the turn is fixed to 120 frames since it covers the entire

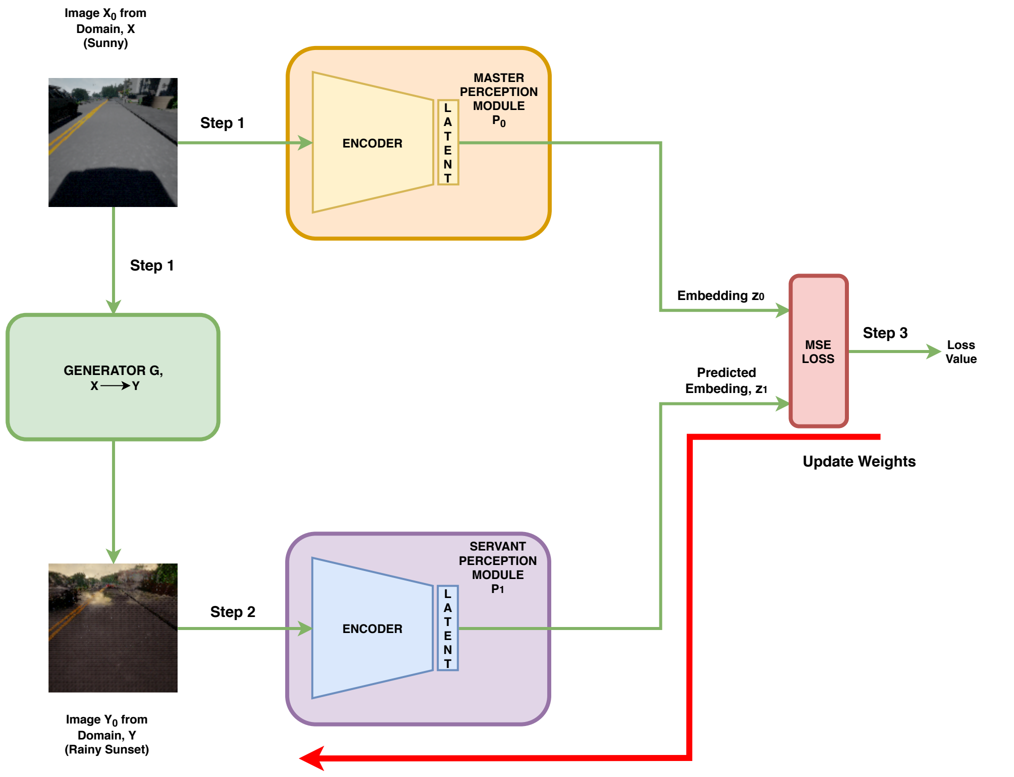


Image X_0 from Domain, X (Sunny)
Step 1
ENCODER
LATENT
MASTER PERCEPTION MODULE P_0
Step 1
GENERATOR G, X ⟶ Y
Embedding z_0
MSE LOSS
Step 3
Loss Value
Predicted Embeding, z_1
Update Weights
SERVANT PERCEPTION MODULE P_1
Step 2
ENCODER
LATENT
Image Y_0 from Domain, Y (Rainy Sunset)

Figure 4: We propose a master-servant architecture to train a servant perception module P_1 for images which do not have semantic labels in an unsupervised manner. Images in domain X have semantic labels and are used to train the perception module P_0, which we refer to as the master perception module. P_0 is pre-trained using the complete encoder-decoder architecture. Images in domain Y do not have semantic labels. The process works as follows. **Step 1:** The generator G is used to convert an image X_0 from domain X to an image Y_0 in domain Y such that the semantic information is preserved. Meanwhile X_0 is also fed to the master perception module P_0 to get the latent embedding z_0. **Step 2:** The image Y_0 is fed to the servant perception module P_1 to get the predicted latent embedding z_1. **Step 3:** Since the semantic labels of X_0 and Y_0 are the same, their latent embeddings should also be the same. We use the mean squared error (MSE) to minimize this difference, wherein the embedding z_0 is used as the true label for training P_1. **Update Weights:** We back-propagate the MSE loss to update the weights of only P_1 so that its embedding matches with that of P_0. The green arrows indicate forward propagation and the red arrow shows back-propagation.

turning maneuver. The turn is considered successful if the car did not crash whilst executing the turn. Furthermore, in order to make a quantitative evaluation of the performance of each of the 5 models, new test data containing the images and the corresponding true steering commands for each of the 15 weather conditions was collected. Figure 5 shows a plot of the mean squared error (MSE) between the actual and the predicted steering commands by the 5 different models across all the weather conditions on samples of the test data. Meanwhile, Table 1 enumerates the percentage of turns each of the 5 models are successfully able to execute across all the 15 weather conditions.

The supplementary material contains the description and some samples of the 15 weather conditions along with video samples demonstrating the performance of the models on certain weather conditions. The dataset can be downloaded at: `https://git.io/fApfH`. The details of the 5 models are given below:

End-to-end, all weathers. An end-to-end model is trained on all weather conditions. Here we have assumed that we have access to the steering commands across all the conditions. As can be seen from Figure 5, this model gives the lowest error particularly for weathers 1 to 14. Moreover, we

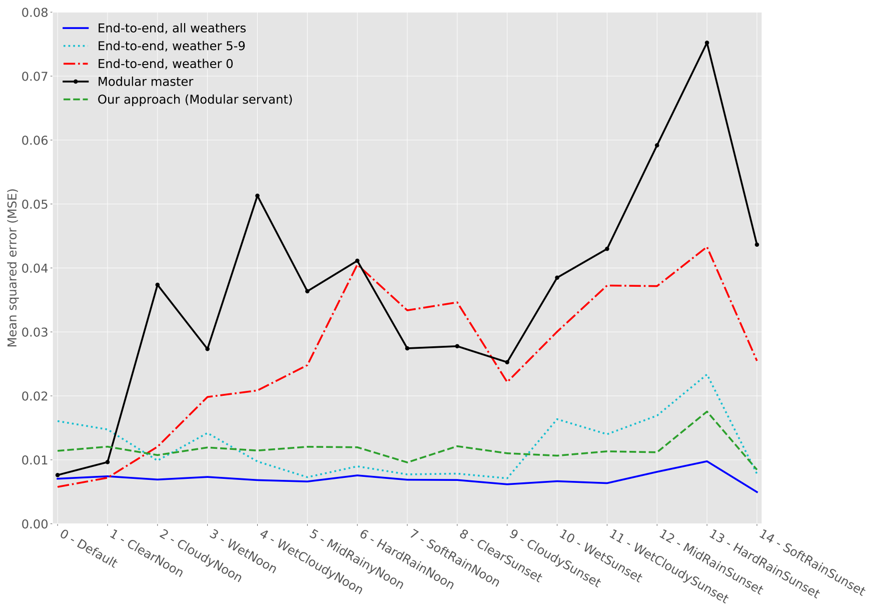


Figure 5: Plot of the mean squared error (MSE) between the actual and the predicted steering commands by 5 different models across the weather conditions 0 to 14. The **blue** line is the error plot for a model trained end-to-end, from images and corresponding steering commands for all the 15 weather conditions. The **cyan** error curve corresponds to the end-to-end model trained on images and steering commands for weathers 5-9. The **red** line is for the model trained end-to-end from images and corresponding steering commands for only the default weather condition 0. The **black** line represents the model referred to as the master whose perception and control modules are trained separately. The perception module is trained using the actual semantic labels available for the default weather condition, whereas the control model is trained from the actual steering commands of the same condition. The **green** curve is the model whose control model is the same as the one for the master, but the perception module is trained as a servant from the master perception module from images generated by the CycleGANs for weather conditions 2, 3, 4, 6, 8, 9, 10, 11, 12, and 13, in addition to the default condition 0.

observe in Table 1 that this model is able to successfully execute a high proportion of the turns across all the weather conditions, since it was trained on all of them. All subsequent models are trained with the steering commands available for a subset of the weather conditions and their performance is compared with this model.

End-to-end, weather 5-9. This model is trained end-to-end on weathers 5, 6, 7, 8, and 9 which were arbitrarily selected just to see how it would perform on unseen weather conditions. As shown in Figure 5 it has a relatively low error on these conditions and a higher error elsewhere. Furthermore, the plot shows that this end-to-end approach only seems to work well on the trained conditions for which we have labeled data. Moreover, as can be seen in Table 1, the model is capable of maneuvering well on the trained weather conditions and on those which are similar or have good visibility. However, on weather conditions 11-14 the model fails to execute the majority of the turns. This is mainly due to the fact that these weather conditions (11-14) are relatively disparate in terms of appearance and visibility as compared to the trained ones (5-9).

In practice, we do not have the steering commands available for all the possible or even a diverse subset of the weather conditions. Rather, the labeled data would correspond to only the condition of the day/period on which it was collected. Therefore, the 3 successive models that we now consider assume that the steering commands and the corresponding images/semantics are only available for the default weather condition (labeled as 0). From this, we evaluate how end-to-end training would compare to the proposed modular approach across all the remaining weather conditions for which no labeled data is available.

End-to-end, weather 0. This model is trained end-to-end from images and steering commands for the default weather condition. Figure 5, shows that this model outperforms all the other models only on weather condition 0 on which it was trained. For all other conditions, it gives high errors.

Modular master. This model is trained on the default weather condition (0) but the task is divided into 2 separate perception and control modules. The perception module P_0 is trained on the semantic labels. We refer to this perception module as the master, since it will later be used to train the servant module for retrieving the semantic information of the unseen weather conditions. The control module is in turn trained with imitation learning to predict the steering angle of the car from the latent embedding generated by the encoder of P_0. The forth row of Table 1 depicts the percentage of turns the model was successfully able to maneuver for each of the 15 conditions. As observed in the table, the model is successful only on the default weather conditions (on which it was trained) and the sunny weather condition (which closely resembles the default condition). Similar to the previous model (trained end-to-end on the default condition), this model also fails on a large proportion when tested on weather conditions that are far off from the default condition in terms of visual appearance. From this, there seems to be no apparent advantage of using a modular approach over the end-to-end training when we have access to the labels for only one weather condition. Nevertheless, the master perception module P_0 obtained through this method will serve as a baseline for training a servant perception module that additionally works for unseen weather conditions. This approach is described in the following.

Our approach (Modular servant). We train one servant perception module to cater for weather conditions on which P_0 failed to perform. We selected a subset of weather conditions (i. e. 2, 3, 4, 6, 8, 9, 10, 11, 12, and 13) to train the servant module. Using CycleGANs, separate generators were trained between each of these conditions and the default weather condition. The images produced by the CycleGAN generators for each of these conditions were fed as an input in equal proportion along with the default images to train only a single servant perception module P_1. Despite having no access to the steering commands and the semantic labels for weather conditions 1 to 14, Figure 5 shows that the error for this model across these 14 weather conditions is significantly lower than the previous 2 models which were also trained only from labels of weather condition 0. Moreover, we see from the last row of Table 1, that this model is successfully able to execute a good proportion of the turns for most of the weather conditions. Only on condition 13 (HardRainSunset), the model fails to perform well. The visibility under this condition is low and the images generated by the CycleGAN do not seem to preserve the semantics, hence resulting in the model to perform relatively poorly. Nevertheless, on all the other remaining weather conditions its performance is comparable to the first end-to-end model trained on steering labels for all the weather conditions.

Table 1: The table reports the percentage of successfully completed turns by the 5 models for each weather condition. Higher is better.

	Weather condition															
Model	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	average
End-to-end, all weathers	100	100	88	88	100	100	100	100	100	100	100	88	88	88	100	96
End-to-end, weather 5-9	88	88	100	88	100	100	100	100	100	100	100	50	50	25	88	85
End-to-end, weather 0	100	63	38	38	13	13	0	0	0	50	13	0	0	0	0	22
Modular master	100	100	88	50	50	63	50	50	50	63	75	50	0	0	50	56
Our approach (Modular servant)	100	100	100	100	88	100	100	100	100	100	100	88	100	63	100	96

6 Conclusion

In this paper, we have shown that in order to generalize vehicle control across unseen weather conditions it is worthwhile to divide the task into separate perception and control modules. This separation eliminates the tedious task of recollecting labeled steering command data for each and every new environment the vehicle might come across. Moreover, retraining of the control module for new environments can be avoided by a simple replacement of the perception module. The initial perception module was trained from the semantic labels available only for one of the weather conditions. For environments for which semantic labels are missing, the proposed master-servant architecture can be deployed for transferring semantic knowledge from one domain to another (i. e. between different weather conditions) in an unsupervised manner using CycleGANs which do not require paired data. We believe that the presented approach to making driving policies more robust by training under different weather conditions will prove useful in future research.

Acknowledgments

This research was partially funded by the Humboldt Foundation through the Sofja Kovalevskaja Award.

References

[1] C. Chen, A. Seff, A. Kornhauser, and J. Xiao. DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2722–2730, 2015.

[2] D. A. Pomerleau. ALVINN: An Autonomous Land Vehicle in a Neural Network. In *Neural Information Processing Systems (NIPS)*, pages 305–313, 1988.

[3] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba. End to End Learning for Self-Driving Cars. *arXiv preprint arXiv:1604.07316*, 2016.

[4] J. Zhang and K. Cho. Query-Efficient Imitation Learning for End-to-End Autonomous Driving. *arXiv preprint arXiv:1605.06450*, 2016.

[5] D. Silver, J. A. Bagnell, and A. Stentz. Learning from Demonstration for Autonomous Navigation in Complex Unstructured Terrain. *The International Journal of Robotics Research*, 29 (12):1565–1592, 2010.

[6] Y. LeCun, U. Muller, J. Ben, E. Cosatto, and B. Flepp. Off-Road Obstacle Avoidance through End-to-End Learning. In *Neural Information Processing Systems (NIPS)*, pages 739–746, 2006.

[7] L. Karacan, Z. Akata, A. Erdem, and E. Erdem. Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts. *arXiv preprint arXiv:1612.00215*, 2016.

[8] Y. You, X. Pan, Z. Wang, and C. Lu. Virtual to Real Reinforcement Learning for Autonomous Driving. *arXiv preprint arXiv:1704.03952*, 2017.

[9] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. A. Efros, and T. Darrell. CyCADA: Cycle-Consistent Adversarial Domain Adaptation. *arXiv preprint arXiv:1711.03213*, 2017.

[10] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3213–3223, 2016.

[11] B.-K. Chen, C. Gong, and J. Yang. Importance-Aware Semantic Segmentation for Autonomous Driving System. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1504–1510, 2017.

[12] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An Open Urban Driving Simulator. In *Conference on Robot Learning (CoRL)*, pages 1–16, 2017.

[13] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*, 2015.

[14] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2242–2251, 2017.

[15] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-Image Translation with Conditional Adversarial Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976, 2017.

Supplementary Material

Table S.1: Encoder-decoder architecture used to train the segmentation perception module for the master and all servant models. The convolution layers numbered 15 and 16 have a kernel size of 4, stride of 1 and no padding. All other convolution layers have kernel size 4, stride of 2 and padding of 1. All the Leaky ReLU activation functions have a negative slope of 0.2. The output of the model has 13 channels with each corresponding to one of the semantic labels. The output of the last layer of the encoder (Layer 15) is fed to the control module to predict the correct steering direction. The same layer is also used to train the encoders of all servant modules. The code and model of the architecture is a modified version of `https://github.com/seangal/dcgan_vae_pytorch`.

Encoder				Decoder			
Layer Number	Layer Type	Layer Input	Layer Output	Layer Number	Layer Type	Layer Input	Layer Output
1	Convolution	$3 \times 128 \times 128$	$32 \times 64 \times 64$	16	Convolution (Transpose)	$64 \times 1 \times 1$	$512 \times 4 \times 4$
2	Leaky ReLU activation	$32 \times 64 \times 64$	$32 \times 64 \times 64$	17	Batch normalization	$512 \times 4 \times 4$	$512 \times 4 \times 4$
3	Convolution	$32 \times 64 \times 64$	$64 \times 32 \times 32$	18	Leaky ReLU activation	$512 \times 4 \times 4$	$512 \times 4 \times 4$
4	Batch normalization	$64 \times 32 \times 32$	$64 \times 32 \times 32$	19	Convolution (Transpose)	$512 \times 4 \times 4$	$256 \times 8 \times 8$
5	Leaky ReLU activation	$64 \times 32 \times 32$	$64 \times 32 \times 32$	20	Batch normalization	$256 \times 8 \times 8$	$256 \times 8 \times 8$
6	Convolution	$64 \times 32 \times 32$	$128 \times 16 \times 16$	21	Leaky ReLU activation	$256 \times 8 \times 8$	$256 \times 8 \times 8$
7	Batch normalization	$128 \times 16 \times 16$	$128 \times 16 \times 16$	22	Convolution (Transpose)	$256 \times 8 \times 8$	$128 \times 16 \times 16$
8	Leaky ReLU activation	$128 \times 16 \times 16$	$128 \times 16 \times 16$	23	Batch normalization	$128 \times 16 \times 16$	$128 \times 16 \times 16$
9	Convolution	$128 \times 16 \times 16$	$256 \times 8 \times 8$	24	Leaky ReLU activation	$128 \times 16 \times 16$	$128 \times 16 \times 16$
10	Batch normalization	$256 \times 8 \times 8$	$256 \times 8 \times 8$	25	Convolution (Transpose)	$128 \times 16 \times 16$	$64 \times 32 \times 32$
11	Leaky ReLU activation	$256 \times 8 \times 8$	$256 \times 8 \times 8$	26	Batch normalization	$64 \times 32 \times 32$	$64 \times 32 \times 32$
12	Convolution	$256 \times 8 \times 8$	$512 \times 4 \times 4$	27	Leaky ReLU activation	$64 \times 32 \times 32$	$64 \times 32 \times 32$
13	Batch normalization	$512 \times 4 \times 4$	$512 \times 4 \times 4$	28	Convolution (Transpose)	$64 \times 32 \times 32$	$32 \times 64 \times 64$
14	Leaky ReLU activation	$512 \times 4 \times 4$	$512 \times 4 \times 4$	29	Batch normalization	$32 \times 64 \times 64$	$32 \times 64 \times 64$
15	Convolution	$512 \times 4 \times 4$	$64 \times 1 \times 1$	30	Leaky ReLU activation	$32 \times 64 \times 64$	$32 \times 64 \times 64$
				31	Convolution (Transpose)	$32 \times 64 \times 64$	$13 \times 128 \times 128$
				32	Sigmoid activation	$13 \times 128 \times 128$	$13 \times 128 \times 128$

Table S.2: Architecture of the control model. Note that the input to the control module is a vector of size 64, corresponding to the size of the latent embedding produced by the encoder of the perception module.

Layer Number	Layer Type	Layer Input	Layer Output
1	Fully connected	64	100
2	ReLU activation	100	100
3	Fully connected	100	50
4	ReLU activation	50	50
5	Fully connected	50	25
6	ReLU activation	25	25
7	Fully connected	25	15
8	ReLU activation	15	15
9	Fully connected	15	8
10	ReLU activation	8	8
11	Fully connected	8	1

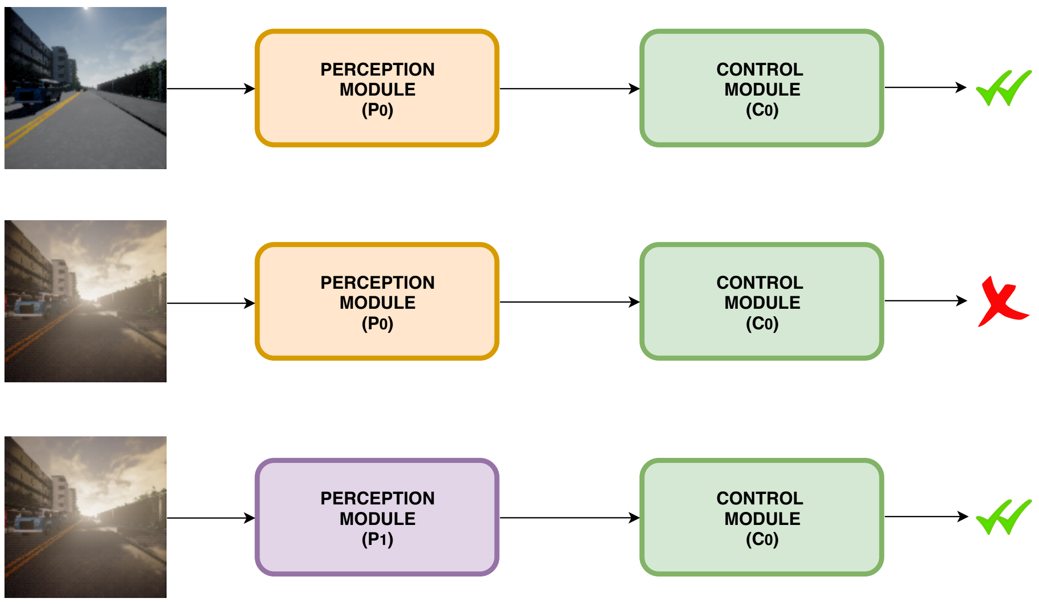


Figure S.1: To obtain a good segmentation based perception module, semantic labels for a diverse range of environmental conditions are required. This may not always be the case since semantic labeling is a tedious and error-prone process. Hence, we may have only access to a limited subset of the labeled data. **Top:** This figure shows a perception module P_0 trained only on sunny weather conditions. Hence, when a similar data is fed to P_0 at test time the control model C_0 performs as per expectation. **Center:** This figure demonstrates that if data from a different weather condition is fed to P_0, the control module C_0 may not necessarily perform as desired. **Bottom:** This figure shows that we would like to replace P_0 with P_1, such that P_1 is capable of handling this unseen environment in a manner to retain the same semantic embedding. Hence, we can use the same control module C_0 with P_1.

Table S.3: Table describing the contents and performance of the videos. Note that for the modular approach the control module was only trained on weather 0. Video pairs (5/6, 7/8, and 9/10) have the same starting position to compare between the modular master and our proposed modular servant approach. The videos are available at `https://www.youtube.com/playlist?list=PLbT2smuiIncsR_s9YA6KFpsa8gMwus5u7`.

Video Id	**Model**	**Tested on Weather**	**Comments**
video1	End-to-end, weather 0	0	successfully turning
video2	Modular master	0	successfully turning
video3	End-to-end, weather 5-9	8	successfully turning
video4	End-to-end, weather 5-9	3	crashing, model only trained on weather 5-9
video5	Modular master	2	crashing, segmentation module only trained on weather 0
video6	Our approach (Modular servant)	2	successfully turning, segmentation module trained with master-servant architecture
video7	Modular master	4	crashing, segmentation module only trained on weather 0
video8	Our approach (Modular servant)	4	successfully turning, segmentation module trained with master-servant architecture
video9	Modular master	11	crashing, segmentation module only trained on weather 0
video10	Our approach (Modular servant)	11	successfully turning, segmentation module trained with master-servant architecture
video11	End-to-end, all weathers	13	successfully turning
video12	End-to-end, weather 0	13	crashing, model only trained on weather 0
video13	Modular master	13	crashing, segmentation module only trained on weather 0
video14	Our approach (Modular servant)	13	turning, unstable segmentation
video15	Our approach (Modular servant)	13	crashing, unstable segmentation

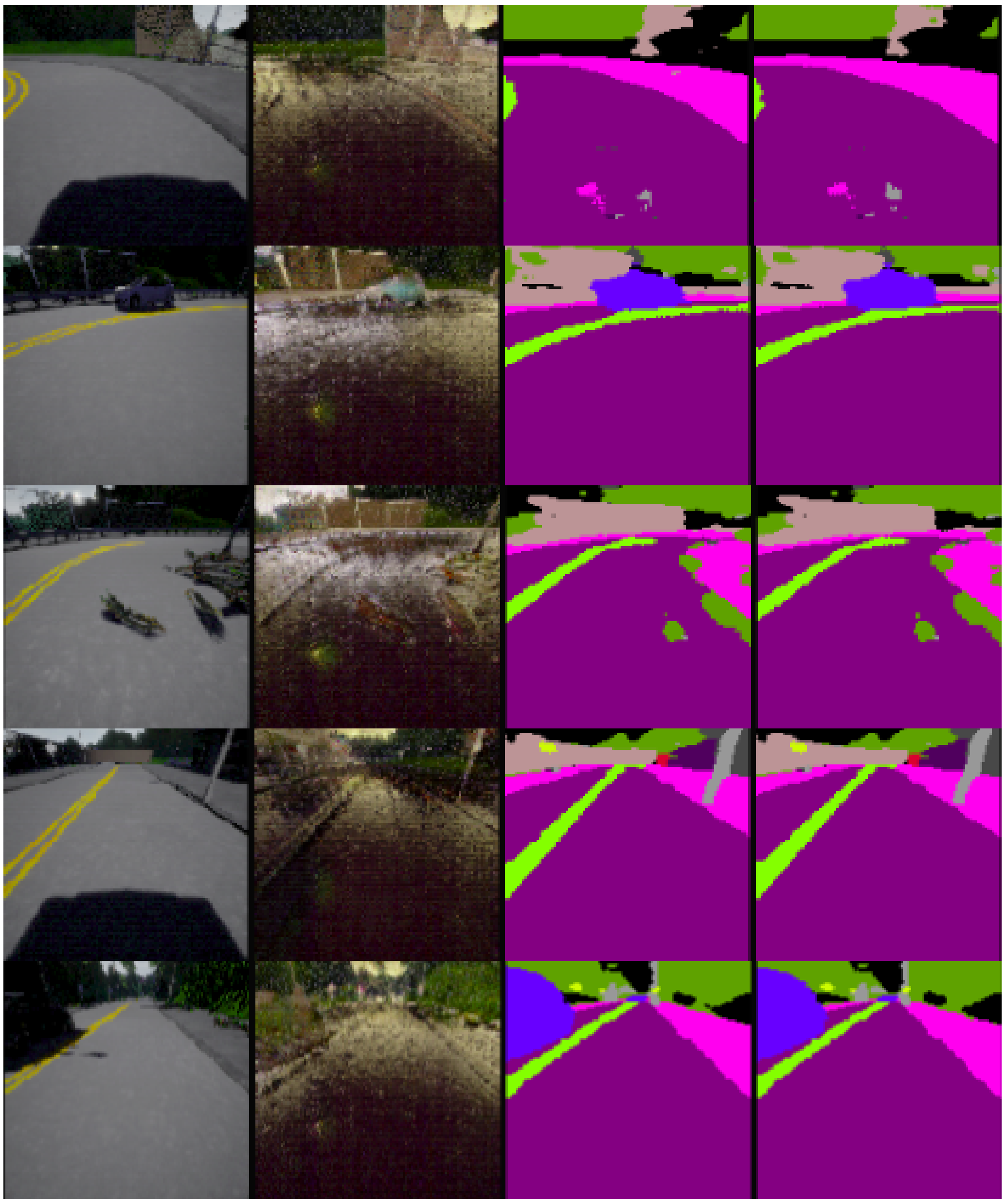


Figure S.2: This figure is with reference to the master-servant architecture. **1st column:** The first column shows five sample images from domain X. **2nd column:** The second column shows corresponding images from domain Y, produced by the generator G, maintaining the semantics of the scene. **3rd column:** The segmentation reconstruction produced by feeding z_1 through the the master decoder. z_1 in turn is generated by feeding the images in the 2nd column through the servant perception module P_1. **4th column:** Segmentation reconstruction produced by feeding z_0 through the master decoder. z_0 is generated by feeding the images in the 1st column through the master perception module P_0. Note that the semantic reconstructions in the 3rd column and 4th column are almost indistinguishable.

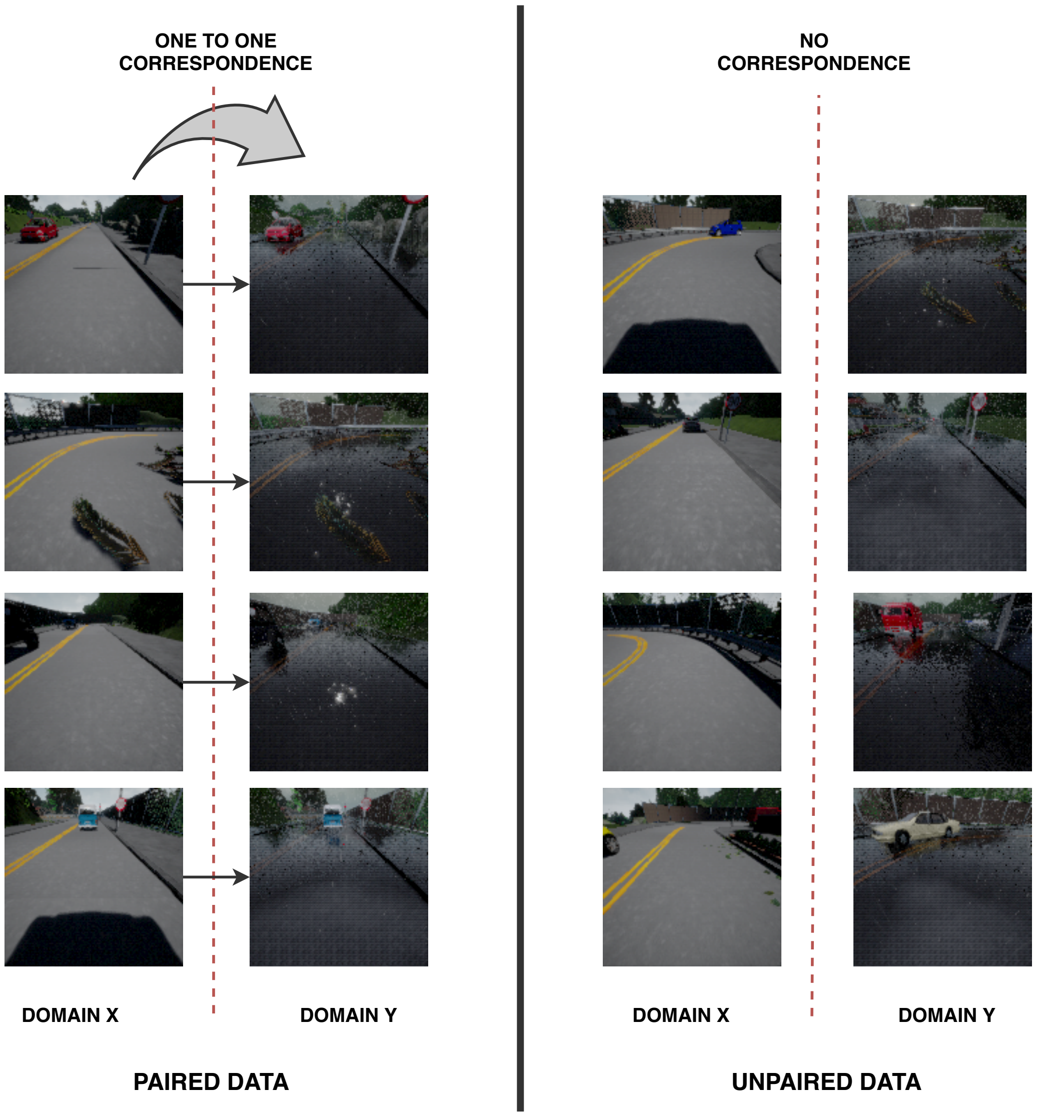


Figure S.3: Examples of paired and unpaired dataset. Note that it is practically not possible to obtain an exact one-to-one correspondence between two differing road conditions. Hence, we use CycleGANs for image-to-image translation between unpaired images. The domains correspond to weather conditions of a sunny day and a rainy afternoon, respectively.

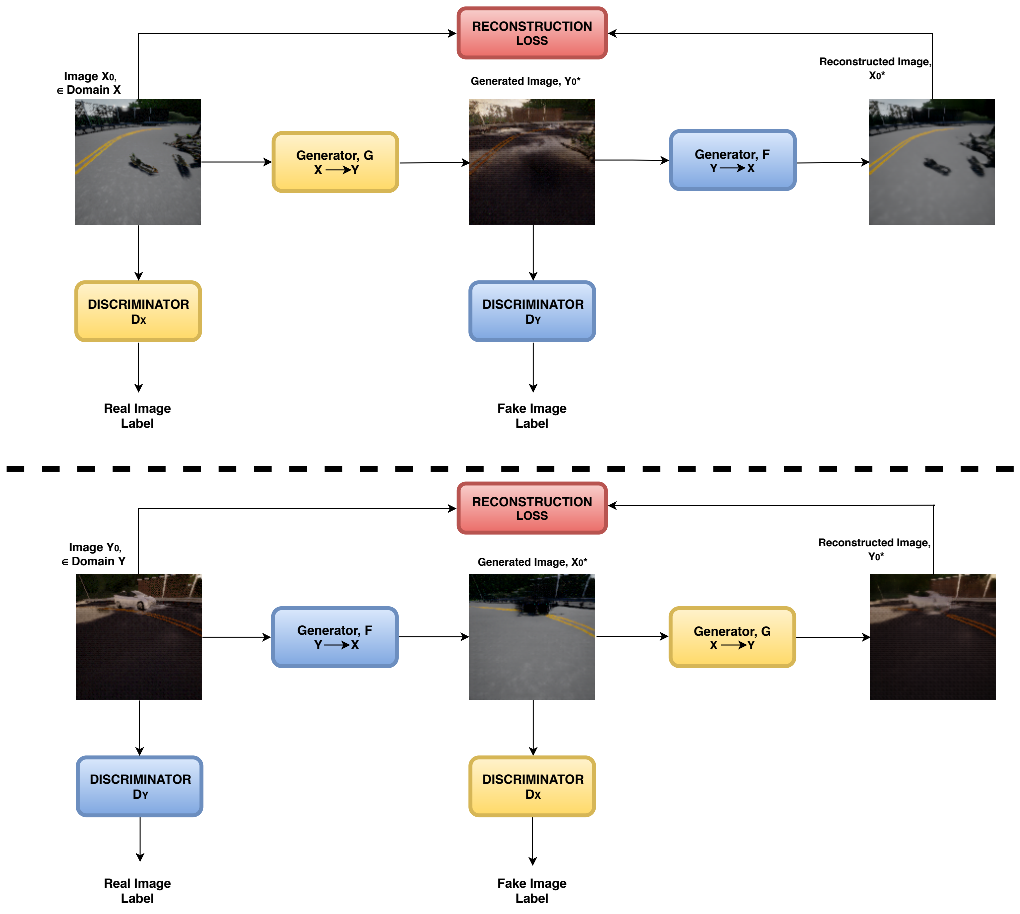


Figure S.4: The critical component in the master-servant architecture in achieving unsupervised training of the servant perception module is the generator G, which transformed images from domain X to domain Y, while maintaining the semantics of the scene. The generator G is trained using CycleGANs. Unpaired images from domain X and Y produced by CARLA are used for training of the model. The top figure shows an arbitrary image X_0 from domain X and is passed through the generator G, which generates an image Y_0^*. The generated image Y_0^* is then fed to another generator F, which generates an image X_0^*. The network is optimized by minimizing the L_1 loss between the real image X_0 and the generated image X_0^*. To make the images appear realistic, each domain has its own discriminator network i. e. D_x and D_y. The bottom figure is analogous to the top one except that here, we fed a realistic image from domain Y and try to minimize the L_1 loss between Y_0 and Y_0^*.

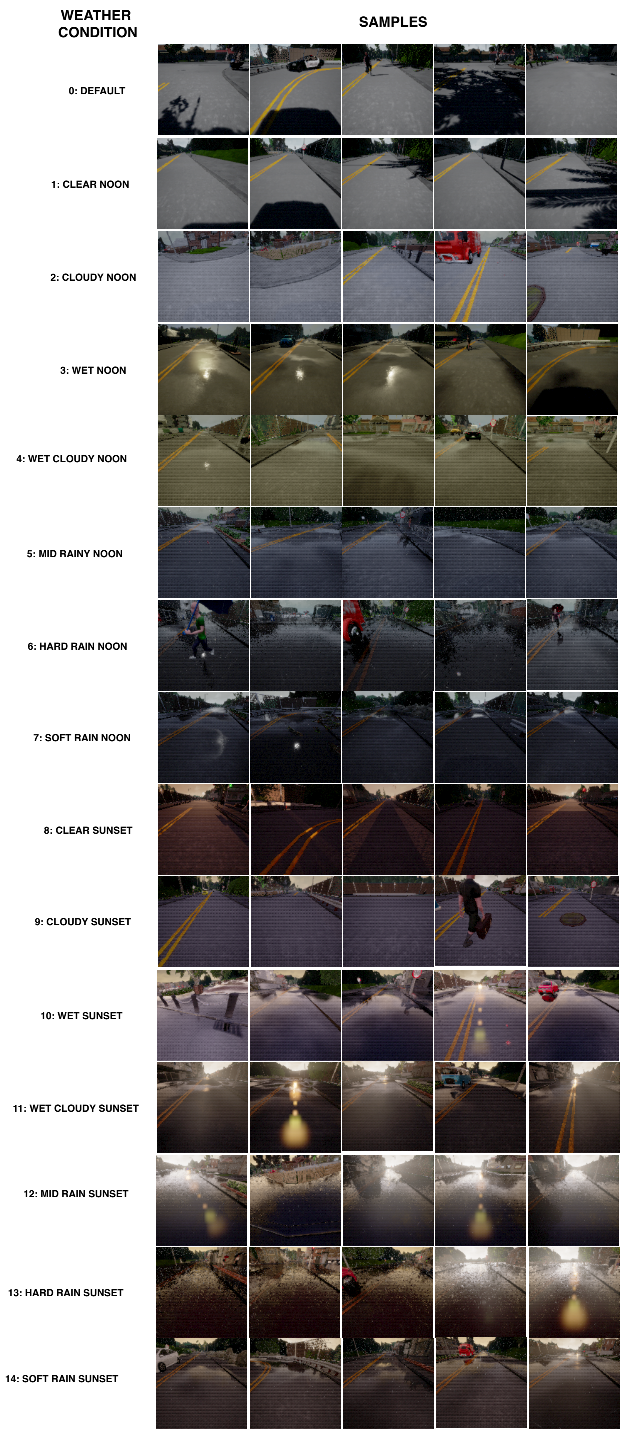


Figure S.5: Some sample images of the 15 different weather conditions along with their description generated by the CARLA simulator. Note that some of the weather conditions are very similar and therefore, a perception module trained for one of the conditions may also work for a similar condition also.

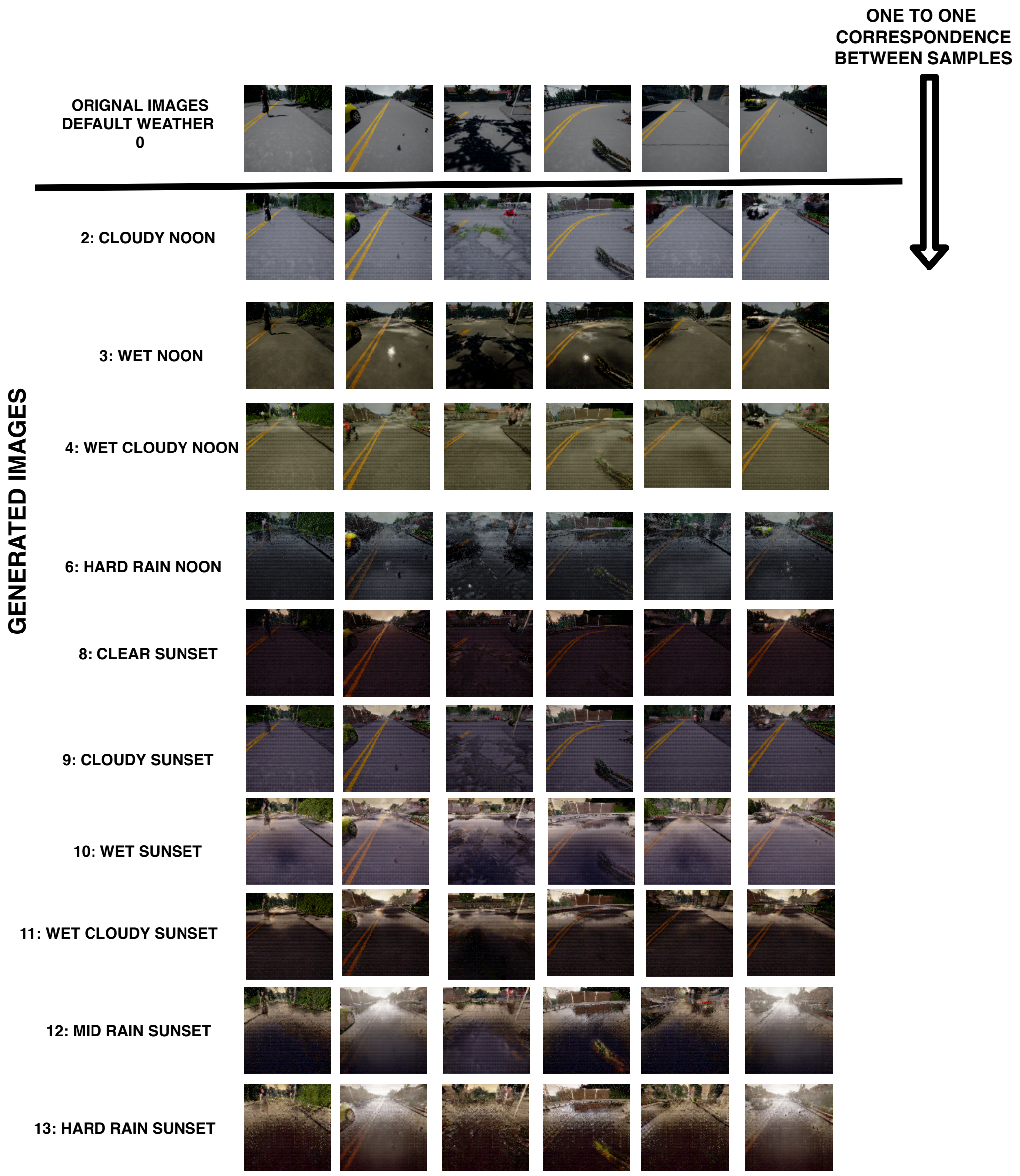


Figure S.6: The figure shows 6 sample images generated from the original default condition for weather conditions 2, 3, 4, 6, 8, 9, 10, 11, 12, and 13 using the CycleGAN approach. The CyleGAN was trained with 3500 images from the default and each of the other weather conditions. Most of the generated images resemble the actual to a reasonable degree. For weather conditions with low visibility, i. e. 12 and 13 some of the generated images (for e. g. sample 3, 5, and 6) give a poor reconstruction.

A.2 TOWARDS GENERALIZING SENSORIMOTOR CONTROL ACROSS WEATHER CONDITIONS

Copyright

Qadeer Khan, Patrick Wenzel, Daniel Cremers, and Laura Leal-Taixé

Towards Generalizing Sensorimotor Control Across Weather Conditions

IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019

DOI: 10.1109/IROS40897.2019.8968451

Abstract

The ability of deep learning models to generalize well across different scenarios depends primarily on the quality and quantity of annotated data. Labeling large amounts of data for all possible scenarios that a model may encounter would not be feasible; if even possible. We propose a framework to deal with limited labeled training data and demonstrate it on the application of vision-based vehicle control. We show how limited steering angle data available for only one condition can be transferred to multiple different weather scenarios. This is done by leveraging unlabeled images in a teacher-student learning paradigm complemented with an image-to-image translation network. The translation network transfers the images to a new domain, whereas the teacher provides soft supervised targets to train the student on this domain. Furthermore, we demonstrate how utilization of auxiliary networks can reduce the size of a model at inference time, without affecting the accuracy. The experiments show that our approach generalizes well across multiple different weather conditions using only ground truth labels from one domain.

Individual Contributions

Problem definition	*significantly contributed*
Literature survey	*significantly contributed*
Implementation	*contributed*
Experimental evaluation	*significantly contributed*
Preparation of the manuscript	*significantly contributed*

Notice

In accordance with the *IEEE Thesis / Dissertation Reuse Permissions*, we include the accepted version of the original publication [2] in the following.

Towards Generalizing Sensorimotor Control Across Weather Conditions

Qadeer Khan[1,2*] Patrick Wenzel[1,2*] Daniel Cremers[1,2] Laura Leal-Taixé[1]

***Abstract*— The ability of deep learning models to generalize well across different scenarios depends primarily on the quality and quantity of annotated data. Labeling large amounts of data for all possible scenarios that a model may encounter would not be feasible; if even possible. We propose a framework to deal with limited labeled training data and demonstrate it on the application of vision-based vehicle control. We show how limited steering angle data available for only one condition can be transferred to multiple different weather scenarios. This is done by leveraging unlabeled images in a teacher-student learning paradigm complemented with an image-to-image translation network. The translation network transfers the images to a new domain, whereas the teacher provides soft supervised targets to train the student on this domain. Furthermore, we demonstrate how utilization of auxiliary networks can reduce the size of a model at inference time, without affecting the accuracy. The experiments show that our approach generalizes well across multiple different weather conditions using only ground truth labels from one domain.**

I. Introduction

The ubiquity of a tremendous amount of processing power in contemporary computing units has proliferated the usage of deep learning-based approaches in control applications. In particular, supervised deep learning methods have made great strides in sensorimotor control, whether it be for autonomous driving [1], robot perception [2], or manipulation tasks [3], [4], [5]. However, the performance of such models is heavily dependent on the availability of ground truth labels. To have the best generalization capability, one should annotate data for all possible scenarios. Nonetheless, obtaining labels of high quality is a tedious, time consuming, and error-prone process.

We propose to instead utilize the information available for one domain and transfer it to a different one without human supervision as shown in Figure 1. This is particularly helpful for many robotic applications wherein a robotic system trained in one environment should generalize across different environments without human intervention. For example in simultaneous localization and mapping (SLAM), it is very important that the algorithm is robust to different lighting conditions [6]. In the context of autonomous driving, transferring knowledge from simulation to the real world or between different weather conditions is of high relevance. Recently, [7], [8], [9] have attempted to tackle these problems by dividing the task of vehicle control into different modules, where each module specialized in extracting features from a particular domain. In these works, semantic labels are used as an intermediate representation for transferring knowledge between different domains. However, obtaining these semantic labels requires human effort which is time-consuming, expensive, and error-prone [9]. In this work, we instead propose to use a teacher-student learning-based approach to generalize sensorimotor control across weather conditions without the need for extra annotations, *e.g.*, semantic segmentation labels.

To this end, we make the following contributions:

- We demonstrate how knowledge of ground truth data for steering angles can be transferred from one weather scenario to multiple different weather conditions. This is achieved without the additional requirement of having semantic labels. We make use of an image-to-image translation network to transfer the images between different domains while preserving information necessary for taking a driving decision.
- We show how the proposed method can also utilize images without ground truth steering commands to train the models using a teacher-student framework. The teacher provides relevant supervised information regarding the unlabeled images to train the features of the

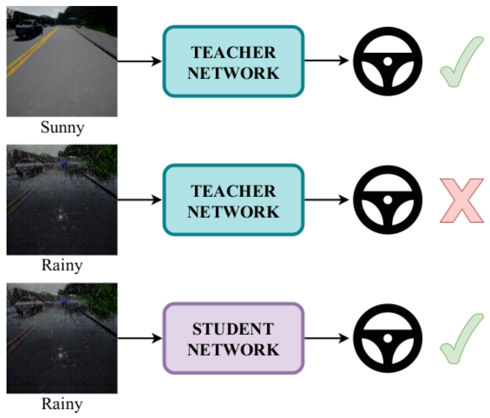


Fig. 1: Teacher-student training for generalizing sensorimotor control across weather conditions. **Top:** The teacher network, trained on ground truth data collected on sunny weather is capable of predicting the correct steering angle when tested on this condition. **Middle:** However, the teacher fails to predict the correct steering when tested on an input image from a different domain (rainy weather). **Bottom:** With our proposed framework, the student network trained with supervised information from the teacher network is capable of predicting the correct steering for the rainy weather. This is done without any additional ground truth labels or semantic information.

*These authors contributed equally.
[1]Technical University of Munich
[2]Artisense

student. Hence, we can eliminate the need for an expert driver for data collection across diverse conditions.

- If the sample data with ground truth labels is limited, then the teacher and student models may tend to overfit. To overcome this, we propose using weighted auxiliary networks connected to the intermediate layers of these models. During inference, the model size can be reduced by eliminating auxiliary layers with low weights without reducing accuracy.

In the following sections, we first review related work. We then present the details of our method, followed by an analysis of our model's performance. Finally, we discuss various parts of our model.

II. Related Work

Vision-based autonomous driving approaches have been studied extensively in an academic and industrial setting [10]. A plenty of real world [11], [12], [13] as well as synthetic [14], [15], [16], [17] datasets for autonomous driving research have become available. In recent years, neural network approaches have significantly advanced the state-of-the-art in computer vision tasks. Especially, end-to-end learning for sensorimotor control has recently gained a lot of interest in the vision and robotics community. In this context, different approaches to autonomous driving are studied: modular pipelines [18], imitation learning [19], conditional imitation learning [20], and direct perception [21].

Embodied agent evaluation. Most available datasets [11], [12] cannot be used for evaluating online driving performance due to their static nature. The evaluation of driving models on realistic data is challenging and often not feasible. Therefore, a lot of interest has emerged in building photo-realistic simulators [22], [23], [24] to analyze those models. However, despite having access to simulation engines, there is currently no universally accepted benchmark to evaluate vision-based control agents. Therefore, our experimental setup is a step towards a field where it is still not quite established how to evaluate and measure the performance of the models [25], [26].

Unpaired image-to-image translation networks. Unsupervised image-to-image translation techniques are rapidly making progress in generating high-fidelity images across various domains [27], [28], [29], [30]. Our framework is agnostic to any particular method. Hence, continual improvements in these networks can be easily integrated into our framework by replacing a previous network.

Transfer learning via semantic modularity. Several works used semantic labels of the scene as an intermediate representation for transferring knowledge between domains. In the context of autonomous driving, the authors of [7] proposed to map the driving policy utilizing semantic segmentation to a local trajectory plan to be able to transfer between simulation and real-world data. Furthermore, for making a reinforcement model trained in a virtual environment workable in the real world, the authors of [8] utilize the intermediate semantic representation as well to translate virtual to real images. However, there is still little work on generalizing driving models across weathers. The work by [9] showed how to transfer knowledge between different weather conditions using a semantic map of the scene. In contrast, in this paper, we demonstrate the possibility of transferring the knowledge between weathers even without semantic labels.

Knowledge distillation. Originally, knowledge distillation [31] was used for network compression (student network is smaller than the teacher while maintaining the accuracy). However, the authors of [32] focus on extracting knowledge from a trained (teacher) network and guide another (student) network in an individual training process. Furthermore, [33] used a slightly modified version of knowledge distillation for the task of pedestrian detection. In this work, we use a teacher-student architecture, but rather to leverage unlabeled data for sensorimotor control.

III. Sensorimotor Control Across Weathers

In this section, we introduce a computational framework for transferring knowledge of ground truth labels from one weather condition to multiple different scenarios without any semantic labels and additional human labeling effort. Figure 2 gives a high-level overview of the framework.

A. Teacher End-to-End Training

In this step, the teacher model is trained end-to-end in a supervised manner to predict the steering command of the vehicle from the raw RGB images generated by the camera placed at the front of the ego-vehicle. The training data is collected by an expert driver only once for that particular weather scenario. We refer to the images recorded under the weather condition under which this data was collected as belonging to domain D_0. Note that the teacher model is itself divided into a Feature Extraction Module (FEM), F_0 and a control module, C_0. The raw image (belonging to D_0) is passed through F_0 to retrieve a lower-dimensional feature representation. This feature representation is in turn fed to the C_0 which predicts the steering angle. A depiction of the model is shown in Figure 3. The FEM, F_0 is a sequential combination of 4 units where each unit comprises a convolutional, pooling, and activation layer. The output of unit 4 is flattened to a size of 800, which is in turn fed as an input to the module, C_0. The control module, C_0 is a series of fully connected layers and outputs the steering command.

Auxiliary network. It might be the case that the amount of images with labels is limited or the model is too large for the task at hand. Hence, the model may tend to overfit. Therefore, during training, to mitigate the effect of overfitting, F_0 additionally uses auxiliary networks connected to its intermediate layers [34]. Each of the auxiliary networks has a control module, C_0 with shared weights. The projection layers, P_1, P_2 and P_3 project the feature maps of the intermediate layers to the dimension of C_0 *i.e.* 800. The overall output of the teacher model is the weighted sum of the outputs of the auxiliary networks. The loss is also described by a weighted combination of the individual losses of the 4 auxiliary networks. The loss for each of the control modules

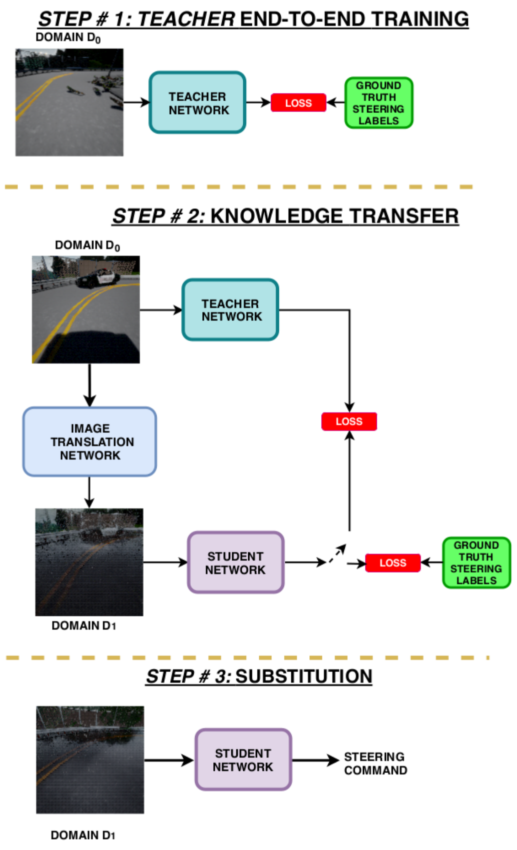

STEP # 1: TEACHER END-TO-END TRAINING
DOMAIN D0
TEACHER NETWORK
LOSS
GROUND TRUTH STEERING LABELS
STEP # 2: KNOWLEDGE TRANSFER
DOMAIN D0
TEACHER NETWORK
IMAGE TRANSLATION NETWORK
LOSS
STUDENT NETWORK
LOSS
GROUND TRUTH STEERING LABELS
DOMAIN D1
STEP # 3: SUBSTITUTION
STUDENT NETWORK
STEERING COMMAND
DOMAIN D1

Fig. 2: This figure gives a high level overview of the 3 steps for transferring knowledge between two domains D_0 and D_1 for the purpose of sensorimotor control. Ground truth steering data for only a limited number of images from domain D_0 is available. Details of the framework are provided in Section III.

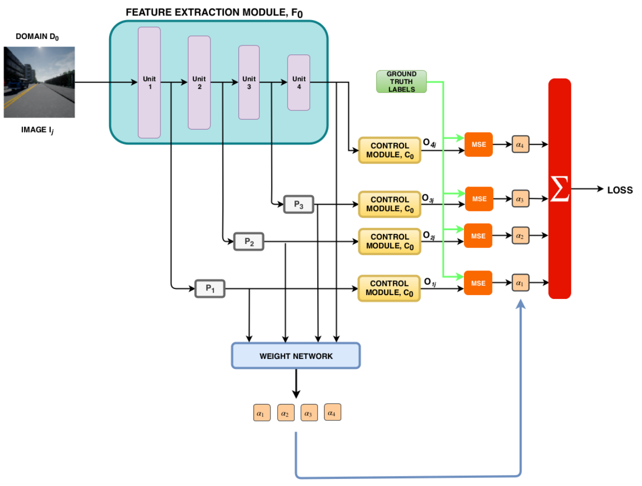


Fig. 3: The figure depicts the general architecture of the model comprised of the FEM and the auxiliary control modules.

is the mean squared error (MSE) between the ground truth label provided by the expert and that predicted by C_0. The overall loss is a weighted sum of the losses from each of the 4 control modules.

$$\mathcal{L} = \sum_{i=1}^{4} \alpha_i \cdot \mathcal{L}_i, \qquad \text{s.t.} \sum_{i=1}^{4} \alpha_i = 1,$$

where α_i, and $\mathcal{L}_i$ are the weighting and the error for the auxiliary network, obtained from the intermediate unit i of the FEM F_0. The error functions are calculated as follows:

$$\mathcal{L}_i = \frac{1}{N} \sum_{j=1}^{N} (y_j - O_{ij})^2,$$

where y_j is the ground truth steering angle obtained from the expert driver for a sample j and N denotes the number of total samples. O_{ij} is the output of the control module corresponding to the ith auxiliary network for the jth sample.

The weights α_i are themselves learned by a separate weight network. The auxiliary network that has the greatest contribution towards the overall output would also have the highest relative weight. This is important in case of limited data, wherein not all layers may be essential to train the model. In such a case the weights of the shallower auxiliary networks would be higher in comparison to the deeper auxiliary networks. Hence, a significant contribution towards the overall prediction would come from these shallow layers, thereby making the deeper layers effectively dormant. An extreme case would be when the labeled data is so small that even the first layer is enough to give a correct model prediction. In such a case, only $\alpha_1 = 1$ and all other $\alpha_i = 0$, for $i = 2, 3, 4$.

B. *Knowledge Transfer*

As described in step 2 of Figure 2, knowledge of ground truth labels from domain D_0 is transferred to domain D_1 using a teacher-student architecture. The output of the auxiliary networks acts as the teacher to provide supervised information to train the student.

We use the FEM, F_0 and control module, C_0 (combined, referred to as teacher) trained on images belonging to domain D_0, for which ground-truth steering labels are available, to transfer knowledge to a different combination of FEM, F_1 and control module, C_1 (referred to as student) for domain D_1, for which we have access to only unlabeled images. The subsequent procedure is detailed in the following steps:

1) Image I_0 belonging to domain D_0 is passed through an image-translation-network to generate image I_1 belonging to domain D_1 in a manner that only the semantic information is preserved but the weather condition is modified. [27], [29], [30] describe methods for training a translation network in an unsupervised manner using generative adversarial networks (GANs). We use [27] for our experiments. A positive implication of using these networks is that they preserve the semantics of the scene and hence the steering angle

label would also be the same.

2) **Hard loss:** If I_0 happens to have a ground truth (*hard*) label then the weights of the student network are updated with these labels and the loss is referred to as the *hard loss*. **Soft loss:** Otherwise, a forward pass can also be done by passing I_0 through the teacher. Meanwhile, the corresponding image I_1 is passed through the student network. The output of the teacher can then used as a soft target for updating the weights of the student via the soft loss. The overall loss is the weighted average of the soft and hard losses. The weights indicate the relative importance given to the soft targets in relation to the ground truth labels.

Note that the student network can be fed not only images from domain D_1 but rather multiple domains including domain D_0. Hence, the student network would not only be capable of predicting the steering for multiple domains but would act as a regularizer for better generalization (See P1 in Section V).

C. Substitution

This refers to step 3 described in Figure 2. At inference time, the teacher network can be substituted with the student network to predict the correct steering command on images from all domains which the student encountered during training.

IV. Experiments

In this section, we evaluate our approach on the CARLA simulator [24] version 0.8.2. It provides a total of 15 different weather conditions (labeled from 0 to 14) for two towns, *Town1* and *Town2*, respectively.

A. Evaluation Metrics

Finding appropriate evaluation metrics is rather challenging for navigation and driving tasks. There is no unique way to quantify these tasks. The authors of [25] discuss different problem statements for embodied navigation and present based on these discussions evaluation metrics for some standard scenarios. In [26], a more extensive study on evaluation metrics for vision-based driving models is carried out. In particular, they analyzed the difference between online and offline evaluation metrics for driving tasks. The preliminary results showed that driving models can have similar mean squared error (MSE) but drastically different driving performance. As a result of this, it is not straight forward to trivially link offline to online performance due to a low correlation between them. Nevertheless, the authors of [26] found that among offline metrics not requiring additional parameters, the mean absolute error between the driving commands and that predicted ones yields the highest correlation with online driving performance.

In addition to using this offline metric, we evaluate the online performance of the models when executing multiple and diverse turnings around corners, since it is a much more challenging task in comparison with simply moving in a straight line. The online performance is tested on the CARLA simulator across all the 15 weather conditions. For each weather condition, we evaluate the models for multiple different turns. In all experiments, the starting positions of the vehicle is just before the curve. The duration of the turn is fixed to 120 frames because it covers the entire curvature of the turn. We report the percentage of time the car remains within the driving lane as a measure of success.

B. Dataset

For collecting ground truth training data, we navigate through the city using the autopilot mode. To demonstrate the superiority of our method, we collect a limited sample size of 6500 images for weather condition 0 of which only 3200 are labeled with ground truth steering commands. Using our proposed method we aim to transfer knowledge to the remaining 14 weather scenarios. Also, note that none of the 6500 images have any semantic labels.

The 3200 sample images with ground truth data are only available for *Town2*, whereas all the offline and online evaluations are performed on *Town1*. To focus the attention on the effectiveness of our approach and preserve compatibility with prior work [1], [13], [26], the models are trained to predict the steering angle of the car while keeping the throttle fixed. The steering angles in CARLA are normalized to values between -1 and 1. The corresponding degrees for these normalized values depends on the vehicle being used. The default vehicle which we use for our experiments has a maximum steering angle of 70°.

C. Models

The offline and online performance of the models described in this section are given in Figure 4 and Table I, respectively. Figure 4 shows the plot of the mean absolute error between the actual steering command and that predicted by all of the models. Table I contains the percentage for which the ego-vehicle remains within the driving lane while making turning maneuvers executed by the models across the 15 weather scenarios.

Oracle: Steering labels for all weathers. Here we have assumed that we have access to the ground truth steering commands across all the 15 different weather conditions for *Town1*. Since we are also evaluating the models on *Town1* across all the weather conditions, we find in both the offline and online evaluation metrics that this model achieves the highest accuracy and hence it could serve as an upper bound for evaluating the other models along with our approach.

Model [9]: Steering and semantic labels for weather 0. Here we adopt the approach of [9], wherein the semantic labels of the images are additionally available for the 3200 labeled samples on weather 0. This additional information is used to first train what we refer to as the feature extraction module (FEM) in a supervised manner. The FEM module, in this case, is trained as an encoder-decoder architecture. The encoder encodes the input image into a lower-dimensional latent vector, while the decoder reconstructs the semantic map of the image from the latent vector. The latent vector is then used to train the control module from the ground truth steering labels. The FEM and control modules are hence trained independently and without any auxiliary networks. This FEM trained on the semantics of weather 0 is used as a teacher to train the student which is capable of producing the semantics of all the other 14 weather conditions. The authors of [9] used the method of [27] and provide 10 separate networks for translating from weather 0 to weathers 2, 3, 4, 6, 8, 9, 10, 11, 12, and 13, respectively. The translated images for each of the 10 weather conditions along with weather 0 are fed in equal proportion to train the student. We would particularly like to evaluate our method which does not have access to any semantic labels against this model. In addition to this, we also evaluate the performance of this method on the model provided by the paper, which was trained with more than 30000 samples from both *Town1* and *Town2*. The performance of this model on *Town1* is far superior since it was trained on much greater data and also had access to ground truth data from *Town1*.

Teacher: Steering angles for weather 0. This model is trained using only the available labeled data for weather 0 in an end-to-end manner. This model has a poor performance for the unseen weather conditions, particularly for conditions 3-14, which are considerably different in visual appearance compared to weather 0. Nevertheless, despite the poor performance this model can be used as a teacher to train the student for predicting the correct steering angles for weather conditions 1-14 for which no ground truth data exists. This approach is described in the next model. Also, note that the unlabeled data remains unutilized here.

Ours: Steering angles for weather 0. This model is trained using the method described in Section III, wherein knowledge is transferred from the teacher network trained on images and ground truth steering commands from weather 0 to the student network which is capable of handling images from all weathers 0-14. For a fair comparison against the model trained with semantic labels (Model [9], described earlier) we use the same data and generative models to translate even the unlabeled images to weathers 2, 3, 4, 6, 8, 9, 10, 11, 12, and 13, respectively. These generated images can then be fed to the student model for predicting the correct steering angles for all the 15 weather conditions.

V. Discussion

In this section, we discuss some critical insights on the experimental observations we obtained while evaluating the models. Here are some points we found worthwhile to provide some commentary based on the results provided in Figure 4 and Table I.

P1 - Better regularization: It is interesting to observe that the teacher model, trained only on the available 3200 labeled samples from *Town2* on weather 0 has a worse offline performance for *Town1* on weather 0 in comparison to our method. This seems to imply that our approach which has been trained on multiple kinds of weather has better generalization capabilities and can even outperform its teacher when evaluated in a different town. Hence, an additional positive consequence of training the student with

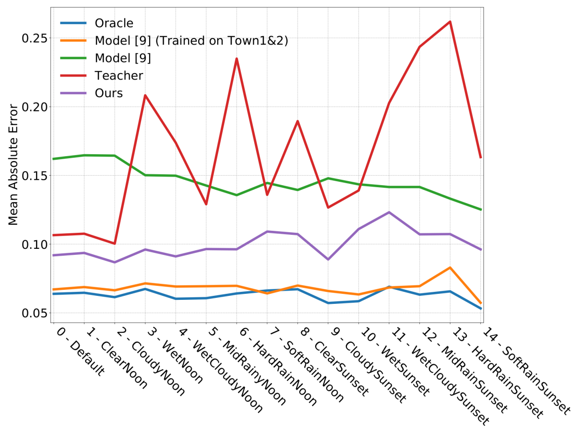


Fig. 4: This plot shows the mean absolute error between the actual steering angle and that predicted by the 5 different models (see subsection IV-C) on data collected across the 15 different weather conditions on *Town1*. Lower is better.

generated images from multiple diverse domains is that it acts as a regularizer tending to prevent overfitting to one specific domain.

P2 - Semantic inconsistency: Note that Model [9] which in addition to having the same data and labels as our approach has also access to ground truth semantic labels. Yet, its performance is significantly poor. Upon investigation, we found that due to the limited number of semantic labels, the FEM trained as an encoder-decoder architecture seemed to be overfitting to the available data. Hence, when tested on unseen environments, the semantic segmentation output of the module breaks. The latent vector representing these broken semantics is then fed to the control module, which is incapable of predicting the correct steering command. Figure 5 shows some sample images with the corresponding semantic segmentation outputs which are considerably different from the true semantics of the scene.

P3 - Modular training constraints: Furthermore, the modular approach of Model [9] wherein the FEM and control module are trained independently as opposed to an end-to-end model served to be a bottleneck in being able to learn the features universally. Also, an assumption to train the control module well is that the FEM would work perfectly well, which is not the case. Hence, the overall error of the modular pipeline would be an accumulation of the errors of the independent FEMs and control modules. We found that if we also shift the training of our approach to a modular one then performance deteriorates. This can be done in our approach by updating only the weights of the FEM of the student from the output features of the FEM of the teacher.

P4 - Auxiliary weights: To prevent overfitting of the models, trained on limited data we used a weighted sum of the outputs of the auxiliary layers. The weights themselves were learned as part of the training. Once training of our student model was complete, we found that more than 97 % of the weight was held by the first auxiliary network. This seemed to imply that only the first unit of the FEM is enough for

Method	Trained on	Weather Conditions															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	overall
Oracle	*Town1*	99.79	99.90	100	97.40	98.96	99.27	98.13	98.85	98.27	99.90	99.27	96.35	93.85	93.96	96.35	98.02
Model [9]	*Town1&2*	99.06	93.44	98.85	98.75	97.92	98.23	97.60	96.56	91.15	96.04	97.29	95.00	94.69	82.08	95.41	95.47
Model [9]	*Town2*	68.33	67.71	50.00	71.77	67.40	64.38	63.85	63.65	61.88	71.35	51.35	67.50	58.33	61.67	66.98	63.74
Teacher	*Town2*	92.19	92.40	82.12	44.38	51.77	73.65	32.50	61.56	49.48	80.10	60.63	48.54	35.20	34.27	50.52	59.29
Ours	*Town2*	93.96	95.21	81.25	99.90	100	94.17	90.42	79.69	77.19	86.77	84.58	65.63	68.54	58.44	80.73	83.77
Ours (Auxiliary network 1)	*Town2*	93.96	93.44	80.73	92.40	100	99.69	90.42	80.10	77.19	87.50	91.98	67.40	66.25	57.29	81.15	83.97

TABLE I: This table shows the percentage for which the ego-vehicle remains within the driving lane while executing a turn for the models across the 15 different weather scenarios on *Town1*. Higher is better.

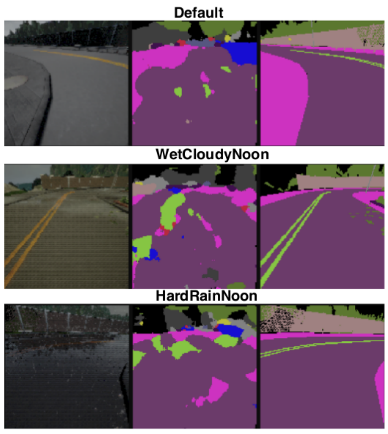


Fig. 5: This plot shows three sample images (column 1) with the corresponding semantic segmentation output by the model (column 2) for 3 different weathers. The segmentation produced by the model does not reflect the actual semantic characteristics of the scene (column 3).

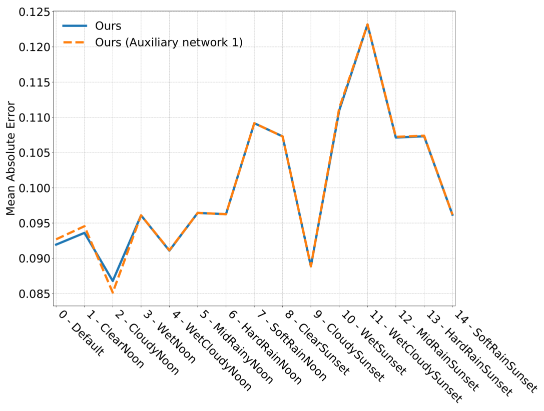


Fig. 6: This plot shows the mean absolute error between the ground truth steering label and that predicted by the two models. The **blue** curve is the weighted sum of all the 4 auxiliary networks of our model. The **orange** line depicts the output of only the first auxiliary network of our model.

predicting the steering command. Hence the remaining unit layers are not providing any additional information for the model. So we evaluated our model based on the output of the first auxiliary network rather than on the weighted sum of the 4 auxiliary networks. The online evaluation of this approach is given in Table I against the row labeled *Ours (Auxiliary network 1)*. It is interesting to note that this approach is comparable in its performance with the original one. Therefore, at test time we can prune the network to a smaller size by making predictions only based on the first auxiliary network and removing the remaining 3 auxiliary networks. This would result in less computation and faster inference.

P5 - Online vs. offline evaluation: Figure 6 shows an offline evaluation of the two variations of our method described in the previous point across the 15 weather conditions. Note that apart from weather 0, 1, and 2, the two curves are indistinguishable from one another. However, the online evaluation results do not correspond with this observation. For weathers 3, 5, 7, and 9-14 the online performance is different despite having the same offline metric. This confirms the intuition presented in [25] and the problems associated with evaluating embodied agents in offline scenarios. The topic of finding a correlation between offline evaluation metrics and online performance has therefore recently started to receive positive traction. It is therefore important to come up with a universal metric for evaluating various algorithms across the same benchmark. Due to the non-existence of such benchmarks, we created our own for the evaluation of the different approaches.

P6 - Activation maps: To understand the behavior of the model, which also works with only the first auxiliary network, we took the sum of the activation maps of the first unit of the FEM of the student and displayed it as a heatmap as shown in Figure 7 for a sample of 2 images. We see that the activation maps are most prominent in regions where there are lane markings, sidewalks, cars, or barriers. Knowing these cues seems to be enough for the network to take an appropriate driving decision in most of the cases. Therefore, the higher-level features determined by the preliminary layers of the model are already enough to detect these objects of interest.

VI. Conclusion

In this work, we showed how a teacher-student learning-based approach can leverage limited labeled data for transferring knowledge between multiple different domains. Our approach, specifically designed to work for sensorimotor control tasks, learns to accurately predict the steering angle under a wide range of conditions. Experimental results showed the effectiveness of the proposed method, even without having access to semantic labels as an intermediate representation between weather conditions. This framework

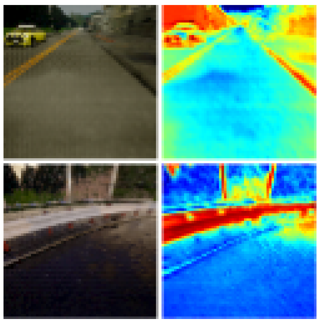


Fig. 7: This figure shows the sum of the activation maps displayed as a heatmap of the first unit of the FEM of the student model for a sample taken from 2 different weather conditions. The activation maps are more prominent in regions where there are lane markings, sidewalks boundaries, other vehicles, or barriers.

may be extendable to other application areas for which a certain domain has ground truth data and shares a common characteristic with other domains for which no labels are available.

References

[1] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to End Learning for Self-Driving Cars," *arXiv preprint arXiv:1604.07316*, 2016.

[2] E. Kaufmann, A. Loquercio, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep Drone Racing: Learning Agile Flight in Dynamic Environments," in *Conference on Robot Learning (CoRL)*, 2018.

[3] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-End Training of Deep Visuomotor Policies," *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1–40, 2016.

[4] A. Nair, D. Chen, P. Agrawal, P. Isola, P. Abbeel, J. Malik, and S. Levine, "Combining Self-Supervised Learning and Imitation for Vision-Based Rope Manipulation," in *International Conference on Robotics and Automation (ICRA)*, 2017.

[5] T. Zhang, Z. McCarthy, O. Jow, D. Lee, K. Goldberg, and P. Abbeel, "Deep Imitation Learning for Complex Manipulation Tasks from Virtual Reality Teleoperation," in *International Conference on Robotics and Automation (ICRA)*, 2018.

[6] G. Pascoe, W. Maddern, M. Tanner, P. Piniés, and P. Newman, "NID-SLAM: Robust Monocular SLAM using Normalised Information Distance," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[7] M. Müller, A. Dosovitskiy, B. Ghanem, and V. Koltun, "Driving Policy Transfer via Modularity and Abstraction," in *Conference on Robot Learning (CoRL)*, 2018.

[8] Y. You, X. Pan, Z. Wang, and C. Lu, "Virtual to Real Reinforcement Learning for Autonomous Driving," in *British Machine Vision Conference (BMVC)*, 2017.

[9] P. Wenzel, Q. Khan, D. Cremers, and L. Leal-Taixé, "Modular Vehicle Control for Transferring Semantic Information Between Weather Conditions Using GANs," in *Conference on Robot Learning (CoRL)*, 2018.

[10] J. Janai, F. Güney, A. Behl, and A. Geiger, "Computer Vision for Autonomous Vehicles: Problems, Datasets and State-of-the-Art," *arXiv preprint arXiv:1704.05519*, 2017.

[11] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[12] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The Cityscapes Dataset for Semantic Urban Scene Understanding," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[13] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end Learning of Driving Models from Large-scale Video Datasets," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[14] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[15] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual Worlds as Proxy for Multi-Object Tracking Analysis," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[16] S. R. Richter, Z. Hayder, and V. Koltun, "Playing for Benchmarks," in *International Conference on Computer Vision (ICCV)*, 2017.

[17] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for Data: Ground Truth from Computer Games," in *European Conference on Computer Vision (ECCV)*, 2016.

[18] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. V. Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Stanley : The Robot that Won the DARPA Grand Challenge," *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[19] D. A. Pomerleau, "ALVINN: An Autonomous Land Vehicle in a Neural Network," in *Neural Information Processing Systems (NIPS)*, 1989.

[20] F. Codevilla, M. Müller, A. Dosovitskiy, A. López, and V. Koltun, "End-to-end Driving via Conditional Imitation Learning," in *International Conference on Robotics and Automation (ICRA)*, 2018.

[21] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving," in *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[22] M. Müller, V. Casser, J. Lahoud, N. Smith, and B. Ghanem, "Sim4CV: A Photo-Realistic Simulator for Computer Vision Applications," *International Journal of Computer Vision (IJCV)*, 2018.

[23] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles," in *Field and Service Robotics (FSR)*, 2017.

[24] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator," in *Conference on Robot Learning (CoRL)*, 2017.

[25] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, and A. R. Zamir, "On Evaluation of Embodied Navigation Agents," *arXiv preprint arXiv:1807.06757*, 2018.

[26] F. Codevilla, A. M. Lopez, V. Koltun, and A. Dosovitskiy, "On Offline Evaluation of Vision-based Driving Models," in *European Conference on Computer Vision (ECCV)*, 2018.

[27] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks," in *International Conference on Computer Vision (ICCV)*, 2017.

[28] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised Image-to-Image Translation Networks," in *Neural Information Processing Systems (NIPS)*, 2017.

[29] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz, "Multimodal Unsupervised Image-to-Image Translation," in *European Conference on Computer Vision (ECCV)*, 2018.

[30] M. Li, H. Huang, L. Ma, W. Liu, T. Zhang, and Y.-G. Jiang, "Unsupervised Image-to-Image Translation with Stacked Cycle-Consistent Adversarial Networks," in *European Conference on Computer Vision (ECCV)*, 2018.

[31] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," *arXiv preprint arXiv:1503.02531*, 2015.

[32] C. Yang, L. Xie, S. Qiao, and A. Yuille, "Training Deep Neural Networks in Generations: A More Tolerant Teacher Educates Better Students," *arXiv preprint arXiv:1805.05551*, 2018.

[33] J. Shen, N. Vesdapunt, V. N. Boddeti, and K. M. Kitani, "In Teacher We Trust : Learning Compressed Models for Pedestrian Detection," *arXiv preprint arXiv:1612.00478*, 2016.

[34] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

A.3 GN-NET: THE GAUSS-NEWTON LOSS FOR MULTI-WEATHER RELOCALIZATION

Copyright

LUKAS VON STUMBERG, PATRICK WENZEL, QADEER KHAN, and DANIEL CREMERS

GN-Net: The Gauss-Newton Loss for Multi-Weather Relocalization

IEEE Robotics and Automation Letters (RA-L), 2020

DOI: 10.1109/LRA.2020.2965031

Abstract

Direct SLAM methods have shown exceptional performance on odometry tasks. However, they are susceptible to dynamic lighting and weather changes while also suffering from a bad initialization on large baselines. To overcome this, we propose GN-Net: a network optimized with the novel Gauss-Newton loss for training weather invariant deep features, tailored for direct image alignment. Our network can be trained with pixel correspondences between images taken from different sequences. Experiments on both simulated and real-world datasets demonstrate that our approach is more robust against bad initialization, variations in day-time, and weather changes thereby outperforming state-of-the-art direct and indirect methods. Furthermore, we release an evaluation benchmark for relocalization tracking against different types of weather. Our benchmark is available at this https://vision.in.tum.de/gn-net.

Individual Contributions

Problem definition	*contributed*
Literature survey	*significantly contributed*
Algorithm development	*significantly contributed*
Method implementation	*significantly contributed*
Experimental evaluation	*significantly contributed*
Preparation of the manuscript	*significantly contributed*

Notice

In accordance with the *IEEE Thesis / Dissertation Reuse Permissions*, we include the accepted version of the original publication [3] in the following.

GN-Net: The Gauss-Newton Loss for Multi-Weather Relocalization

Lukas von Stumberg[1,2*], Patrick Wenzel[1,2*], Qadeer Khan[1,2], and Daniel Cremers[1,2]

***Abstract*—Direct SLAM methods have shown exceptional performance on odometry tasks. However, they are susceptible to dynamic lighting and weather changes while also suffering from a bad initialization on large baselines. To overcome this, we propose GN-Net: a network optimized with the novel Gauss-Newton loss for training weather invariant deep features, tailored for direct image alignment. Our network can be trained with pixel correspondences between images taken from different sequences. Experiments on both simulated and real-world datasets demonstrate that our approach is more robust against bad initialization, variations in day-time, and weather changes thereby outperforming state-of-the-art direct and indirect methods. Furthermore, we release an evaluation benchmark for relocalization tracking against different types of weather. Our benchmark is available at https://vision.in.tum.de/gn-net.**

***Index Terms*—Localization, Visual Learning, SLAM**

I. Introduction

In recent years, very powerful visual SLAM algorithms have been proposed [1], [2]. In particular, direct visual SLAM methods have shown great performance, outperforming indirect methods on most benchmarks [3], [4], [5]. They directly leverage the brightness data of the sensor to estimate localization and 3D maps rather than extracting a heuristically selected sparse subset of feature points. As a result, they exhibit a boost in precision and robustness. Nevertheless, compared to indirect methods, direct methods suffer from two major drawbacks:

1) Direct methods need a good initialization, making them less robust for large baseline tracking or cameras with a low frame rate.
2) Direct methods cannot handle changing lighting/weather conditions. In such situations, their advantage of being able to pick up very subtle brightness variations becomes a disadvantage to the more lighting invariant features.

In the last years, researchers have tackled the multiple-daytime tracking challenge with deep learning approaches that are designed to convert nighttime images to daytime images *e.g.* using GANs [6], [7], [8]. While this improves the robustness to changing lighting, one may ask why images should be the best input representation. Could there be better alternate representations?

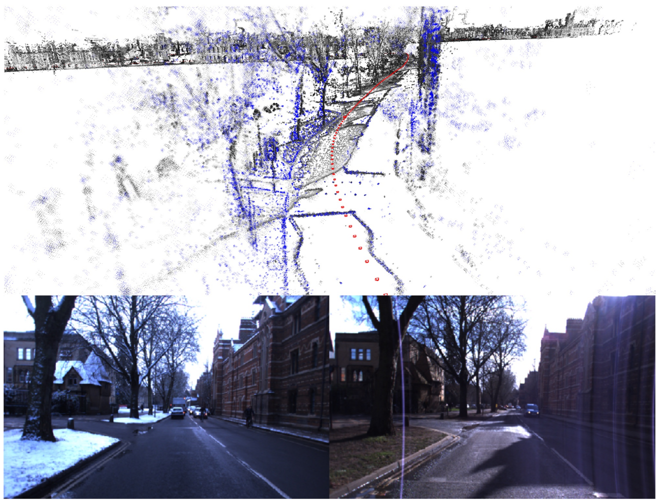


Fig. 1: We relocalize a snowy sequence from the Oxford RobotCar dataset in a pre-built map created using a sunny weather condition. The points from the prior map (gray) align well with the new points from the current run (blue), indicating that the relocalization is indeed accurate.

This paper addresses the problem of adapting direct SLAM methods to challenging lighting and weather conditions. In this work, we show how to convert images into a multi-dimensional feature map which is invariant to lighting/weather changes and has by construction a larger basin of convergence. Thereby we overcome the aforementioned problems simultaneously. The deep features are trained with a novel Gauss-Newton loss formulation in a self-supervised manner. We employ a Siamese network trained with labels obtained either from simulation data or any state-of-the-art SLAM algorithm. This eliminates the additional cost of human labeling that is typically necessary for training a neural network. We exploit the probabilistic interpretation of the commonly used Gauss-Newton algorithm for direct image alignment. For this, we propose the Gauss-Newton loss which is designed to maximize the probability of identifying the correct pixel correspondence. The proposed loss function thereby enforces a feature representation that is designed to admit a large basin of convergence for the subsequent Gauss-Newton optimization. The superiority of our method stems from its ability to generate these multi-channel, weather-invariant deep features that facilitate relocalization across different weathers. Figure 1

Manuscript received: September 10, 2019; Revised November 26, 2019; Accepted December 23, 2019.

This paper was recommended for publication by Editor Sven Behnke upon evaluation of the Associate Editor and Reviewers' comments.

*These authors contributed equally.

[1]All authors are with the Computer Vision Group, Technical University of Munich, Germany. {stumberg, wenzel, khan, cremers}@in.tum.de

[2]All authors are with Artisense Corporation.

Digital Object Identifier (DOI): see top of this page.

shows how our method can successfully relocalize a snowy sequence in a pre-built map created using a sunny sequence.

In common benchmarks [9], localizing accurately in a pre-built map has been tackled by finding nearby images (*e.g.* by using NetVLAD [10]) and tracking the relative pose (6DOF) between them. However, we propose to split this into two separate tasks. In this work, we focus on the second challenge which we refer to as *relocalization tracking*. This way, we can evaluate its performance in isolation. This is formalized to what we refer to as *relocalization tracking*. Since there is no publicly available dataset to evaluate *relocalization tracking* performance across multiple types of weathers, we are releasing an evaluation benchmark having the following 3 attributes:

- It contains sequences from multiple different kinds of weathers.
- Pixel-wise correspondences between sequences are provided for both simulated and real-world datasets.
- It decouples relocalization tracking from the image retrieval task.

The challenge here in comparison with normal pose estimation datasets [11], [12] is that the images involved are usually captured at different daytimes/seasons and there is no good initialization of the pose. We summarize the main contributions of our paper as:

- We derive the Gauss-Newton loss formulation based on the properties of direct image alignment and demonstrate that it improves the robustness to large baselines and illumination/weather changes.
- Our experimental evaluation shows, that GN-Net outperforms both state-of-the-art direct and indirect SLAM methods on the task of *relocalization tracking*.
- We release a new evaluation benchmark for the task of *relocalization tracking* with ground-truth poses. It is collected under dynamic conditions such as illumination changes, and different weathers. Sequences are taken from the the CARLA [13] simulator as well as from the Oxford RobotCar dataset [14].

II. Related Work

We review the following main areas of related work: visual SLAM, visual descriptor learning, deep direct image alignment, and image-based relocalization in SLAM.

Direct versus indirect SLAM methods: Most existing SLAM systems that have used feature descriptors are based on traditional manual feature engineering, such as ORB-SLAM [2], MonoSLAM [15], and PTAM [1].

An alternative to feature-based methods is to skip the preprocessing step of the raw sensor measurements and rather use the pixel intensities directly. Popular direct visual methods are DTAM [16], LSD-SLAM [3], DSO [5], and PhotoBundle [4]. However, the main limitation of direct methods is the *brightness constancy* assumption which is rarely fulfilled in any real-world robotic application [17]. The authors of [18] propose to use binary feature descriptors for direct tracking called Bit-planes. While improving the robustness to bad lighting situations it was also found that Bit-planes have a smaller convergence basin than intensities. This makes their method less robust to bad initialization. In contrast, the features we propose *both* improve robustness to lighting and the convergence basin.

Visual descriptor learning: Feature descriptors play an important role in a variety of computer vision tasks. For example, [19] proposed a novel correspondence contrastive loss which allows for faster training and demonstrates their effectiveness for both geometric and semantic matching across intra-class shape or appearance variations. In [20], a deep neural network is trained using a contrastive loss to produce viewpoint- and lighting-invariant descriptors for single-frame localization. The authors of [21] proposed a CNN-based model that learns local patterns for image matching without a global geometric model. [22] uses convolutional neural networks to compute descriptors which allow for efficient detection of poorly textured objects and estimation of their 3D pose. In [23], the authors propose to train features for optical flow estimation using a Hinge loss based on correspondences. In contrast to our work, their loss function does not have a probabilistic derivation and they do not apply their features to pose estimation. [24] uses deep learning to improve SLAM performance in challenging situations. They synthetically create images and choose the one with most gradient information as the ground-truth for training. In contrast to them, we do not limit our network to output images similar to the real world. In [25], the authors compare dense descriptors from a standard CNN, SIFT, and normal image intensities for dense Lucas-Kanade tracking. There, it can be seen that grayscale values have a better convergence basin than the other features, which is something we overcome with our approach.

Deep direct image alignment: BA-NET [26] introduces a network architecture to solve the structure from motion (SfM) problem via feature-metric bundle adjustment. Unlike the BA-NET, instead of predicting the depth and the camera motion simultaneously, we propose to only train on correspondences obtained from a direct SLAM system. The advantage is that correspondences are oftentimes easier to obtain than accurate ground-truth poses. Furthermore, we combine our method with a state-of-the-art direct SLAM system and utilize its depth estimation, whereas BA-NET purely relies on deep learning. RegNet [27] is another line of work which tries to replace the handcrafted numerical Jacobian by a learned Jacobian with the help of a depth prediction neural network. However, predicting a dense depth map is often inaccurate and computationally demanding. The authors of [28] propose to use a learning-based inverse compositional algorithm for dense image alignment. The drawback of this approach is that the algorithm is very sensitive to the data distribution and constrained towards selecting the right hyperparameters. In [29] they use high-dimensional features in a direct image alignment framework for monocular VO. In contrast to us, they only use already existing features and do not apply them for relocalization.

Relocalization: An important task of relocalization is to approximate the pose of an image by simply querying the most similar image from a database [30], [31]. However, this has only limited accuracy unless the 6DOF pose between the queried and the current image is estimated in a second step. Typically, this works by matching 2D-3D correspondences between an image and a point cloud and estimating the pose using indirect image alignment [32]. In contrast, we propose to use direct image alignment paired with deep features.

Relocalization benchmarks: The authors of [9] have done sequence alignment on the Oxford RobotCar dataset, however, they have not made the matching correspondences public. The Photo Tourism [33] is another dataset providing images and ground-truth correspondences of popular monuments from different camera angles and across different weather/lighting conditions. However, since the images are not recorded as a sequence, relocalization tracking is not possible. Furthermore, their benchmark only supports the submission of features rather than poses, thereby restricting evaluation to only indirect methods.

III. Deep Direct SLAM

In this work, we argue that a network trained to output features which produce better inputs for direct SLAM as opposed to normal images should have the following properties:

- Pixels corresponding to the same 3D point should have similar features.
- Pixels corresponding to different 3D points should have dissimilar features.
- When starting in a vicinity around the correct pixel, the Gauss-Newton algorithm should move towards the correct solution.

For optimizing the last property, we propose the novel Gauss-Newton loss which makes use of the probabilistic background of the Gauss-Newton algorithm for direct image alignment. The final loss is a weighted sum of the pixel-wise contrastive loss and the Gauss-Newton loss.

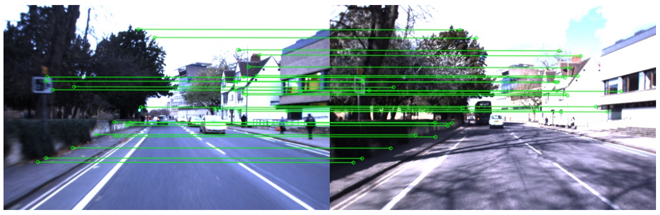


Fig. 2: This figure shows training correspondences between a pair of images from our benchmark.

Architecture: We are interested in learning a non-linear mapping, which maps images, $\mathbb{R}^{W\times H\times C}$ to a dense visual descriptor space, $\mathbb{R}^{W\times H\times D}$, where each pixel is represented by a D-dimensional vector. The training is performed by a Siamese encoder-decoder structured network, where we feed a pair of images, $\mathbf{I}_a$ and $\mathbf{I}_b$, producing multi-scale feature pyramids $\mathbf{F}_a^l$ and $\mathbf{F}_b^l$, where l represents the level of the decoder. For each image pair, we use a certain number of matches, denoted by N_{pos}, and a certain number of non-matches, denoted by N_{neg}. A pixel $\mathbf{u}_a \in \mathbb{R}^2$ from image $\mathbf{I}_a$ is considered to be a positive example if the pixel $\mathbf{u}_b \in \mathbb{R}^2$ from image $\mathbf{I}_b$ corresponds to the same 3D vertex (Figure 2). We make use of the inherent multi-scale hierarchy of the U-Net [34] architecture to apply the different loss terms from coarser to finer scaled pyramid levels. With this approach, our learned features will have a larger convergence radius for visual SLAM methods.

Pixelwise contrastive loss: The pixelwise contrastive loss attempts to minimize the distance between positive pairs, and maximize the distance between negative pairs. It can be computed as follows: $\mathcal{L}_{\text{contrastive}}(\mathbf{F}_a, \mathbf{F}_b, l) = \mathcal{L}_{\text{pos}}(\mathbf{F}_a, \mathbf{F}_b, l) + \mathcal{L}_{\text{neg}}(\mathbf{F}_a, \mathbf{F}_b, l)$.

$$\mathcal{L}_{\text{pos}}(\mathbf{F}_a, \mathbf{F}_b, l) = \frac{1}{N_{\text{pos}}} \sum_{N_{\text{pos}}} D_{\text{feat}}^2 \tag{1}$$

$$\mathcal{L}_{\text{neg}}(\mathbf{F}_a, \mathbf{F}_b, l) = \frac{1}{N_{\text{neg}}} \sum_{N_{\text{neg}}} \max(0, M - D_{\text{feat}})^2 \tag{2}$$

where $D_{\text{feat}}(\cdot)$ is the L_2 distance between the feature embeddings: $D_{\text{feat}} = ||\mathbf{F}_a^l(\mathbf{u}_a) - \mathbf{F}_b^l(\mathbf{u}_b)||_2$ and M is the margin and set to 1.

Gauss-Newton algorithm for direct image alignment: Our learned deep features are ultimately applied to pose estimation. This is done using direct image alignment but generalized to a multi-channel feature map $\mathbf{F}$ with D channels. The input to this algorithm is a reference feature map $\mathbf{F}$ with known depths for some pixels in the image, and a target feature map $\mathbf{F}'$. The output is the predicted relative pose $\boldsymbol{\xi}$. Starting from an initial guess the following steps are performed iteratively:

1) All points $\mathbf{p}_i$ with known depth values are projected from the reference feature map $\mathbf{F}$ into the target feature map $\mathbf{F}'$ yielding the point $\mathbf{p'}_i$. For each of them a residual vector $\mathbf{r} \in \mathbb{R}^D$ is computed, enforcing that the reference pixel and the target pixel should be similar:

$$\mathbf{r}_i(\mathbf{p}_i, \mathbf{p'}_i) = \mathbf{F}'(\mathbf{p'}_i) - \mathbf{F}(\mathbf{p}_i) \tag{3}$$

2) For each residual the derivative with respect to the relative pose is:

$$\mathbf{J}_i = \frac{d\mathbf{r}_i}{d\boldsymbol{\xi}} = \frac{d\mathbf{F}'(\mathbf{p'}_i)}{d\mathbf{p'}_i} \cdot \frac{d\mathbf{p'}_i}{d\boldsymbol{\xi}} \tag{4}$$

Notice that the reference point $\mathbf{p}_i$ does not change for different solutions $\boldsymbol{\xi}$, therefore it does not appear in the derivative.

3) Using the stacked residual vector $\mathbf{r}$, the stacked Jacobian $\mathbf{J}$, and a diagonal weight matrix $\mathbf{W}$, the Gaussian system and the step $\boldsymbol{\delta}$ is computed as follows:

$$\mathbf{H} = \mathbf{J}^T\mathbf{W}\mathbf{J} \text{ and } \mathbf{b} = -\mathbf{J}^T\mathbf{W}\mathbf{r} \text{ and } \boldsymbol{\delta} = \mathbf{H}^{-1}\mathbf{b} \tag{5}$$

Note that this derivation is equivalent to normal direct image alignment (as done in the frame-to-frame tracking from DSO) when replacing $\mathbf{F}$ with the image $\mathbf{I}$. In the computation of the Jacobian the numerical derivative of the features $\frac{d\mathbf{F}'(\mathbf{p'}_i)}{d\mathbf{p'}_i}$ is used. As typical images are extremely non-convex this derivative is only valid in a small vicinity (usually 1-2 pixels) around the current solution which is the main reason why direct image alignment needs a good initialization. To partially overcome this, a pyramid scheme is often used. Usually tracking on multiple channels instead of one can decrease the convergence radius ([18], [25]). However, in our case, we train the feature maps to in fact have a larger convergence basin than images by enforcing smoothness in the vicinity of the correct correspondence.

Gauss-Newton on individual pixels: Instead of running the Gauss-Newton algorithm on the 6DOF pose we can instead use it on each point $\mathbf{p}_i$ individually (which is similar to the Lucas-Kanade algorithm [35]). Compared to direct image alignment, this optimization problem has the same residual, but the parameter being optimized is the point position instead of the relative pose. In this case, the Hessian will be a 2-by-2 matrix and the step $\boldsymbol{\delta}$ can simply be added to the current pixel position (we leave out $\mathbf{W}$ for simplicity):

$$\mathbf{J'}_i = \frac{d\mathbf{F}'(\mathbf{p'}_i)}{d\mathbf{p'}_i} \text{ and } \mathbf{H}'_i = \mathbf{J'}_i^T \mathbf{J}_i \text{ and } \mathbf{b'}_i = \mathbf{J'}_i^T \mathbf{r}_i \tag{6}$$

These individual Gauss-Newton systems can be combined with the system for 6DOF pose estimation (Equation (5)) using:
$\mathbf{H} = \sum_i \left(\frac{d\mathbf{p'}_i}{d\boldsymbol{\xi}}\right)^T \mathbf{H'}_i \left(\frac{d\mathbf{p'}_i}{d\boldsymbol{\xi}}\right)$ and $\mathbf{b} = \sum_i \left(\frac{d\mathbf{p'}_i}{d\boldsymbol{\xi}}\right)^T \mathbf{b'}_i$.

The difference between our simplified systems and the one for pose estimation is only the derivative with respect to the pose, which is much smoother than the image derivative [5].

This means that if the Gauss-Newton algorithm performs well on individual pixels it will also work well on estimating the full pose. Therefore, we propose to train a neural network on correspondences which are easy to obtain, *e.g.* using a SLAM method, and then later apply it for pose estimation.

We argue that training on these individual points is superior to training on the 6DOF pose. The estimated pose can be correct even if some points contribute very wrong estimates. This increases robustness at runtime but when training we want to improve the information each point provides. Also, when training on the 6DOF pose we only have one supervision signal for each image pair, whereas when training on correspondences we have over a thousand signals. Hence, our method exhibits exceptional generalization capabilities as shown in the results section.

The probabilistic Gauss-Newton loss: The linear system described in Equation (6) defines a 2-dimensional Gaussian probability distribution. The reason is that the Gauss-Newton algorithm tries to find the solution with maximum probability in a least squares fashion. This can be derived using the negative log-likelihood of the Gaussian distribution:

$$E(\mathbf{x}) = -\log f_X(\mathbf{x}) = \tag{7}$$

$$\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}) + \log\left(2\pi\sqrt{|\boldsymbol{\Sigma}|}\right) = \tag{8}$$

$$\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \mathbf{H}(\mathbf{x}-\boldsymbol{\mu}) + \log(2\pi) - \frac{1}{2}\log(|\mathbf{H}|) \tag{9}$$

where $\mathbf{x}$ is a pixel position and $\boldsymbol{\mu}$ is the mean.

In the Gauss-Newton algorithm the mean (which also corresponds to the point with maximum probability) is computed with $\boldsymbol{\mu} = \mathbf{x}_s + \boldsymbol{\delta}$, where the $\boldsymbol{\delta}$ comes from Equation (5) and $\mathbf{x}_s$ denotes the start point. To derive this, only the first term is used (because the latter parts are constant for all solutions $\mathbf{x}$). In our case, however, the second term is very relevant, because the network can influence both $\boldsymbol{\mu}$ and $\mathbf{H}$.

This derivation shows, that $\mathbf{H}, \mathbf{b}$ as computed in the GN-algorithm, also define a Gaussian probability distribution with mean $\mathbf{x}_s + \mathbf{H}^{-1}\mathbf{b}$ and covariance $\mathbf{H}^{-1}$.

When starting with an initial solution $\mathbf{x}_s$ the network should assign maximal probability to the pixel that marks the correct correspondence. With $\mathbf{x}$ being the correct correspondence, we therefore use $E(\mathbf{x})$ = Equation (9) as our loss function which we call the *Gauss-Newton loss* (see Algorithm 1).

Algorithm 1 Compute Gauss-Newton loss

```
$\mathbf{F}_a \leftarrow$ network$(\mathbf{I}_a)$
$\mathbf{F}_b \leftarrow$ network$(\mathbf{I}_b)$
$e \leftarrow 0$                                          ▷ Total error
for all correspondences $\mathbf{u}_a, \mathbf{u}_b$ do
    $\mathbf{f}_t \leftarrow \mathbf{F}_a(\mathbf{u}_a)$                          ▷ Target feature
    $\mathbf{x}_s \leftarrow \mathbf{u}_b +$ rand(vicinity)          ▷ Compute start point
    $\mathbf{f}_s \leftarrow \mathbf{F}_b(\mathbf{x}_s)$
    $\mathbf{r} \leftarrow \mathbf{f}_s - \mathbf{f}_t$                                ▷ Residual
    $\mathbf{J} \leftarrow \frac{d\mathbf{F}_b}{d\mathbf{x}_s}$                    ▷ Numerical derivative
    $\mathbf{H} \leftarrow \mathbf{J}^T\mathbf{J} + \epsilon \cdot \mathrm{Id}$     ▷ Added epsilon for invertibility
    $\mathbf{b} \leftarrow \mathbf{J}^T r$
    $\boldsymbol{\mu} \leftarrow \mathbf{x}_s - \mathbf{H}^{-1}\mathbf{b}$
    $e_1 \leftarrow \frac{1}{2}(\mathbf{u}_b - \boldsymbol{\mu})^T\mathbf{H}(\mathbf{u}_b - \boldsymbol{\mu})$        ▷ First error term
    $e_2 \leftarrow \log(2\pi) - \frac{1}{2}\log(|\mathbf{H}|)$          ▷ Second error term
    $e \leftarrow e + e_1 + e_2$
end for
```

In the algorithm, a small number ϵ is added to the diagonal of the Hessian, to ensure it is invertible.

Analysis of the Gauss-Newton loss: By minimizing Equation (9) the network has to maximize the probability density of the correct solution. As the integral over the probability densities always has to be 1, the network has the choice to either focus all the density on a small set of solutions (with more risk of being penalized if this solution is wrong), or to distribute the density to more solutions which in turn will have a lower individual density. By maximizing the probability of the correct solution, the network is incentivized to improve the estimated solution and its certainty.

This is also reflected in the two parts of the loss. The first term $e_1 = \frac{1}{2}(\mathbf{u}_b - \boldsymbol{\mu})^T\mathbf{H}(\mathbf{u}_b - \boldsymbol{\mu})$ penalizes deviations between the estimated and the correct solution, scaled with the Hessian $\mathbf{H}$. The second term $e_2 = \log(2\pi) - \frac{1}{2}\log(|\mathbf{H}|)$ is large if the network does not output enough certainty for its solution. This means that the network can reduce the first error term e_1 by making $\mathbf{H}$ smaller. As a consequence, the second error term will be increased, as this will also reduce the determinant of $\mathbf{H}$. Notice also that this can be done in both dimensions independently. The network has the ability to output a large uncertainty in one direction, but a small uncertainty in the other direction. This is one of the traditional advantages of direct methods which are naturally able to utilize also lines instead of just feature points.

From Equation (9) it can be observed that the predicted uncertainty depends only on the numerical derivative of the target image at the start position. The higher the gradients the higher the predicted certainty. In DSO this is an unwanted effect that is counteracted by the gradient-dependent weighting

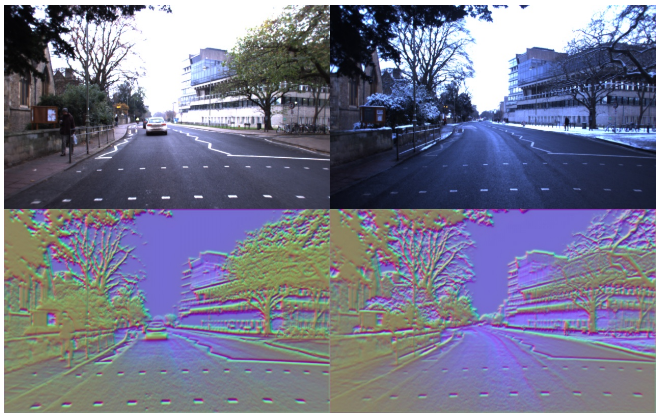


Fig. 3: This figure shows images and their corresponding feature maps predicted by our GN-Net for the Oxford RobotCar dataset. Each column depicts the image and feature map for a sample taken from 2 different sequences. Despite lighting and weather changes, the feature maps are robust to these variations. The visualization of the features shows the high-dimensional descriptors reduced to 3D through PCA.

applied to the cost-function [5, Equation (7)]. In our case, however, it gives the network the possibility to express its certainty and incentivizes it to output discriminative features.

Upon training the network with our loss formulation, we observe that the features are very similar despite being generated from images taken from sequences with different lighting/weather conditions, as shown in Figure 3.

IV. Relocalization Tracking Benchmark

Previous tasks for localization/odometry can primarily be divided into two categories:

- Odometry datasets [11], [12], where there is a continuous stream of images (sometimes combined with additional sensor data like IMUs).
- Image collections where individual images are usually further apart from each other in space/time [36], [9].

We argue that for several applications a combination of these two tasks which we refer to as *relocalization tracking* is a more realistic scenario. The idea is that the algorithm has two inputs:

1) An image sequence (like a normal odometry dataset).
2) A collection of individual images (possibly with different weathers/times), each of which shall be tracked against one specific image from point 1.

The algorithm is supposed to track the normal sequential image sequence and at the same time perform tracking of the images in point 2. The advantage of this task is that the used algorithm can utilize the temporally continuous sequence from point 1 to compute accurate depth values for a part of the image (using a standard visual odometry method), which can then be used to improve the tracking of the individual images of point 2.

This task is very realistic as it comes up when tracking an image sequence and at the same time trying to relocalize this sequence in a prior map. A similar challenge occurs by trying to merge multiple maps from different times. In both cases, one has more information than just a random collection of images. It is important to reiterate here that the task of finding relocalization candidates is not considered but rather tracking them with maximum accuracy/robustness is the focus. This is because our benchmark decouples image retrieval from tracking.

We have created a benchmark for *relocalization tracking* using the CARLA simulator and the Oxford RobotCar dataset. Our benchmark includes ground-truth poses between different sequences for both training, validation, and testing. We focus on the use-case of relocalization in the context of autonomous driving. Therefore, our datasets contain limited point-of-view changes but strong lighting and weather changes.

CARLA: For synthetic evaluations, we use CARLA version 0.8.2. We collect data for 3 different weather conditions representing *WetNoon*, *SoftRainNoon*, and *WetCloudySunset*. We recorded the images at a fixed framerate of 10 frames per second (FPS). At each time step, we record images and its corresponding dense depth map from 6 different cameras with different poses rendered from the simulation engine, which means that the poses in the benchmark are not limited to just 2DOF. The images and the dense depth maps are of size 512×512. For each weather condition, we collected 3 different sequences comprising 500-time steps with an average distance of 1.6m. This is done for training, validation, and testing, meaning there are 27 sequences, containing 6 cameras each. Training, validation, and test sequences were all recorded in different parts of the CARLA town. We have generated the test sequences after all hyperparameter tuning of our method was finished, meaning we had no access to the test data when developing the method. In accordance, we shall withhold the ground-truth for the test sequences.

Oxford RobotCar: Creating a multi-weather benchmark for this dataset imposes various challenges because the GPS-based ground-truth is very inaccurate. To find the relative poses between images from different sequences we have used the following approach. For pairs of images from two different sequences, we accumulate the point cloud captured by the 2D lidar for 60 meters using the visual odometry result provided by the Oxford dataset. The resulting two point clouds are aligned with the global registration followed by ICP alignment using the implementation of Open3D [37]. We provide the first pair of images manually and the following pairs are found using the previous solution. We have performed this alignment for the following sequences: *2014-12-02-15-30-08 (overcast)* and *2015-03-24-13-47-33 (sunny)* for training. For testing, we use the reference sequence *2015-02-24-12-32-19 (sunny)* and align it with the sequences *2015-03-17-11-08-44 (overcast)*, *2014-12-05-11-09-10 (rainy)*, and *2015-02-03-08-45-10 (snow)*. The average relocalization distance across all sequences is 0.84m.

V. Experimental Evaluation

We perform our experiments on the *relocalization tracking* benchmark described in Section IV. We demonstrate the multi-weather relocalization performance on both the CARLA and the Oxford RobotCar dataset. For the latter, we show that our method even generalizes well to unseen weather conditions like rain or snow while being trained only on the sunny and overcast conditions. Furthermore, a qualitative relocalization demo[1] on the Oxford RobotCar dataset is provided, where we demonstrate that our GN-Net can facilitate precise relocalization between weather conditions.

We train our method using sparse depths created by running Stereo DSO on the training sequences. We use intra-sequence correspondences calculated using the DSO depths and the DSO pose. Meanwhile, inter-sequence correspondences are obtained using DSO depths and the ground-truth poses provided by our benchmark. The ground truth poses are obtained via Lidar alignment for Oxford and directly from the simulation engine for CARLA as explained in Section IV. Training is done from scratch with randomly initialized weights and an ADAM optimizer with a learning rate of 10^{-6}. The image pair fed to the Siamese network is randomly selected from any of the training sequences while ensuring that the images in the pair do not differ by more than 5 keyframes. Each branch of the Siamese network is a modified U-Net architecture with shared weights. Further details of the architecture and training can be found in the supplementary material[1]. Note that at inference time, only one image is needed to extract the deep visual descriptors, used as input to the SLAM algorithm. While in principle, our approach can be deployed in conjunction with any direct method, we have coupled our deep features with Direct Sparse Odometry (DSO).

We compare to state-of-the-art **direct** methods:

Stereo Direct Sparse Odometry (DSO) [38]: Whenever there is a relocalization candidate for a frame we ensure that the system creates the corresponding keyframe. This candidate is tracked using the *coarse tracker*, performing direct image alignment in a pyramid scheme. We use the identity as initialization without any other random guesses for the pose.

GN-Net (Ours): Same as with DSO, however, for relocalization tracking, we replace the grayscale images with features created by our GN-Net on all levels of the feature pyramid. The network is trained with the Gauss-Newton loss formulation described in Section III.

We also compare to state-of-the-art **indirect** methods:

ORB-SLAM [32]: For relocalization tracking, we use the standard feature-based 2-frame pose optimization also used for frame-to-keyframe tracking. We have also tried the RANSAC scheme implemented in ORB-SLAM for relocalization, however, it yielded worse results overall. Thus we will report only the default results.

D2-Net [39], SuperPoint [40]: For both methods we use the models provided by the authors. The relative pose is estimated using the OpenCV implementation of the PnP algorithm in a RANSAC scheme.

We also evaluated the Deeper Inverse Compositional Algorithm [28] on the *relocalization tracking* benchmark. However, the original implementation didn't converge despite multiple training trials with different hyperparameters.

[1]https://vision.in.tum.de/gn-net.

For all our quantitative experiments we plot a cumulative distribution of the relocalization error, which is the norm of the translation between the estimated and the correct solution in meters. For each relocalization error between 0 and 1 meter, it plots the percentage of relocalization candidates that have been tracked with at least this accuracy.

A. Quantitative multi-weather evaluation

We demonstrate the relocalization tracking accuracy on our new benchmark across different weathers. For these experiments, tracking is performed only across sequences with a different weather condition.

CARLA: For this experiment, we train on the training sequences provided by our benchmark. For all learning-based approaches, the best epoch is selected using the relocalization tracking performance on the validation set. The results on the test data are shown in the supplementary[1].

Oxford RobotCar: We train on the sunny and overcast condition correspondences provided by our *relocalization tracking* benchmark for the Oxford dataset. For the learning-based methods, we select the best epoch based on the relocalization tracking performance on the training set. We use the same hyperparameters that were found using the CARLA validation set. We show the results on the test data in Figure 4. Our method significantly outperforms the baselines. The Gauss-Newton loss has a large impact as compared to the model trained with only the contrastive loss.

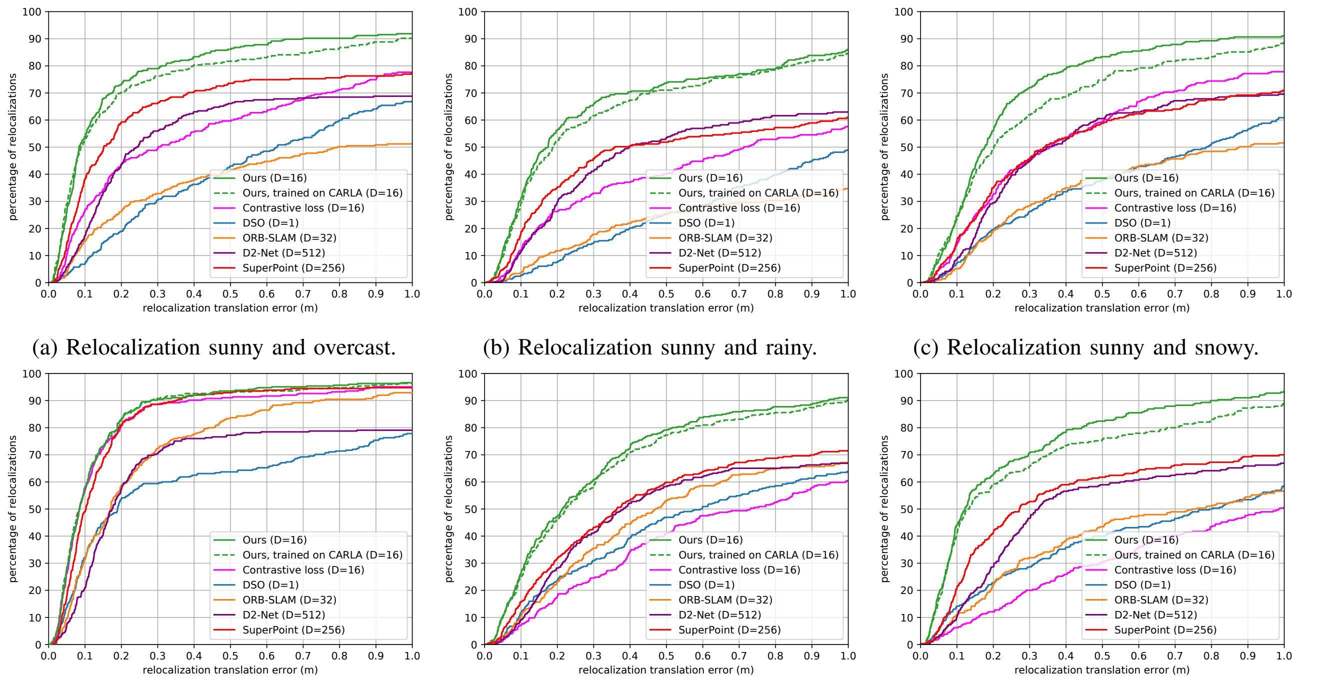
Figures 4b-f show how well our model generalizes to unseen weather conditions. Despite being trained only on two sequences with overcast and sunny conditions the results for tracking against a rainy and a snowy sequence are almost the same. Interestingly our model which was trained only on the CARLA benchmark outperforms all baselines significantly.

B. Qualitative multi-weather evaluation

Finally, we show a relocalization demo comparing our GN-Net to DSO. For this, we load a point cloud from a sequence recorded in the sunny condition and relocalize against sequences from rainy and snowy conditions. For each keyframe, we try to track it against the nearest keyframe in the map according to the currently estimated transformation between the trajectory and the map. Figure 6 shows that the point clouds from the different sequences align nicely, despite belonging to different weather conditions. This experiment shows that our method can perform the desired operations successfully on a real-world application, including relocalization against unseen weather conditions. Figure 7 demonstrates the difference between our Gauss-Newton loss and the contrastive loss. This shows that the quantitative improvement has a visible effect on the application of relocalization. Figure 5 shows sample images used in the qualitative relocalizations.

C. Additional experiments on EuRoC and CARLA

In the supplementary, we provide more evaluations on datasets with and without brightness variations. This includes relocalization tracking on the CARLA benchmark and visual



(a) Relocalization sunny and overcast.

(b) Relocalization sunny and rainy.

(c) Relocalization sunny and snowy.

(d) Relocalization overcast and rainy.

(e) Relocalization overcast and snowy.

(f) Relocalization rainy and snowy.

Fig. 4: This figure shows the cumulative relocalization accuracy on the Oxford RobotCar dataset for different sequences. D denotes the dimension of the feature descriptor. Our method achieves the highest accuracy across all sequences. It is interesting to observe that despite being trained only on two sequences in overcast and sunny condition, our model still generalizes very well to even *unseen* rainy and snowy conditions. Even the model trained only on the synthetic CARLA benchmark outperforms all baselines, showing exceptional generalization capabilities.

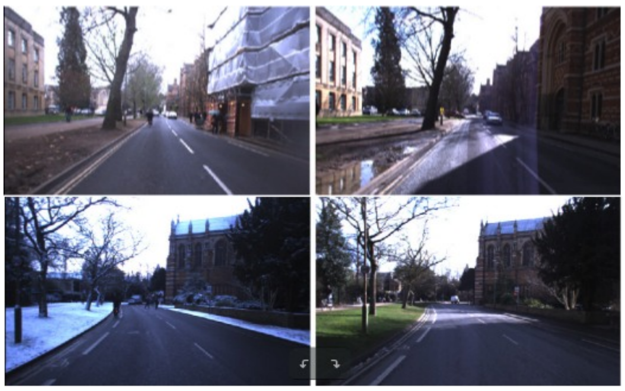


Fig. 5: shows image pairs used in the qualitative relocalizations. Left: rainy (top row) and snowy (bottom row) images relocalized against the sunny reference images (right).

odometry on the EuRoC [11] dataset. We show that also in these situations our deep features significantly outperform DSO and ORB-SLAM because of their robustness to large-baselines. On the EuRoC dataset, we improve the DSO performance by almost a factor of 2 for low-framerates.

VI. Conclusion & Future Work

With the advent of deep learning, we can devise feature space encodings that are designed to be optimally suited for the subsequent visual SLAM algorithms. More specifically, we propose to exploit the probabilistic interpretation of the commonly used Gauss-Newton algorithm to devise a novel loss function for feature space encoding that we call the Gauss-Newton loss. It is designed to promote robustness to strong lighting and weather changes while enforcing a maximal basin of convergence for the respective SLAM algorithm. Quantitative experiments on synthetic and real-world data demonstrates that the Gauss-Newton loss allows us to significantly expand the realm of applicability of direct visual SLAM methods, enabling relocalization and map merging across drastic variations in weather and illumination.

References

[1] G. Klein and D. W. Murray, "Parallel tracking and mapping for small ar workspaces," in *IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2007.

[2] R. Mur-Artal, J. M. Montiel, and J. D. Tardos, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE T-RO*, vol. 31, no. 5, 2015.

[3] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *ECCV*, 2014.

[4] H. Alismail, B. Browning, and S. Lucey, "Photometric Bundle Adjustment for Vision-Based SLAM," in *ACCV*, 2017.

[5] J. Engel, V. Koltun, and D. Cremers, "Direct Sparse Odometry," *IEEE TPAMI*, vol. 40, no. 3, 2018.

[6] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised Image-to-Image Translation Networks," in *NIPS*, 2017.

[7] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks," in *IEEE CVPR*, 2017.

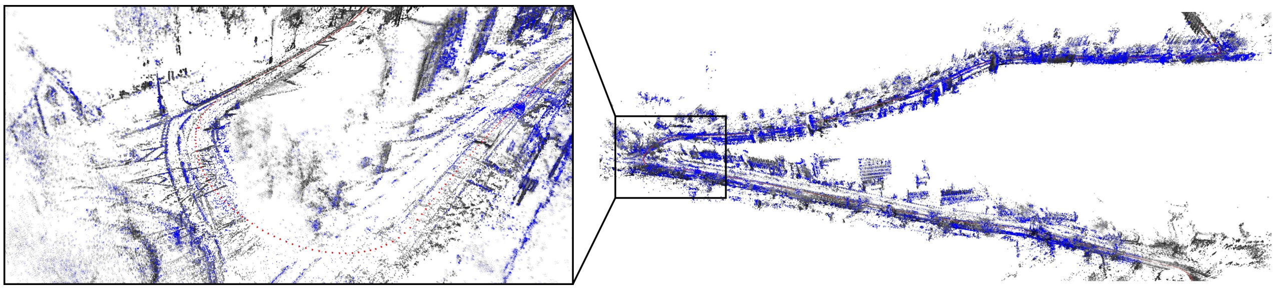


Fig. 6: This figure shows a point cloud result of our GN-Net. We relocalize a rainy sequence (blue) against a reference map created from the sunny sequence (gray).

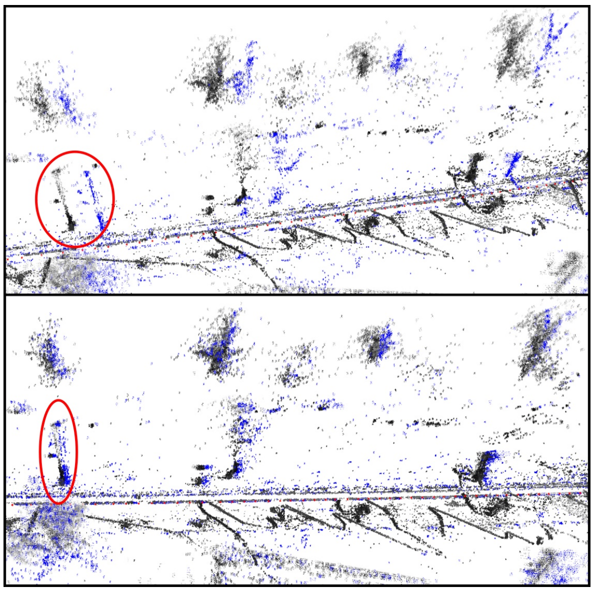


Fig. 7: Top: relocalization using the model trained with only the contrastive loss. Bottom: relocalization using the model trained with our loss formulation. This visually demonstrates the influence of the Gauss-Newton loss.

[8] H. Porav, W. Maddern, and P. Newman, "Adversarial Training for Adverse Conditions: Robust Metric Localisation using Appearance Transfer," in *IEEE ICRA*, 2018.

[9] T. Sattler, W. Maddern, C. Toft, A. Torii, L. Hammarstrand, E. Stenborg, D. Safari, M. Okutomi, M. Pollefeys, J. Sivic, F. Kahl, and T. Pajdla, "Benchmarking 6DOF Outdoor Visual Localization in Changing Conditions," in *IEEE CVPR*, 2018.

[10] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN Architecture for Weakly Supervised Place Recognition," in *IEEE CVPR*, 2016.

[11] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The EuRoC Micro Aerial Vehicle Datasets," *IJRR*, vol. 35, no. 10, 2016.

[12] J. Engel, V. Usenko, and D. Cremers, "A photometrically calibrated benchmark for monocular visual odometry," in *arXiv 2016*, 2016.

[13] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator," in *CoRL*, 2017.

[14] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 Year, 1000km: The Oxford RobotCar Dataset," *IJRR*, vol. 36, no. 1, 2017.

[15] A. Davison, I. Reid, N. Molton, and O. Stasse, "MonoSLAM: Real-Time Single Camera SLAM," *IEEE TPAMI*, no. 6, 2007.

[16] R. Newcombe, S. Lovegrove, and A. Davison, "DTAM: Dense tracking and mapping in real-time," in *ICCV*, 2011.

[17] S. Park, T. Schöps, and M. Pollefeys, "Illumination Change Robustness in Direct Visual SLAM," in *IEEE ICRA*, 2017.

[18] H. Alismail, M. Kaess, B. Browning, and S. Lucey, "Direct Visual Odometry in Low Light Using Binary Descriptors," *RA-L*, vol. 2, 2017.

[19] C. B. Choy, J. Gwak, S. Savarese, and M. Chandraker, "Universal Correspondence Network," in *NIPS*, 2016.

[20] T. Schmidt, R. Newcombe, and D. Fox, "Self-supervised Visual Descriptor Learning for Dense Correspondence," *RA-L*, vol. 2, 2017.

[21] I. Rocco, M. Cimpoi, R. Arandjelović, A. Torii, T. Pajdla, and J. Sivic, "Neighbourhood consensus networks," in *NeurIPS*, 2018.

[22] P. Wohlhart and V. Lepetit, "Learning descriptors for object recognition and 3d pose estimation," *IEEE CVPR*, 2015.

[23] C.-H. Chang, C.-N. Chou, and E. Y. Chang, "CLKN: Cascaded Lucas-Kanade Networks for Image Alignment," in *IEEE CVPR*, 2017.

[24] R. Gomez-Ojeda, Z. Zhang, J. Gonzalez-Jimenez, and D. Scaramuzza, "Learning-based image enhancement for visual odometry in challenging hdr environments," in *IEEE ICRA*, 2018.

[25] J. Czarnowski, S. Leutenegger, and A. J. Davison, "Semantic Texture for Robust Dense Tracking," in *ICCVW*, 2017.

[26] C. Tang and P. Tan, "BA-Net: Dense Bundle Adjustment Network," in *ICLR*, 2019.

[27] L. Han, M. Ji, L. Fang, and M. Nießner, "Regnet: Learning the optimization of direct image-to-image pose registration," in *arXiv 2018*, 2018.

[28] Z. Lv, F. Dellaert, J. Rehg, and A. Geiger, "Taking a deeper look at the inverse compositional algorithm," in *IEEE CVPR*, 2019.

[29] C. Jaramillo, Y. Taguchi, and C. Feng, "Direct multichannel tracking," in *3DV*, 2017.

[30] M. Cummins and P. Newman, "FAB-MAP : Probabilistic Localization and Mapping in the Space of Appearance," *IJRR*, vol. 27, no. 6, 2008.

[31] I. Kapsouras and N. Nikolaidis, "A vector of locally aggregated descriptors framework for action recognition on motion capture data," in *EUSIPCO*, 2018.

[32] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE T-RO*, vol. 33, no. 5, 2017.

[33] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: Exploring photo collections in 3d," in *SIGGRAPH*, 2006.

[34] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *MICCAI*, 2015.

[35] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *IJCAI*, 1981.

[36] A. Kendall, M. Grimes, and R. Cipolla, "PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization," in *ICCV*, 2015.

[37] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv 2018*, 2018.

[38] R. Wang, M. Schwörer, and D. Cremers, "Stereo dso: Large-scale direct sparse visual odometry with stereo cameras," in *ICCV*, 2017.

[39] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler, "D2-Net: A Trainable CNN for Joint Description and Detection of Local Features," in *IEEE CVPR*, 2019.

[40] D. DeTone, T. Malisiewicz, and A. Rabinovich, "SuperPoint: Self-Supervised Interest Point Detection and Description," in *IEEE CVPRW*, 2018.

A.4 4SEASONS: A CROSS-SEASON DATASET FOR MULTI-WEATHER SLAM IN AUTONOMOUS DRIVING

Copyright

Abstract

We present a novel dataset covering seasonal and challenging perceptual conditions for autonomous driving. Among others, it enables research on visual odometry, global place recognition, and map-based re-localization tracking. The data was collected in different scenarios and under a wide variety of weather conditions and illuminations, including day and night. This resulted in more than 350 km of recordings in nine different environments ranging from multi-level parking garage over urban (including tunnels) to countryside and highway. We provide globally consistent reference poses with up-to centimeter accuracy obtained from the fusion of direct stereo visual-inertial odometry with RTK-GNSS. The full dataset is available at https://www.4seasons-dataset.com.

Individual Contributions

Problem definition	*significantly contributed*
Literature survey	*significantly contributed*
Implementation	*significantly contributed*
Experimental evaluation	*significantly contributed*
Preparation of the manuscript	*significantly contributed*

Notice

In accordance with the *Springer Nature Thesis / Dissertation Reuse Permissions*, we include the accepted version of the original publication [7] in the following.

4Seasons: A Cross-Season Dataset for Multi-Weather SLAM in Autonomous Driving

Patrick Wenzel[1,2], Rui Wang[1,2], Nan Yang[1,2], Qing Cheng[2], Qadeer Khan[1,2], Lukas von Stumberg[1,2], Niclas Zeller[2], and Daniel Cremers[1,2]

[1] Technical University of Munich
[2] Artisense
wenzel@cs.tum.edu

Abstract. We present a novel dataset covering seasonal and challenging perceptual conditions for autonomous driving. Among others, it enables research on visual odometry, global place recognition, and map-based re-localization tracking. The data was collected in different scenarios and under a wide variety of weather conditions and illuminations, including day and night. This resulted in more than 350 km of recordings in nine different environments ranging from multi-level parking garage over urban (including tunnels) to countryside and highway. We provide globally consistent reference poses with up-to centimeter accuracy obtained from the fusion of direct stereo visual-inertial odometry with RTK-GNSS. The full dataset is available at `https://www.4seasons-dataset.com`.

Keywords: autonomous driving, long-term localization, SLAM, visual learning, visual odometry

1 Introduction

During the last decade, research on visual odometry (VO) and simultaneous localization and mapping (SLAM) has made tremendous strides [30,12,29,11] particularly in the context of autonomous driving (AD) [9,44,46,28]. One reason for this progress has been the publication of large-scale datasets [7,14,6] tailored for benchmarking these methods. Naturally, the next logical step towards progressing research in the direction of visual SLAM has been to make it robust under dynamically changing and challenging conditions. This includes VO, *e.g.* at night or rain, as well as long-term place recognition and re-localization against a pre-built map. In this regard, the advent of deep learning has exhibited itself to be a promising potential in complementing the performance of visual SLAM [8,22,39,20]. Therefore, it has become all the more important to have datasets that are commensurate with handling the challenges of any real-world environment while also being capable of discerning the performance of state-of-the-art approaches.

To accommodate this demand, we present in this paper a versatile cross-season and multi-weather dataset on a large-scale focusing on long-term localization for autonomous driving. By traversing the same stretch under different

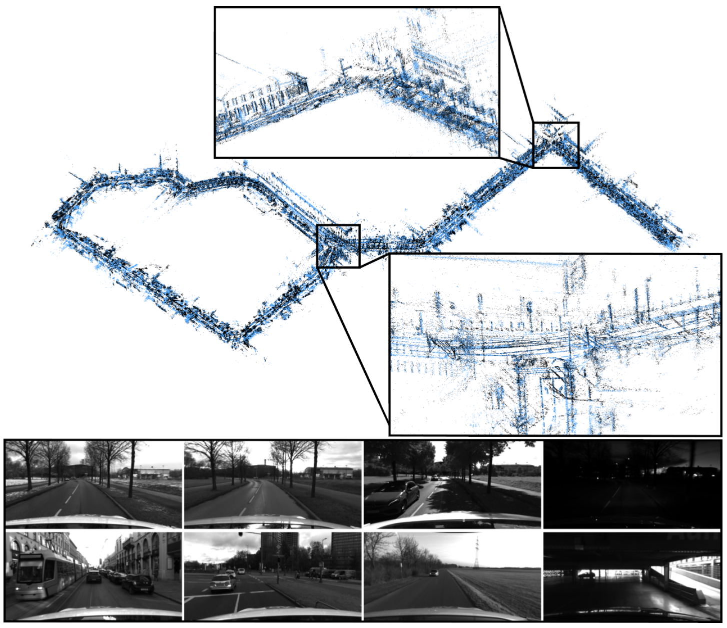


Fig. 1: **Dataset overview.** Top: overlaid maps recorded at different times and environmental conditions. The points from the reference map (black) align well with the points from the query map (blue), indicating that the reference poses are indeed accurate. Bottom: sample images demonstrating the diversity of our dataset. The first row shows a collection from the same scene across different weather and lighting conditions: snowy, overcast, sunny, and night. The second row depicts the variety of scenarios within the dataset: inner city, suburban, countryside, and a parking garage.

conditions and over a long-term time horizon, we capture variety in illumination and weather as well as in the appearance of the scenes. Figure 1 visualizes two overlaid 3D maps recorded at different times as well as sample images of the dataset.

In detail this work adds the following contributions to the state-of-the-art:

- A cross-season/multi-weather dataset for long-term visual SLAM in automotive applications containing more than 350 km of recordings.
- Sequences covering nine different kinds of environments ranging from multi-level parking garage over urban (including tunnels) to countryside and highway.
- Global six degrees of freedom (6DoF) reference poses with up-to centimeter accuracy obtained from the fusion of direct stereo visual-inertial odometry (VIO) with RTK-GNSS.
- Accurate cross-seasonal pixel-wise correspondences to train dense feature representations.

2 Related Work

There exists a variety of benchmarks and datasets focusing on VO and SLAM for AD. Here, we divide these datasets into the ones which focus only on the task of VO as well as those covering different weather conditions and therefore aiming towards long-term SLAM.

2.1 Visual Odometry

The most popular benchmark for AD certainly is KITTI [14]. This multi-sensor dataset covers a wide range of tasks including not only VO, but also 3D object detection, and tracking, scene flow estimation as well as semantic scene understanding. The dataset contains diverse scenarios ranging from urban over countryside to highway. Nevertheless, all scenarios are only recorded once and under similar weather conditions. Ground truth is obtained based on a high-end inertial navigation system (INS).

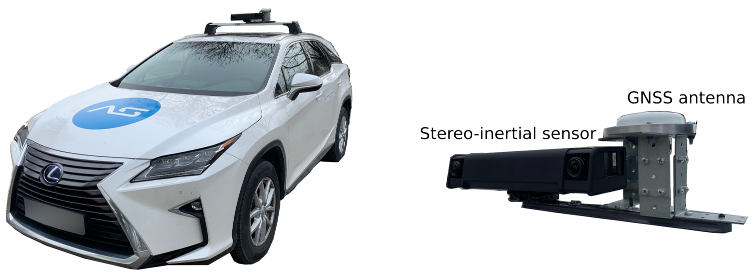
Another dataset containing LiDAR, inertial measurement unit (IMU), and image data at a large-scale is the Málaga Urban dataset [4]. However, in contrast to KITTI, no accurate 6DoF ground truth is provided and therefore it does not allow for a quantitative evaluation based on this dataset.

Other popular datasets for the evaluation of VO and VIO algorithms not related to AD include [40] (handheld RGB-D), [5] (UAV stereo-inertial), [10] (handheld mono), and [36] (handheld stereo-inertial).

2.2 Long-Term SLAM

More related to our work are datasets containing multiple traversals of the same environment over a long period of time. With respect to SLAM for AD the Oxford RobotCar Dataset [27] represents a kind of pioneer work. This dataset consists of large-scale sequences recorded multiple times for the same environment over a period of one year. Hence, it covers large variations in the appearance and structure of the scene. However, the diversity of the scenarios is only limited to an urban environment. Also, the ground truth provided for the dataset is not accurate up-to centimeter-level and therefore, requires additional manual effort to establish accurate cross-sequence correspondences.

The work by [34] represents a kind of extension to [27]. This benchmark is based on subsequences from [27] as well as other datasets. The ground truth of the RobotCar Seasons [34] dataset is obtained based on structure from motion (SfM) and LiDAR point cloud alignment. However, due to inaccurate GNSS measurements [27], a globally consistent ground truth up-to centimeter-level can not be guaranteed. Furthermore, this dataset only provides one reference traversal in the overcast condition. In contrast, we provide globally consistent reference models for all traversals covering a wide variety of conditions. Hence, every traversal can be used as a reference model that allows further research, *e.g.* on analyzing suitable reference-query pairs for long-term localization and mapping.



(a) Test vehicle. (b) Sensor system.

Fig. 2: **Recording setup.** Test vehicle and sensor system used for dataset recording. The sensor system consists of a custom stereo-inertial sensor with a stereo baseline of 30 cm and a high-end RTK-GNSS receiver from Septentrio.

2.3 Other Datasets

Examples of further multi-purpose AD datasets which also can be used for VO are [7,45,19,6].

As stated in Section 1, our proposed dataset differentiates from previous related work in terms of being both large-scale (similar to [14]) as well as having high variations in appearance and conditions (similar to [27]). Furthermore, we are providing accurate reference poses based on the fusion of direct stereo VIO and RTK-GNSS.

3 System Overview

This section presents the sensor setup which is used for data recording (Section 3.1). Furthermore, we describe the calibration of the entire sensor suite (Section 3.2) as well as our approach to obtain up-to centimeter-accurate global 6DoF reference poses (Section 3.3).

3.1 Sensor Setup

The hardware setup consists of a custom stereo-inertial sensor for 6DoF pose estimation as well as a high-end RTK-GNSS receiver for global positioning and global pose refinement. Figure 2 shows our test vehicle equipped with the sensor system used for data recording.

Stereo-Inertial Sensor. The core of the sensor system is our custom stereo-inertial sensor. This sensor consists of a pair of monochrome industrial-grade global shutter cameras (Basler acA2040-35gm) and lenses with a fixed focal length of $f = 3.5$ mm (Stemmer Imaging CVO GMTHR23514MCN). The cameras are mounted on a highly-rigid aluminum rail with a stereo baseline of 30 cm.

On the same rail, an IMU (Analog Devices ADIS16465) is mounted. All sensors, cameras, and IMU are triggered over an external clock generated by an field-programmable gate array (FPGA). Here, the trigger accounts for exposure compensations, meaning that the time between the centers of the exposure interval for two consecutive images is always kept constant (1/[frame rate]) independent of the exposure time itself.

Furthermore, based on the FPGA, the IMU is properly synchronized with the cameras. In the dataset, we record stereo sequences with a frame rate of 30 fps. We perform pixel binning with a factor of two and crop the image to a resolution of 800×400. This results in a field of view of approximately $77°$ horizontally and $43°$ vertically. The IMU is recorded at a frequency of 2000 Hz. During recording, we run our custom auto-exposure algorithm, which guarantees equal exposure times for all stereo image pairs as well as a smooth exposure transition in highly dynamic lighting conditions, as it is required for visual SLAM. We provide those exposure times for each frame.

GNSS Receiver. For global positioning and to compensate drift in the VIO system we utilize an RTK-GNSS receiver (mosaic-X5) from Septentrio in combination with an Antcom Active G8 GNSS antenna. The GNSS receiver provides a horizontal position accuracy of up-to 6 mm by utilizing RTK corrections. While the high-end GNSS receiver is used for accurate positioning, we use a second receiver connected to the time-synchronization FPGA to achieve synchronization between the GNSS receiver and the stereo-inertial sensor.

3.2 Calibration

Aperture and Focus Adjustment. The lenses used in the stereo-system have both adjustable aperture and focus. Therefore, before performing the geometric calibration of all sensors, we manually adjust both cameras for a matching average brightness and a minimum focus blur [18], across a structured planar target in 10 m distance.

Stereo Camera and IMU. For the intrinsic and extrinsic calibration of the stereo cameras as well as the extrinsic calibration and time-synchronization of the IMU, we use a slightly customized version of *Kalibr*[1] [32]. The stereo cameras are modeled using the Kannala-Brandt model [23], which is a generic camera model consisting of in total eight parameters. To guarantee an accurate calibration over a long-term period, we perform a feature-based epipolar-line consistency check for each sequence recorded in the dataset and re-calibrate before a recording session if necessary.

GNSS Antenna. Since the GNSS antenna does not have any orientation but has an isotropic reception pattern, only the 3D translation vector between one

[1] `https://github.com/ethz-asl/kalibr`

of the cameras and the antenna within the camera frame has to be known. This vector was measured manually for our sensor setup.

3.3 Ground Truth Generation

Reference poses (*i.e.* ground truth) for VO and SLAM should provide high accuracy in both local relative 6DoF transformations and global positioning. To fulfill the first requirement, we extend the state-of-the-art stereo direct sparse VO [44] by integrating IMU measurements [43], achieving a stereo-inertial SLAM system offering average tracking drift around 0.6 % of the traveled distance. To fulfill the second requirement, the poses estimated by our stereo-inertial system are integrated into a global pose graph, each with an additional constraint from the corresponding RTK-GNSS measurement. Our adopted RTK-GNSS system can provide global positioning with up-to centimeter accuracy. The pose graph is optimized globally using the Gauss-Newton method, ending up with 6DoF camera poses with superior accuracy both locally and globally. For the optimization, we make use of the g2o library [25].

One crucial aspect for the dataset is that the reference poses which we provide are actually accurate enough, even though some of the recorded sequences partially contain challenging conditions in GNSS-denied environments. Despite the fact that the stereo-inertial sensor system has an average drift around 0.6 %, this cannot be guaranteed for all cases. Hence, for the reference poses in our dataset, we report whether a pose can be considered to be reliable by measuring the distance to the corresponding RTK-GNSS measurement. Only RTK-GNSS measurements with a reported standard deviation of less than 0.01 m are considered as accurate. For all poses, without corresponding RTK-GNSS measurement we do not guarantee a certain accuracy. Nevertheless, due to the highly accurate stereo-inertial odometry system, these poses still can be considered to be accurate in most cases even in GNSS-denied environments, *e.g.* tunnels or areas with tall buildings.

4 Scenarios

This section describes the different scenarios we have collected for the dataset. The scenarios involve different sequences – ranging from urban driving to parking garage and rural areas. We provide complex trajectories, which include partially overlapping routes, and multiple loops within a sequence. For each scenario, we have collected multiple traversals covering a large range of variation in environmental appearance and structure due to weather, illumination, dynamic objects, and seasonal effects. In total, our dataset consists of nine different scenarios, *i.e.* industrial area, highway, local neighborhood, ring road, countryside, suburban, inner city, monumental site, and multi-level parking garage.

We provide reference poses and 3D models generated by our ground truth generation pipeline (*c.f.* Figure 3) along with the corresponding raw image

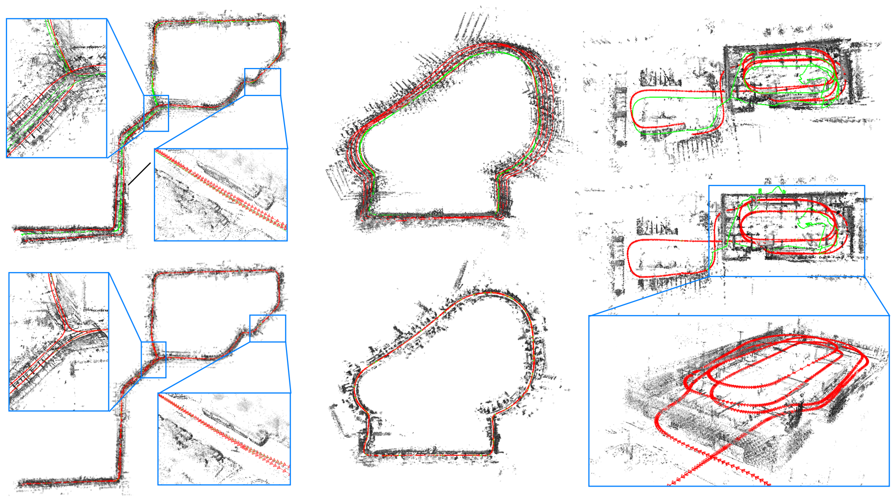


Fig. 3: **3D models of different scenarios contained in the dataset**. The figure shows a loop around an industrial area (left), multiple loops around an area with high buildings (middle), and a stretch recorded in a multi-level parking garage (right). The green lines encode the GNSS trajectories, and the red lines encode the VIO trajectories. Top: shows the trajectories before the fusion using pose graph optimization. Bottom: shows the result after the pose graph optimization. Note that after the pose graph optimization the reference trajectory is well aligned.

frames and raw IMU measurements. Figure 4 shows another example of the optimized trajectory, which depicts the accuracy of the provided reference poses.

The dataset will challenge current approaches on long-term localization and mapping since it contains data from various seasons and weather conditions as well as from different times of the day as shown in the bottom part of Figure 1.

4.1 Ground Truth Validation

The top part of Figure 1 shows two overlaid point clouds from different runs across the same scene. Note that despite the weather and seasonal differences the point clouds align very well. This shows that our reference poses are indeed very accurate. Furthermore, a qualitative assessment of the point-to-point correspondences is shown in Figure 5. The figure shows a subset of very accurate pixel-wise correspondences across different seasons (*autumn*/*winter*) in the top and different illumination conditions (*sunny*/*night*) in the bottom. These point-to-point correspondences are a result of our up-to centimeter-accurate global reference poses and are obtained in a completely self-supervised manner. This makes them suitable as training pairs for learning-based algorithms. Recently, there has been an increasing demand for pixel-wise cross-season correspondences

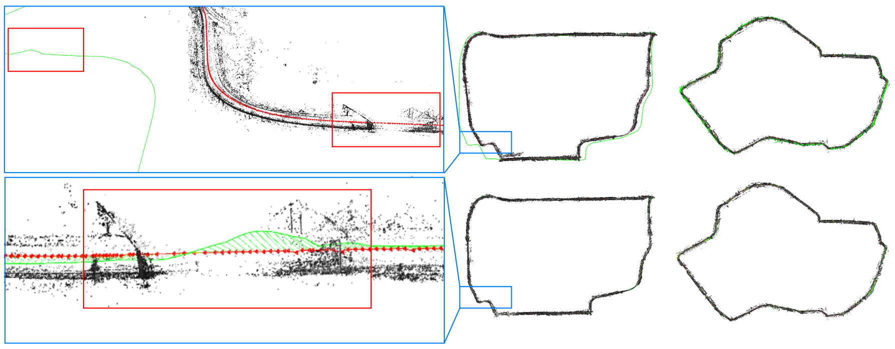


Fig. 4: **Reference poses validation.** This figure shows two additional 3D models of the scenarios collected. Note that these two sequences are quite large (more than 10 km and 6 km, respectively). Top: before the fusion using pose graph optimization. Bottom: results after optimization. The green lines encode the GNSS trajectories, the red lines show the VIO trajectories (before fusion) and the fused trajectories (after fusion). The left part of the figure shows a zoomed-in view of a tunnel, where the GNSS signal becomes very noisy as highlighted in the red boxes. Besides, due to the large size of the sequence, the accumulated tracking error leads to a significant deviation of the VIO trajectory from the GNSS recordings. Our pose graph optimization, by depending globally on GNSS positions and locally on VIO relative poses, successfully eliminates global VIO drifts and local GNSS positioning flaws.

which are needed to learn dense feature descriptors [38,8,33]. However, there is still a lack of datasets to satisfy this demand. The KITTI [14] dataset does not provide cross-seasons data. The Oxford RobotCar Dataset [27] provides cross-seasons data, however, since the ground truth is not accurate enough, the paper does not recommend benchmarking localization and mapping approaches.

Recently, RobotCar Seasons [34] was proposed to overcome the inaccuracy of the provided ground truth. However, similar to the authors of [38], we found that it is still challenging to obtain accurate cross-seasonal pixel-wise matches due to pose inconsistencies. Furthermore, this dataset only provides images captured from three synchronized cameras mounted on a car, pointing to the rear-left, rear, and rear-right, respectively. Moreover, the size of the dataset is quite small and a significant portion of it suffers from strong motion blur and low image quality.

To the best of our knowledge, our dataset is the first that exhibits accurate cross-season reference poses for the AD domain.

5 Tasks

This section describes the different tasks of the dataset. The provided globally consistent 6DoF reference poses for diverse conditions will be valuable to develop

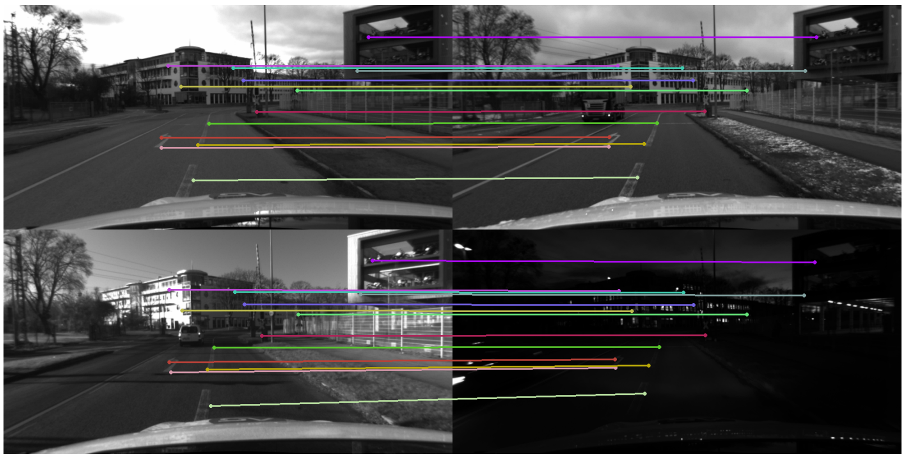


Fig. 5: **Accurate pixel-wise correspondences, making cross-seasonal training possible.** Qualitative assessment of the accuracy of our data collection and geometric reconstruction method for a sample of four different conditions (from top left in clockwise order: *overcast, snowy, night, sunny*) across the same scene. Each same colored point in the four images corresponds to the same geometric point in the world. The cameras corresponding to these images have different poses in the global frame of reference. Please note that the points are not matched but rather a result of our accurate reference poses and geometric reconstruction. This way we are capable of obtaining sub-pixel level accuracy. On average we get more than 1000 of those correspondences per image pair.

and improve the state-of-the-art for different SLAM related tasks. Here the major tasks are robust VO, global place recognition, and map-based re-localization tracking.

In the following, we will present the different subtasks for our dataset.

5.1 Visual Odometry in Different Weather Conditions

VO aims to accurately estimate the 6DoF pose for every frame relative to a starting position. To benchmark the task of VO there already exist various datasets [15,40,10]. All of these existing datasets consist of sequences recorded at rather homogeneous conditions (indoors, or sunny/overcast outdoor conditions). However, especially methods developed for AD use cases must perform robustly under almost any condition. We believe that the proposed dataset will contribute to improving the performance of VO under diverse weather and lighting conditions in an automotive environment. Therefore, instead of replacing existing benchmarks and datasets, we aim to provide an extension that is more focusing on challenging conditions in AD. As we provide frame-wise accurate poses for large portions of the sequences, metrics well known from other benchmarks like

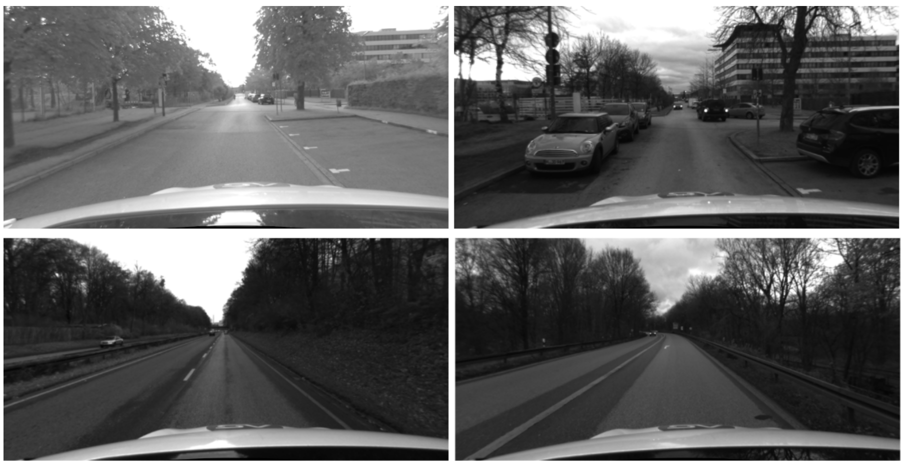


Fig. 6: **Challenging scenes for global place recognition.** Top: two pictures share the same location with different appearances. Bottom: two pictures have similar appearance but are taken at different locations.

absolute trajectory error (ATE) or relative pose error (RPE) [15,40] are also applicable to our data.

5.2 Global Place Recognition

Global place recognition refers to the task of retrieving the most similar database image given a query image [26]. In order to improve the searching efficiency and the robustness against different weather conditions, tremendous progress on global descriptors [21,3,1,13] has been seen. For the re-localization pipeline, visual place recognition serves as the initialization step to the downstream local pose refinement by providing the most similar database images as well as the corresponding global poses. Due to the advent of deep neural networks [37,24,17,41], methods aggregating deep image features are proposed and have shown advantages over classical methods [2,16,31,42].

The proposed dataset is challenging for global place recognition since it contains not only cross-season images that have different appearances but share a similar geographical location but also the intra-season images which share similar appearances but with different locations. Figure 6 depicts example pairs of these scenarios. We suggest to follow the standard metric widely used for global place recognition [2,3,35,16].

5.3 Map-Based Re-Localization Tracking

Map-based re-localization tracking [39] refers to the task of locally refining the 6DoF pose between reference images from a pre-built reference map and images from a query sequence. In contrast to wide-baseline stereo matching, for re-localization tracking, it is also possible to utilize the sequential information of the sequence. This allows us to estimate depth values by running a standard VO method. Those depth estimates can then be used to improve the tracking of the individual re-localization candidates.

In this task we assume to know the mapping between reference and query samples. This allows us to evaluate the performance of local feature descriptor methods in isolation. In practice, this mapping can be found using image retrieval techniques like NetVLAD [2] as described in Section 5.2 or by aligning the point clouds from the reference and query sequences [34], respectively.

Accurately re-localizing in a pre-built map is a challenging problem, especially if the visual appearance of the query sequence significantly differs from the base map. This makes it extremely difficult especially for vision-based systems since the localization accuracy is often limited by the discriminative power of feature descriptors. Our proposed dataset allows us to evaluate re-localization tracking across multiple types of weather conditions and diverse scenes, ranging from urban to countryside driving. Furthermore, our up to centimeter-accurate ground truth allows us to create diverse and challenging re-localization tracking candidates with an increased level of difficulty. By being able to precisely changing the re-localization distances and the camera orientation between the reference and query samples, we can generate more challenging scenarios. This allows us to determine the limitations and robustness of current state-of-the-art methods.

6 Conclusion

We have presented a cross-season dataset for the purpose of multi-weather SLAM, global visual localization, and local map-based re-localization tracking for AD applications. Compared to other datasets, like KITTI [14] or Oxford RobotCar [27], the presented dataset provides diversity in both multiplicities of scenarios and environmental conditions. Furthermore, based on the fusion of direct stereo VIO and RTK-GNSS we are able to provide up-to centimeter-accurate reference poses as well as highly accurate cross-sequence correspondences. One drawback of the dataset is that the accuracy of the reference poses can only be guaranteed in environments with good GNSS receptions. However, due to the low drift of the stereo VIO system, the obtained reference poses are also very accurate in GNSS-denied environments, *e.g.* tunnels, garages, or urban canyons.

We believe that this dataset will help the research community to further understand the limitations and challenges of long-term visual SLAM in changing conditions and environments and will contribute to advance the state-of-the-art. To the best of our knowledge, ours is the first large-scale dataset for AD providing cross-seasonal accurate pixel-wise correspondences for diverse scenarios. This will help to vastly increase robustness against environmental changes for deep learning methods. The dataset is made publicly available to facilitate further research.

References

1. Angeli, A., Filliat, D., Doncieux, S., Meyer, J.A.: Fast and incremental method for loop-closure detection using bags of visual words. IEEE Transactions on Robotics (T-RO) **24**(5), 1027–1037 (2008)
2. Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., Sivic, J.: NetVLAD: CNN architecture for weakly supervised place recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5297–5307 (2016)
3. Arandjelovic, R., Zisserman, A.: All about VLAD. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1578–1585 (2013)
4. Blanco-Claraco, J.L., Ángel Moreno-Dueñas, F., González-Jiménez, J.: The Málaga urban dataset: High-rate stereo and LiDAR in a realistic urban scenario. International Journal of Robotics Research (IJRR) **33**(2), 207–214 (2014)
5. Burri, M., Nikolic, J., Gohl, P., Schneider, T., Rehder, J., Omari, S., Achtelik, M.W., Siegwart, R.: The EuRoC micro aerial vehicle datasets. International Journal of Robotics Research (IJRR) **35**(10), 1157–1163 (2016)
6. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuScenes: A multimodal dataset for autonomous driving. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 11621–11631 (2020)
7. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3213–3223 (2016)
8. Dusmanu, M., Rocco, I., Pajdla, T., Pollefeys, M., Sivic, J., Torii, A., Sattler, T.: D2-Net: A trainable CNN for joint detection and description of local features. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 8092–8101 (2019)
9. Engel, J., Stückler, J., Cremers, D.: Large-scale direct SLAM with stereo cameras. In: Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS). pp. 1935–1942 (2015)
10. Engel, J., Usenko, V., Cremers, D.: A photometrically calibrated benchmark for monocular visual odometry. In: arXiv preprint arXiv:1607.02555 (2016)
11. Engel, J., Koltun, V., Cremers, D.: Direct sparse odometry. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) **40**(3), 611–625 (2017)
12. Engel, J., Schöps, T., Cremers, D.: LSD-SLAM: Large-scale direct monocular SLAM. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 834–849 (2014)
13. Gálvez-López, D., Tardos, J.D.: Bags of binary words for fast place recognition in image sequences. IEEE Transactions on Robotics (T-RO) **28**(5), 1188–1197 (2012)
14. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The KITTI dataset. International Journal of Robotics Research (IJRR) **32**(11), 1231–1237 (2013)
15. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the KITTI vision benchmark suite. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3354–3361 (2012)
16. Gordo, A., Almazán, J., Revaud, J., Larlus, D.: Deep image retrieval: Learning global representations for image search. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 241–257 (2016)

17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778 (2016)
18. Hu, H., de Haan, G.: Low cost robust blur estimator. In: Proceedings of the IEEE International Conference on Image Processing (ICIP). pp. 617–620 (2006)
19. Huang, X., Cheng, X., Geng, Q., Cao, B., Zhou, D., Wang, P., Lin, Y., Yang, R.: The ApolloScape dataset for autonomous driving. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 954–960 (2018)
20. Jaramillo, C.: Direct multichannel tracking. In: Proceedings of the International Conference on 3D Vision (3DV). pp. 347–355 (2017)
21. Jégou, H., Douze, M., Schmid, C., Pérez, P.: Aggregating local descriptors into a compact image representation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3304–3311 (2010)
22. Jung, E., Yang, N., Cremers, D.: Multi-frame GAN: Image enhancement for stereo visual odometry in low light. In: Conference on Robot Learning (CoRL). pp. 651–660 (2019)
23. Kannala, J., Brandt, S.S.: A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) **28**(8), 1335–1340 (2006)
24. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Neural Information Processing Systems (NeurIPS). pp. 1097–1105 (2012)
25. Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., Burgard, W.: g2o: A general framework for graph optimization. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). pp. 3607–3613 (2011)
26. Lowry, S., Sünderhauf, N., Newman, P., Leonard, J.J., Cox, D., Corke, P., Milford, M.J.: Visual place recognition: A survey. IEEE Transactions on Robotics (T-RO) **32**(1), 1–19 (2015)
27. Maddern, W., Pascoe, G., Linegar, C., Newman, P.: 1 year, 1000 km: The oxford robotcar dataset. International Journal of Robotics Research (IJRR) **36**(1), 3–15 (2017)
28. Mur-Artal, R., Tardós, J.D.: ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras. IEEE Transactions on Robotics (T-RO) **33**(5), 1255–1262 (2017)
29. Mur-Artal, R., Montiel, J.M.M., Tardos, J.D.: ORB-SLAM: a versatile and accurate monocular SLAM system. IEEE Transactions on Robotics (T-RO) **31**(5), 1147–1163 (2015)
30. Newcombe, R.A., Lovegrove, S.J., Davison, A.J.: DTAM: dense tracking and mapping in real-time. In: Proceedings of the International Conference on Computer Vision (ICCV). pp. 2320–2327 (2011)
31. Radenović, F., Tolias, G., Chum, O.: Fine-tuning CNN image retrieval with no human annotation. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) **41**(7), 1655–1668 (2018)
32. Rehder, J., Nikolic, J., Schneider, T., Hinzmann, T., Siegwart, R.: Extending kalibr: Calibrating the extrinsics of multiple IMUs and of individual axes. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). pp. 4304–4311 (2016)
33. Revaud, J., Weinzaepfel, P., de Souza, C.R., Humenberger, M.: R2D2: repeatable and reliable detector and descriptor. In: Neural Information Processing Systems (NeurIPS). pp. 12405–12415 (2019)

34. Sattler, T., Maddern, W., Toft, C., Torii, A., Hammarstrand, L., Stenborg, E., Safari, D., Okutomi, M., Pollefeys, M., Sivic, J., Kahl, F., Pajdla, T.: Benchmarking 6DOF outdoor visual localization in changing conditions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 8601–8610 (2018)
35. Sattler, T., Weyand, T., Leibe, B., Kobbelt, L.: Image retrieval for image-based localization revisited. In: Proceedings of the British Machine Vision Conference (BMVC) (2012)
36. Schubert, D., Goll, T., Demmel, N., Usenko, V., Stückler, J., Cremers, D.: The TUM VI benchmark for evaluating visual-inertial odometry. In: Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS). pp. 1680–1687 (2018)
37. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: Proceedings of the International Conference on Learning Representations (ICLR) (2015)
38. Spencer, J., Bowden, R., Hadfield, S.: Same features, different day: Weakly supervised feature learning for seasonal invariance. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6459–6468 (2020)
39. von Stumberg, L., Wenzel, P., Khan, Q., Cremers, D.: GN-Net: The gauss-newton loss for multi-weather relocalization. IEEE Robotics and Automation Letters (RA-L) **5**(2), 890–897 (2020)
40. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of RGB-D SLAM systems. In: Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS). pp. 573–580 (2012)
41. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1–9 (2015)
42. Tolias, G., Sicre, R., Jégou, H.: Particular object retrieval with integral max-pooling of CNN activations. arXiv preprint arXiv:1511.05879 (2015)
43. Von Stumberg, L., Usenko, V., Cremers, D.: Direct sparse visual-inertial odometry using dynamic marginalization. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). pp. 2510–2517 (2018)
44. Wang, R., Schwörer, M., Cremers, D.: Stereo DSO: Large-scale direct sparse visual odometry with stereo cameras. In: Proceedings of the International Conference on Computer Vision (ICCV). pp. 3903–3911 (2017)
45. Wang, S., Bai, M., Mattyus, G., Chu, H., Luo, W., Yang, B., Liang, J., Cheverie, J., Fidler, S., Urtasun, R.: TorontoCity: Seeing the world with a million eyes. In: Proceedings of the International Conference on Computer Vision (ICCV) (2017)
46. Yang, N., Wang, R., Stückler, J., Cremers, D.: Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 817–833 (2018)

A.5 LM-RELOC: LEVENBERG-MARQUARDT BASED DIRECT VISUAL RELOCALIZATION

Copyright

Abstract

We present LM-Reloc – a novel approach for visual relocalization based on direct image alignment. In contrast to prior works that tackle the problem with a feature-based formulation, the proposed method does not rely on feature matching and RANSAC. Hence, the method can utilize not only corners but any region of the image with gradients. In particular, we propose a loss formulation inspired by the classical Levenberg-Marquardt algorithm to train LM-Net. The learned features significantly improve the robustness of direct image alignment, especially for relocalization across different conditions. To further improve the robustness of LM-Net against large image baselines, we propose a pose estimation network, CorrPoseNet, which regresses the relative pose to bootstrap the direct image alignment. Evaluations on the CARLA and Oxford RobotCar relocalization tracking benchmark show that our approach delivers more accurate results than previous state-of-the-art methods while being comparable in terms of robustness.

Individual Contributions

Problem definition	*significantly contributed*
Literature survey	*significantly contributed*
Algorithm development	*contributed*
Method implementation	*contributed*
Experimental evaluation	*contributed*
Preparation of the manuscript	*significantly contributed*

Notice

In accordance with the *IEEE Thesis / Dissertation Reuse Permissions*, we include the accepted version of the original publication [4] in the following.

LM-Reloc: Levenberg-Marquardt Based Direct Visual Relocalization

Lukas von Stumberg[1,2*] Patrick Wenzel[1,2*] Nan Yang[1,2] Daniel Cremers[1,2]
[1] Technical University of Munich [2] Artisense

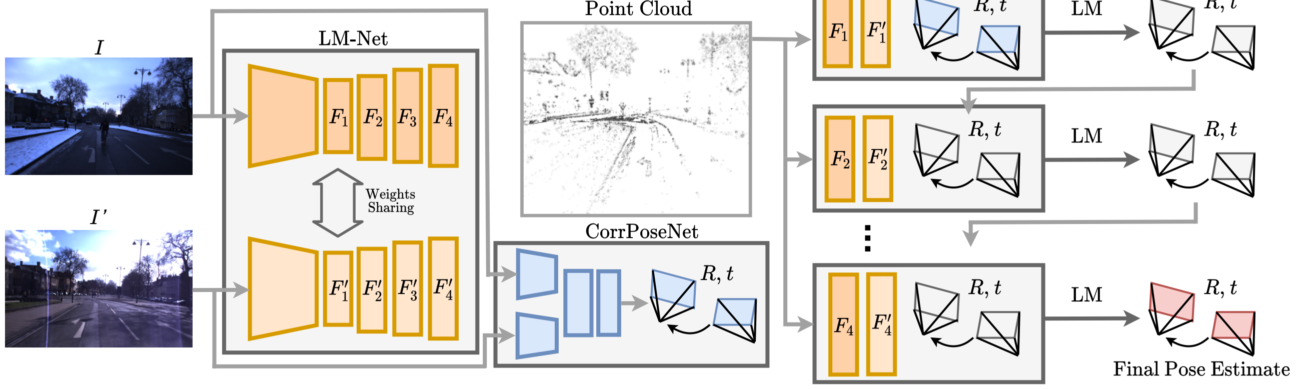


Figure 1: We propose LM-Reloc – a novel approach for visual relocalization based on direct image alignment. It consists of two deep neural networks: LM-Net, an encoder-decoder network for learning dense visual descriptors and a CorrPoseNet to bootstrap the direct image alignment. The final 6DoF relative pose estimate between image I and I' is obtained in a coarse-to-fine pyramid scheme leveraging the learned feature maps. The initialization for the direct image alignment is obtained by the CorrPoseNet.

Abstract

We present LM-Reloc – a novel approach for visual relocalization based on direct image alignment. In contrast to prior works that tackle the problem with a feature-based formulation, the proposed method does not rely on feature matching and RANSAC. Hence, the method can utilize not only corners but any region of the image with gradients. In particular, we propose a loss formulation inspired by the classical Levenberg-Marquardt algorithm to train LM-Net. The learned features significantly improve the robustness of direct image alignment, especially for relocalization across different conditions. To further improve the robustness of LM-Net against large image baselines, we propose a pose estimation network, CorrPoseNet, which regresses the relative pose to bootstrap the direct image alignment. Evaluations on the CARLA and Oxford RobotCar relocalization tracking benchmark show that our approach delivers more accurate results than previous state-of-the-art methods while being comparable in terms of robustness.

1. Introduction

Map-based relocalization, that is, to localize a camera within a pre-built reference map, is becoming more and more important for robotics [6], autonomous driving [24, 4] and AR/VR [29]. Sequential-based approaches, which leverage the temporal structure of the scene provide more stable pose estimations and also deliver the positions in global coordinates compared to single image-based localization methods. The map is usually generated by either using LiDAR or visual Simultaneous Localization and Mapping (vSLAM) solutions. In this paper, we consider vSLAM maps due to the lower-cost visual sensors and the richer semantic information from the images. Feature-based methods [14, 7, 22, 23] and direct methods [13, 12, 11, 1] are two main lines of research for vSLAM.

Once a map is available, the problem of relocalizing within this map at any later point in time requires to deal with long-term changes in the environment. This makes a centimeter-accurate global localization challenging, especially in the presence of drastic lighting and appearance changes in the scene. For this task, feature-based methods are the most commonly used approaches to estimate the ego-pose and its orientation. This is mainly due to the advantage that features are more robust against changes in lighting/illumination in the scene.

However, feature-based methods can only utilize keypoints that have to be matched across the images before the pose estimation begins. Thus they ignore large parts of the

*Equal contribution.

available information. Direct methods, in contrast, can take advantage of all image regions with sufficient gradients and as a result, are known to be more accurate on visual odometry benchmarks [41, 11, 39].

In this paper, we propose LM-Reloc, which applies direct techniques to the task of relocalization. LM-Reloc consists of LM-Net, CorrPoseNet, and a non-linear optimizer, which work seamlessly together to deliver reliable pose estimation without RANSAC and feature matching. In particular, we derive a loss formulation, which is specifically designed to work well with the Levenberg-Marquardt (LM) algorithm [16, 20]. We use a deep neural network, LM-Net, to train descriptors that are being fed to the direct image alignment algorithm. Using these features results in better robustness against bad initializations, large baselines, and against illumination changes.

While the robustness improvements gained with our loss formulation are sufficient in many cases, for very large baselines or strong rotations, some initialization can still be necessary. To this end, we propose a pose estimation network. Based on two images it directly regresses the 6DoF pose, which we utilize as initialization for LM-Net. The CorrPoseNet contains a correlation layer as proposed in [27], which ensures that the network can handle large displacements. The proposed CorrPoseNet displays a lot of synergies with LM-Net. Despite being quite robust, the predictions of the CorrPoseNet are not very accurate. Thus it is best used in conjunction with our LM-Net, resulting in very robust and accurate pose estimates.

We evaluate our approach on the relocalization tracking benchmark from [36], which contains scenes simulated using CARLA [9], as well as sequences from the Oxford RobotCar dataset [19]. Our LM-Net shows superior accuracy especially in terms of rotation while being competitive in terms of robustness.

We summarize our main contributions:

- LM-Reloc, a novel pipeline for visual relocalization based on direct image alignment, which consists of LM-Net, CorrPoseNet, and a non-linear optimizer.
- A novel loss formulation together with a point sampling strategy that is used to train LM-Net such that the resulting feature descriptors are optimally suited to work with the LM algorithm.
- Extensive evaluations on the CARLA and Oxford RobotCar relocalization tracking benchmark which show that the proposed approach achieves state-of-the-art relocalization accuracy without relying on feature matching or RANSAC.

2. Related Work

In this section, we review the main topics that are closely related to our work, including direct methods for visual localization and feature-based visual localization methods.

Direct methods for visual localization. In recent years, direct methods [13, 12, 11] for SLAM and visual odometry have seen a great progress. Unlike feature-based methods [14, 7, 22, 23] which firstly extracts keypoints as well as the corresponding descriptors, and then minimize the geometric errors, direct methods minimize the energy function based on the photometric constancy assumption without performing feature matching or RANSAC. By utilizing more points from the images, direct methods show higher accuracy than feature-based methods [39]. However, classical direct methods show lower robustness than feature-based methods when the photometric constancy assumption is violated due to, e.g., the lighting and weather changes which are typical for long-term localization [33]. In [2] and [25], the authors propose to use the handcrafted features to improve the robustness of direct methods against low light or global appearance changes. Some recent works [5, 18, 36] address the issue by using learned features from deep neural networks [15]. In [5] they train deep features using a Hinge-Loss based on the Lucas-Kanade method, however, in contrast to us, they estimate the optical flow instead of applying the features to the task of relocalization. The most related work to ours is GN-Net [36] which proposes a Gauss-Newton loss to learn deep features. By performing direct image alignment on the learned features, GN-Net can deliver reliable pose estimation between the images taken from different weather or season conditions. The proposed LM-Net further derives the loss formulation based on Levenberg-Marquardt to improve the robustness against bad initialization compared to the Gauss-Newton method. Inspired by D3VO [38], LM-Reloc also proposes a relative pose estimation network with a correlation layer [27] to regress a pose estimate which is used as the initialization for the optimization.

Feature-based visual localization. Most approaches for relocalization utilize feature detectors and descriptors, which can either be handcrafted, such as SIFT [17] or ORB [28], or especially in the context of drastic lighting and appearance changes can be learned. Recently, many descriptor learning methods have been proposed which follow a *detect-and-describe* paradigm, e.g., SuperPoint [8], D2-Net [10], or R2D2 [26]. Moreover, SuperGlue [32], a learning-based alternative to the matching step of feature-based methods has been proposed and yields significant performance improvements. For a complete relocalization pipeline the local pose refinement part has to be preceded by finding the closest image in a database given a query [3]. While some approaches [31, 30, 35] address the joint problem, in this work, we decouple these two tasks and only focus on the pose refinement part.

3. Method

In this work, we address the problem of computing the 6DoF pose $\boldsymbol{\xi} \in SE(3)$ between two given images I and I'. Furthermore, we assume that depths for a sparse set of points P are available, e.g., by running a direct visual SLAM system such as DSO [11].

The overall pipeline of our approach is shown in Figure 1. It is composed of LM-Net, CorrPoseNet, and a non-linear optimizer using the LM algorithm. LM-Net is trained with a novel loss formulation designed to learn feature descriptors optimally suited for the LM algorithm. The encoder-decoder architecture takes as input a reference image I as well as a target image I'. The network is trained end-to-end and will produce multi-scale feature maps F_l and F'_l, where $l = 1, 2, 3, 4$ denotes the different levels of the feature pyramid. In order to obtain an initial pose estimate for the non-linear optimization, we propose CorrPoseNet, which takes I and I' as the inputs and regress their relative pose. Finally, the multi-scale feature maps together with the depths obtained from DSO [11] form the non-linear energy function which is minimized using LM algorithm in a coarse-to-fine manner to obtain the final relative pose estimate. In the following, we will describe the individual components of our approach in more detail.

3.1. Direct Image Alignment with Levenberg-Marquardt

In order to optimize the pose $\boldsymbol{\xi}$ (consisting of rotation matrix $\mathbf{R}$ and translation $\mathbf{t}$), we minimize the feature-metric error:

$$E(\boldsymbol{\xi}) = \sum_{\mathbf{p} \in P} \left\| F'_l(\mathbf{p}') - F_l(\mathbf{p}) \right\|_\gamma , \quad (1)$$

where $||\cdot||_\gamma$ is the Huber norm and $\mathbf{p}'$ is the point projected onto the target image I' using the depths and the pose:

$$\mathbf{p}' = \Pi \left(\mathbf{R} \, \Pi^{-1}(\mathbf{p}, d_\mathbf{p}) + \mathbf{t} \right) . \quad (2)$$

This energy function is first minimized on the coarsest pyramid level 1, whose feature maps F_1 have a size of $(w/8, h/8)$, yielding a rough pose estimate. The estimate is refined by further minimizing the energy function on the subsequent pyramid levels 2, 3, and 4, where F_4 has the size of the original image (w, h). In the following, we provide details of the minimization performed in every level and for simplicity we will denote F_l as F from now on.

Minimization is performed using the Levenberg-Marquardt algorithm. In each iteration we compute the update $\boldsymbol{\delta} \in \mathbb{R}^6$ in the Lie algebra $\mathfrak{se}(3)$ as follows: Using the residual vector $\mathbf{r} \in \mathbb{R}^n$, the Huber weight matrix $\mathbf{W} \in \mathbb{R}^{n \times n}$, and the Jacobian of the residual vector with respect to the pose $\mathbf{J} \in \mathbb{R}^{n \times 6}$, we compute the Gauss-Newton system:

$$\mathbf{H} = \mathbf{J}^T \mathbf{W} \mathbf{J} \text{ and } \mathbf{b} = -\mathbf{J}^T \mathbf{W} \mathbf{r}. \quad (3)$$

The damped system can be obtained with either Levenberg's formula [16]:

$$\mathbf{H}' = \mathbf{H} + \lambda \mathbf{I} \quad (4)$$

or the Marquardt's formula [20]:

$$\mathbf{H}' = \mathbf{H} + \lambda \operatorname{diag}(\mathbf{H}) \quad (5)$$

depending on the specific application.

The update $\boldsymbol{\delta}$ and the pose $\boldsymbol{\xi}^i$ in the iteration i are computed as:

$$\boldsymbol{\delta} = \mathbf{H'}^{-1} \mathbf{b} \text{ and } \boldsymbol{\xi}^i = \boldsymbol{\delta} \boxplus \boldsymbol{\xi}^{i-1}, \quad (6)$$

where $\boxplus : \mathfrak{se}(3) \times SE(3) \rightarrow SE(3)$ is defined as in [11].

The parameter λ can be seen as an interpolation factor between gradient descent and the Gauss-Newton algorithm. When λ is high the method behaves like gradient descent with a small step size, and when it is low it is equivalent to the Gauss-Newton algorithm. In practice, we start with a relatively large λ and multiply it by 0.5 after a successful iteration, and by 4 after a failed iteration [11].

Figure 2 shows the typical behaviour of the algorithm. In the beginning the initial pose is inaccurate, resulting in projected point positions, which are a couple of pixels away from the correct location. λ will be high meaning that the algorithm will behave similar to gradient descent. After a couple of iterations, the pose got more accurate, and the projected points are in a closer vicinity to the correct location. By now, λ has probably decreased, so the algorithm will behave more similar to the Gauss-Newton algorithm. Now we expect the algorithm to converge quickly.

3.2. Loss Formulation for Levenberg-Marquardt

The key contribution of this work is LM-Net which provides feature maps F that improve the convergence behaviour of the LM algorithm and, in the meantime, are invariant to different conditions. We train our network in a Siamese fashion based on ground-truth pixel correspondences.

In this section, $\mathbf{p}$ denotes a reference point (located on image I) and the ground-truth correspondence (located on image I') is $\mathbf{p}'_{\text{gt}}$. For the loss functions explained below we further categorize $\mathbf{p}'$ into $\mathbf{p}'_{\text{neg}}$, $\mathbf{p}'_{\nabla}$, and $\mathbf{p}'_{\nabla^2}$, which is realized by using different negative correspondence sampling. Our loss formulation is inspired by the typical behaviour of the Levenberg-Marquardt algorithm explained in the previous section (see Figure 2). For a point, we distinguish four cases which can happen during the optimization:

1. The point is at the correct location ($\mathbf{p}'_{\text{gt}}$).

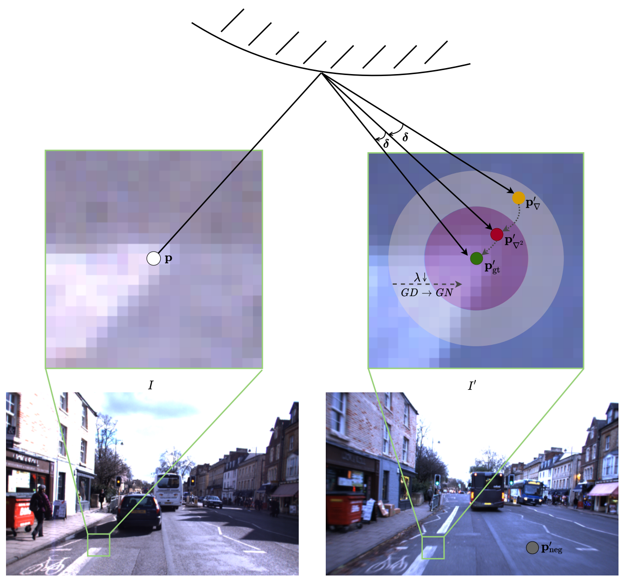

δ
δ
$\mathbf{p}$
$\mathbf{p}'_{\triangledown}$
$\mathbf{p}'_{\triangledown^2}$
$\mathbf{p}'_{\text{gt}}$
$\lambda \downarrow$
$GD \rightarrow GN$
I
I'
$\mathbf{p}'_{\text{neg}}$

Figure 2: Visualization of the typical behavior of direct image alignment with Levenberg-Marquardt. Initially, the projected point position (orange point, $\mathbf{p}'_{\triangledown}$) is far away from the correct solution (green point, $\mathbf{p}'_{\text{gt}}$), and λ is large, yielding an update step similar to gradient descent. After some iterations the projected point position gets closer to the optimum (red point, $\mathbf{p}'_{\triangledown^2}$) and at the same time λ will get smaller, leading to an update step similar to the Gauss-Newton algorithm. This is the intuition behind our point sampling strategy, where we utilize the ground-truth correspondence $\mathbf{p}'_{\text{gt}}$ for Equation (7), a negative $\mathbf{p}'_{\text{neg}}$ sampled across the whole image for Equation (8), a negative $\mathbf{p}'_{\triangledown}$ sampled in a far vicinity for Equation (12), and a negative $\mathbf{p}'_{\triangledown^2}$ sampled in a close vicinity for Equation (14).

2. The point is an outlier ($\mathbf{p}'_{\text{neg}}$).
3. The point is relatively far from the correct solution ($\mathbf{p}'_{\triangledown}$).
4. The point is very close to the correct solution ($\mathbf{p}'_{\triangledown^2}$).

In the following we will derive a loss function for each of the 4 cases:

1. The point is already at the correct location. In this case we would like the residual to be as small as possible, in the best case 0.

$$E_{\text{pos}} = \|F'(\mathbf{p}'_{\text{gt}}) - F(\mathbf{p})\|^2 \tag{7}$$

2. The point is an outlier or the pose estimate is completely wrong. In this case the projected point position can be at a completely different location than the correct correspondence. In this scenario we would like the residual of this pixel to be very large to reflect this, and potentially reject a wrong update. To enforce this property we sample a negative correspondences $\mathbf{p}'_{\text{neg}}$ uniformly across the whole image, and compute

$$E_{\text{neg}} = \max\left(M - \|F'(\mathbf{p}'_{\text{neg}}) - F(\mathbf{p})\|^2, 0\right) \tag{8}$$

where M is the margin how large we would like the energy of a wrong correspondence to be. In practice, we set it to 1.
3. The predicted pose is relatively far away from the optimum, meaning that the projected point position will be a couple of pixels away from the correct location. As this typically happens during the beginning of the optimization we assume that λ will be relatively large and the algorithm behaves similar to gradient descent. In this case we want that the gradient of this point is oriented in the direction of the correct solution, so that the point has a positive influence on the update step.

For computing a loss function to enforce this property we sample a random negative correspondence $\mathbf{p}'_{\triangledown}$ in a relatively large vicinity around the correct solution (in our experiments we use 5 pixels distance). Starting from this negative correspondence $\mathbf{p}'_{\triangledown}$ we first compute the 2×2 Gauss-Newton system for this individual point, similarly to how it is done for optical flow estimation using Lucas-Kanade:

$$\mathbf{r}_{\mathbf{p}}(\mathbf{p}, \mathbf{p}'_{\triangledown}) = \mathbf{F}'(\mathbf{p}'_{\triangledown}) - \mathbf{F}(\mathbf{p}) \tag{9}$$

$$\mathbf{J}_{\mathbf{p}} = \frac{d\mathbf{F}'(\mathbf{p}'_{\triangledown})}{d\mathbf{p}'_{\triangledown}} \text{ and } \mathbf{H}_{\mathbf{p}} = \mathbf{J}_{\mathbf{p}}^T \mathbf{J}_{\mathbf{p}} \text{ and } \mathbf{b}_{\mathbf{p}} = \mathbf{J}_{\mathbf{p}}^T \mathbf{r}_{\mathbf{p}} \tag{10}$$

We compute the damped system using a relatively large fixed λ_f, as well as the optical flow step[*]

$$\mathbf{H}'_{\mathbf{p}} = \mathbf{H}_{\mathbf{p}} + \lambda_f \, \mathbf{I} \text{ and } \mathbf{p}'_{\text{after}} = \mathbf{p}'_{\triangledown} + \mathbf{H}'^{-1}_{\mathbf{p}} \mathbf{b}_{\mathbf{p}}. \tag{11}$$

In order for this point to have a useful contribution to the direct image alignment, this update step should move in the correct direction by at least δ. We enforce this using a Gradient-Descent loss function which is small only if the distance to the correct correspondence *after* the update is smaller than before the update:

$$E_{\text{GD}} = \max\left(\|\mathbf{p}'_{\text{after}} - \mathbf{p}'_{\text{gt}}\|^2 - \|\mathbf{p}'_{\triangledown} - \mathbf{p}'_{\text{gt}}\|^2 + \delta, 0\right) \tag{12}$$

In practice, we choose $\lambda_f = 2.0$ and $\delta = 0.1$.
4. The predicted pose is very close to the optimum, yielding a projected point position in very close proximity of the correct correspondence, and typically λ will be very small, so the update will mostly be a Gauss-Newton step. In this case we would like the algorithm to converge as quickly as possible, with subpixel accuracy. We enforce this using the Gauss-Newton loss [36]. To compute it we first sample a random negative correspondence $\mathbf{p}'_{\triangledown^2}$ in a 1-pixel vicinity

[*]Here we use Equation (4) instead of Equation (5) since we find it more stable for training LM-Net.

around the correct location. Then we use Equations (9) and (10), replacing $\mathbf{p}'_{\nabla}$ with $\mathbf{p}'_{\nabla^2}$ to obtain the Gauss-Newton system formed by $\mathbf{H_p}$ and $\mathbf{b_p}$. We compute the updated pixel location:

$$\mathbf{p}'_{\text{after}} = \mathbf{p}'_{\nabla^2} + (\mathbf{H_p} + \epsilon\,\mathbf{I})^{-1}\mathbf{b_p} \tag{13}$$

Note that in contrast to the computation of the LM-Loss (Equation (12)), in this case ϵ is just added to ensure invertibility and therefore ϵ is much smaller than the λ_f used above. The Gauss-Newton loss is computed with:

$$\begin{aligned} E_{\text{GN}} = \frac{1}{2}(\mathbf{p}'_{\text{after}} - \mathbf{p}'_{\text{gt}})^T\mathbf{H_p}(\mathbf{p}'_{\text{after}} - \mathbf{p}'_{\text{gt}}) \\ + \log(2\pi) - \frac{1}{2}\log(|\mathbf{H_p}|) \end{aligned} \tag{14}$$

Note how all our 4 loss components use a different way to sample the involved points, depicted also in Figure 2. With the derivation above we argue that each loss component is important to achieve optimal performance and we demonstrate this in the results section. Note that the Gauss-Newton systems computed for the GD-Loss and the GN-Loss are very relevant for the application of direct image alignment. In fact the full Gauss-Newton system containing all points (Equation (3)), can be computed from these individual Gauss-Newton systems (Equation (10)) by simply summing them up and multiplying them with the derivative with respect to the pose [36].

3.3. CorrPoseNet

In order to deal with the large baselines between the images, we propose CorrPoseNet to regress the relative pose between two images I and I', which serves as the initialization of LM optimization. As our network shall work even in cases of large baselines and strong rotations, we utilize the correlation layer proposed in [27] which is known to boost the performance of affine image transformation and optical flow [21] estimation for large displacements, but has not been applied to pose estimation before.

Our network first computes deep features $\mathbf{f}_{\text{corr}}, \mathbf{f}'_{\text{corr}} \in \mathbb{R}^{h\times w\times c}$ from both images individually using multiple strided convolutions with ReLU activations in between. Then the correlation layer correlates each pixel from the normalized source features with each pixel from the normalized target features yielding the correlation map $\mathbf{c} \in \mathbb{R}^{h\times w\times(h\times w)}$:

$$\mathbf{c}(i,j,(i',j')) = \mathbf{f}_{\text{corr}}(i,j)^T\mathbf{f}'_{\text{corr}}(i',j') \tag{15}$$

The correlation map is then normalized in the channel dimension and fed into 2 convolutional layers each followed by batch norm and ReLU. Finally we regress the Euler angle $\mathbf{r}^{\text{euler}}$ and translation $\mathbf{t}$ using a fully connected layer. More details on the architecture are shown in the supplementary material.

We train CorrPoseNet from scratch with image pairs and groundtruth poses $\mathbf{r}^{\text{euler}}_{\text{gt}}, \mathbf{t}_{\text{gt}}$. We utilize an L2-loss working directly on Euler angles and translation:

$$E = \|\mathbf{t} - \mathbf{t}_{\text{gt}}\|_2 + \lambda\|\mathbf{r}^{\text{euler}} - \mathbf{r}^{\text{euler}}_{\text{gt}}\|_2, \tag{16}$$

where λ is the weight, which we set to 10 in practice.

As the distribution of groundtruth poses in the Oxford training data is limited we apply the following data augmentation. We first generate dense depths for all training images using a state-of-the-art dense stereo matching algorithm [40]. The resulting depths are then used to warp the images to a different pose sampled from a uniform distribution. In detail, we first warp the depth image to the random target pose, then inpaint the depth image using the OpenCV implementation of Navier Stokes, and finally warp our image to the target pose using this depth map. Note that the dense depths are only necessary for training, not for evaluation. We show an ablation study on the usage of correlation layers and the proposed data augmentation in the supplementary material.

4. Experiments

We evaluate our method on the relocalization tracking benchmark proposed in [36], which contains images created with the CARLA simulator [9], and scenes from the Oxford RobotCar dataset [19]. We train our method on the respective datasets from scratch. LM-Net is trained using the Adam optimizer with a learning rate of 10^{-6} and for CorrPoseNet we use a learning rate of 10^{-4}. For both networks we choose hyperparameters and epoch based on the results on the validation data. Our networks use the same hyperparameters for all experiments except where stated otherwise; the direct image alignment code is slightly adapted for Oxford RobotCar, mainly to improve performance when the ego-vehicle is standing still.

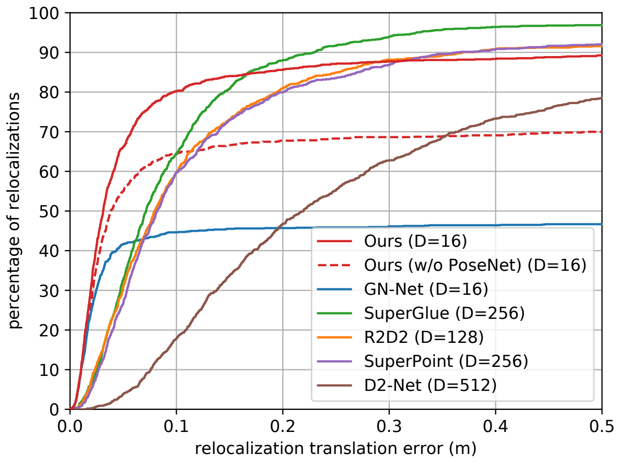
As the original relocalization tracking benchmark [36] does not include validation data on Oxford RobotCar we have manually aligned two new sequences, namely *2015-04-17-09-06-25* and *2015-05-19-14-06-38*, and extend the benchmark with these sequences as validation data.

Evaluation metrics: We evaluate the predicted translation $\mathbf{t}_{\text{est}}$ and rotation $\mathbf{R}_{\text{est}}$ against the ground-truth $\mathbf{t}_{\text{gt}}$ and $\mathbf{R}_{\text{gt}}$ according to Equations (17) and (18).

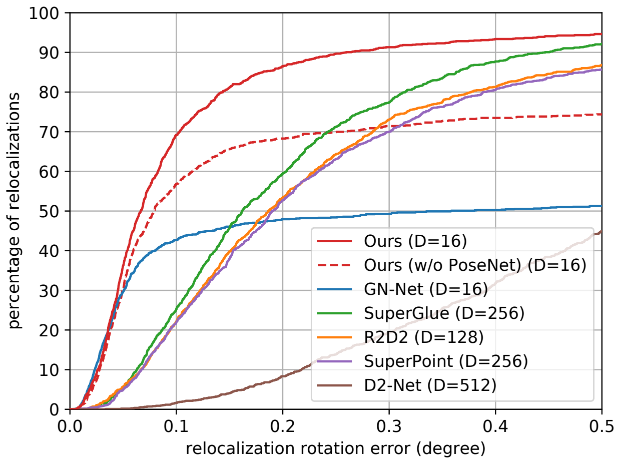
$$t_\Delta = \|\mathbf{t}_{\text{est}} - \mathbf{t}_{\text{gt}}\|_2 \tag{17}$$

$$R_\Delta = \arccos\left(\frac{\text{trace}(\mathbf{R}_{\text{est}}^{-1}\mathbf{R}_{\text{gt}}) - 1}{2}\right) \tag{18}$$

In this section, we plot the cumulative translation and rotation error until 0.5m and 0.5°, respectively. For quantitative results we compute the area under curve (AUC) of these



(a) Translation error.



(b) Rotation error.

Figure 3: Results on the CARLA relocalization tracking benchmark test data [36]. For each error threshold we show the percentage of relocalizations (cumulative error plot) for LM-Reloc (ours) and other state-of-the-art methods. Compared to the indirect methods our approach exhibits significantly better accuracy in both translation and rotation, while having a similar robustness. Compared to GN-Net, the novel loss formulation (see red dashed line), and the CorrPoseNet (see red line) both boost the robustness. D is the feature dimensionality.

cumulative curves in percent, which we denote as t_{AUC} for translation and R_{AUC} for rotation from now on.

We evaluate the following direct methods:

Ours: The full LM-Reloc approach consisting of CorrPoseNet, LM-Net features and direct image alignment based on Levenberg-Marquardt. The depths used for the image alignment are estimated with the stereo version [37] of DSO [11].

Ours (w/o CorrPoseNet): For a more fair comparison to GN-Net we use identity as initialization for the direct image alignment instead of CorrPoseNet. This enables a direct comparison between the two loss formulations.

GN-Net [36]: In this work, we have also improved the parameters of the direct image alignment pipeline based on DSO [11]. Thus we have re-evaluated GN-Net with this improved pipeline to make the comparison as fair as possible. These re-evaluated results are better than the results computed in the original GN-Net paper.

Baseline methods: Additionally, we evaluate against current state-of-the-art indirect methods, namely SuperGlue [32], R2D2 [26], SuperPoint [8], and D2-Net [10]. For these methods, we estimate the relative pose using the models provided by the authors and the OpenCV implementation of solvePnPRansac. We have tuned the parameters of RANSAC on the validation data and used 1000 iterations and a reprojection error threshold of 3 for all methods. For estimating depth values at keypoint locations we use OpenCV stereo matching. It would be possible to achieve a higher accuracy by using SfM and MVS solutions such as COLMAP [34]. However, one important disadvantage of these approaches is, that building a map is rather time consuming and computationally expensive, whereas all other approaches evaluated on the benchmark [36] are able to create the map close to real-time, enabling applications like long-term loop-closure and map-merging.

Table 1: This table shows the AUC until 0.5 meters / 0.5 degrees for the relocalization error on the CARLA relocalization tracking benchmark test data. Powered by our novel loss formulation and the combination with CorrPoseNet, LM-Reloc achieves lower rotation and translation errors compared to the state-of-the-art.

Method	t_{AUC}	R_{AUC}
Ours	**80.65**	**77.83**
SuperGlue [32]	78.99	59.31
R2D2 [26]	73.47	54.42
SuperPoint [8]	72.76	53.38
D2-Net [10]	47.62	16.47
Ours (w/o CorrPoseNet)	63.88	61.9
GN-Net [36]	43.72	44.08

4.1. CARLA Relocalization Benchmark

Figure 3 depicts the results on the test data of the CARLA benchmark. For all methods we show the cumulative error plot for translation in meters and rotation in degree. It can be seen that our method is more accurate than the state-of-the-art while performing similarly in terms of robustness. We also show the AUC for the two Figures in Table 1. Compared to GN-Net it can be seen that our new loss formulation significantly improves the results,

Table 2: Results on the Oxford RobotCar relocalization tracking benchmark [36]. We compare LM-Net (Ours) against other state-of-the-art methods (SuperGlue, R2D2, SuperPoint, and D2-Net). As can be seen from the results, our method almost consistently outperforms other SOTA approaches in terms of rotation AUC whilst achieving comparable results on translation AUC.

Sequence	Ours		SuperGlue [32]		R2D2 [26]		SuperPoint [8]		D2-Net [10]	
	t_{AUC}	R_{AUC}	t_{AUC}	R_{AUC}	t_{AUC}	R_{AUC}	t_{AUC}	R_{AUC}	t_{AUC}	R_{AUC}
Sunny-Overcast	79.83	**55.48**	**81.01**	52.83	80.86	53.57	78.95	50.03	71.93	39.0
Sunny-Rainy	71.54	**43.7**	**75.58**	40.59	74.84	41.23	69.76	37.12	65.63	27.5
Sunny-Snowy	59.69	**44.06**	**63.57**	43.64	62.92	41.78	60.85	40.02	55.65	30.86
Overcast-Rainy	80.54	**63.7**	79.99	61.64	**81.29**	61.23	80.36	61.56	75.66	51.06
Overcast-Snowy	55.38	47.88	57.67	47.16	**57.68**	**48.41**	55.39	44.96	51.17	34.54
Rainy-Snowy	68.57	**41.67**	69.91	39.87	**71.79**	39.86	67.7	38.05	61.91	27.74

Table 3: This table shows the results on the Oxford RobotCar relocalization tracking benchmark test data against GN-Net. Thanks to our LM-based loss formulation we consistently outperform GN-Net on all sequences.

Sequence	Ours (w/o CorrPoseNet)		GN-Net [36]	
	t_{AUC}	R_{AUC}	t_{AUC}	R_{AUC}
Sunny-Overcast	**79.61**	**55.45**	73.53	49.31
Sunny-Rainy	**70.46**	**42.86**	64.58	37.27
Sunny-Snowy	**59.7**	**44.17**	55.27	41.36
Overcast-Rainy	**79.67**	**63.08**	75.72	60.13
Overcast-Snowy	**54.94**	**47.19**	51.34	42.91
Rainy-Snowy	**66.23**	**39.93**	62.63	36.2

even when used without the CorrPoseNet as initialization. The figure conveys that the direct methods (Ours, GN-Net) are more accurate than the evaluated indirect methods.

4.2. Oxford RobotCar Relocalization Benchmark

We compare to the state-of-the-art indirect methods on the 6 test sequence pairs consisting of the sequences *2015-02-24-12-32-19* (sunny), *2015-03-17-11-08-44* (overcast), *2014-12-05-11-09-10* (rainy), and *2015-02-03-08-45-10* (snowy). In Table 2, we show the area under curve until 0.5 meters / 0.5 degrees for all methods. It can be seen that our method clearly outperforms the state-of-the-art in terms of rotation accuracy, while being competitive in terms of translation error. It should be noted that the ground-truth for these sequences was generated using ICP alignment of the 2D-LiDAR data accumulated for 60 meters. We have computed that the average root mean square error of the ICP alignment is 16 centimeters. Therefore, especially the ground-truth translations have limited accuracy. As can be seen from Figure 3, the accuracy improvements our method provides are especially visible in the range below 0.15 meters which is hard to measure on this dataset. The rotation error of LiDAR alignment is lower than the translational

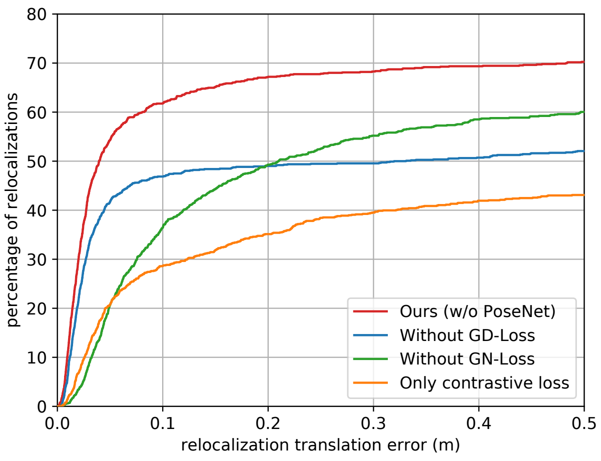


Figure 4: This plot shows our ablation study for removing different loss parts on the CARLA relocalization tracking benchmark. Without the GD-loss the achieved robustness is reduced, whereas removing the GN-loss leads to decreased accuracy. Using our full loss formulation yields a large improvement.

one, which is why we clearly observe the improvements of our method on the rotations.

In Table 3, we compare LM-Net without the CorrPoseNet to GN-Net. Due to our novel loss formulation LM-Net outperforms the competitor on all sequences significantly.

4.3. Ablation Studies

We evaluate LM-Net on the CARLA validation data with and without the various losses (Figure 4). Compared to a normal contrastive loss, the given loss formulation is a large improvement. As expected, E_{GD} (green line) mainly improves the robustness, whereas E_{GN} (blue line) improves the accuracy. Only when used together (our method) we achieve large robustness and large accuracy, confirming our theoretical derivation in Section 3.

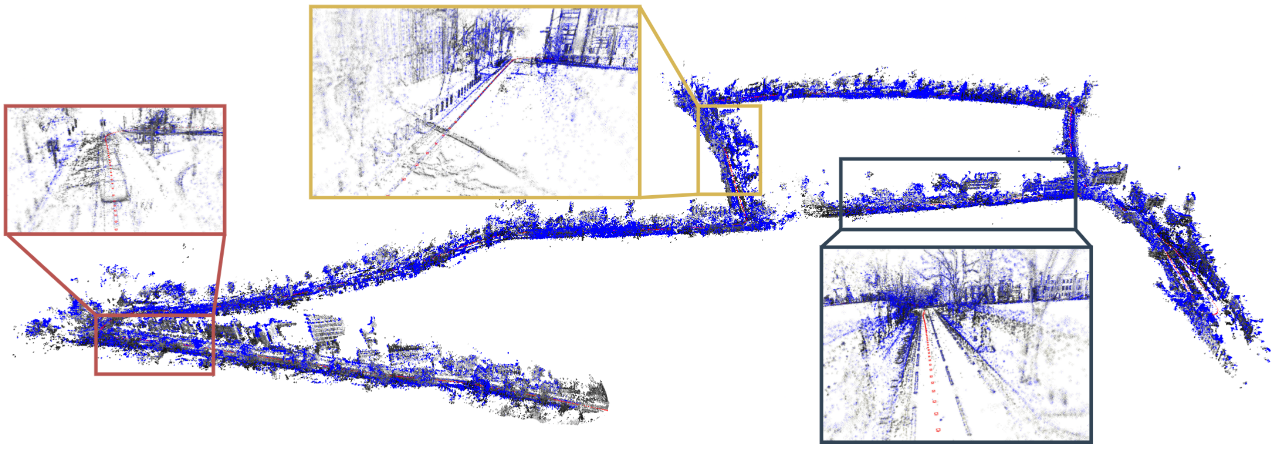


Figure 5: This figure shows a point cloud from a sunny reference map (grey points) overlayed with the point cloud from a relocalized snowy sequence (blue points). The well aligned point clouds demonstrate the high relocalization accuracy of LM-Reloc.

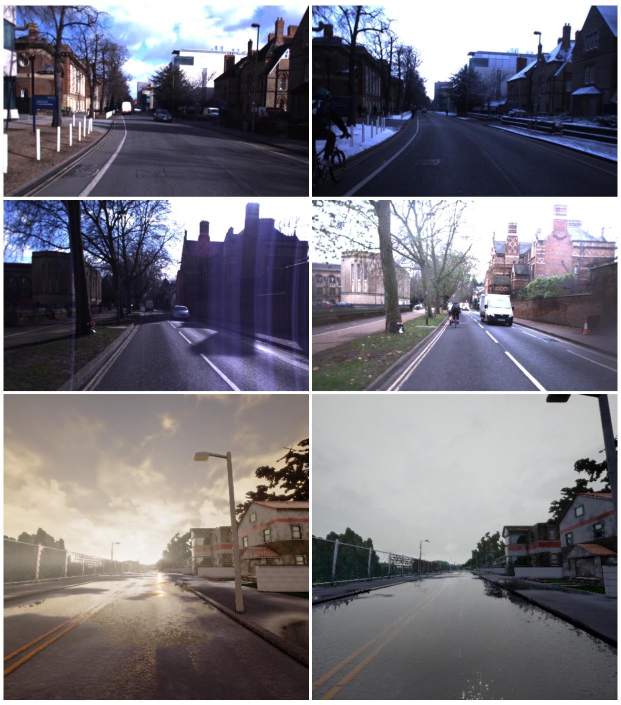


Figure 6: Example image pairs from the relocalization tracking benchmark which have been successfully relocalized by LM-Reloc (with an accuracy of better than 10 cm). Top row: Oxford sunny against snowy condition, middle row: Oxford sunny against rainy condition, bottom row: CARLA benchmark.

4.4. Qualitative Results

To demonstrate the accuracy of our approach in practice, we show qualitative results on the Oxford RobotCar dataset. We track the snowy test sequence *2015-02-03-08-45-10* using Stereo DSO [37] and at the same time perform relocalization against the sunny reference map *2015-02-24-12-32-19*. Relocalization between the current keyframe and the closest map image is performed using LM-Net. Initially, we give the algorithm the first corresponding map image (which would in practice be provided by an image retrieval approach such as NetVLAD [3]). Afterwards we find the closest map image for each keyframe using the previous solution for the transformation between the map and the current SLAM world T_{w_m}. We visualize the current point cloud (blue) and the point cloud from the map (grey) overlayed using the smoothed T_{w_m} (Figure 5). The point clouds will align only if the relocalization is accurate. As can be seen in Figure 5, the lane markings, poles, and buildings between the reference and query map align well, hence qualitatively showing the high relocalization accuracy of our method. We recommend watching the video at `https://vision.in.tum.de/lm-reloc`. In Figure 6 we show example images from the benchmark.

5. Conclusion

We have presented LM-Reloc as a novel approach for direct visual localization. In order to estimate the relative 6DoF pose between two images from different conditions, our approach performs direct image alignment on the trained features from LM-Net without relying on feature matching or RANSAC. In particular, with the loss function designed seamlessly for the Levenberg-Marquart algorithm, LM-Net provides deep feature maps that coin the characteristics of direct image alignment and are also invariant to changes in lighting and appearance of the scene. The experiments on the CARLA and Oxford RobotCar relocalization tracking benchmark exhibit the state-of-the-art performance of our approach. In addition, the ablation studies also show the effectiveness of the different components of LM-Reloc.

References

[1] H. Alismail, B. Browning, and S. Lucey. Photometric bundle adjustment for vision-based SLAM. In *ACCV*, 2017.

[2] H. Alismail, M. Kaess, B. Browning, and S. Lucey. Direct visual odometry in low light using binary descriptors. *RA-L*, 2, 2017.

[3] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. NetVLAD: CNN architecture for weakly supervised place recognition. In *CVPR*, 2016.

[4] M. A. Brubaker, A. Geiger, and R. Urtasun. Map-based probabilistic visual self-localization. *PAMI*, 38(4):652–665, 2015.

[5] C.-H. Chang, C.-N. Chou, and E. Y. Chang. CLKN: Cascaded lucas-kanade networks for image alignment. In *CVPR*, pages 2213–2221, 2017.

[6] M. Cummins and P. Newman. FAB-MAP: probabilistic localization and mapping in the space of appearance. *IJRR*, 27(6), 2008.

[7] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *PAMI*, 29(6):1052–1067, 2007.

[8] D. DeTone, T. Malisiewicz, and A. Rabinovich. SuperPoint: self-supervised interest point detection and description. In *CVPRW*, 2018.

[9] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: an open urban driving simulator. In *CoRL*, 2017.

[10] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler. D2-Net: a trainable CNN for joint description and detection of local features. In *CVPR*, 2019.

[11] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *PAMI*, 40(3), 2018.

[12] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *ECCV*, 2014.

[13] C. Kerl, J. Sturm, and D. Cremers. Dense visual SLAM for RGB-D cameras. In *IROS*, pages 2100–2106. IEEE, 2013.

[14] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *ISMAR*, pages 225–234. IEEE, 2007.

[15] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[16] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2(2):164–168, 1944.

[17] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[18] Z. Lv, F. Dellaert, J. M. Rehg, and A. Geiger. Taking a deeper look at the inverse compositional algorithm. In *CVPR*, pages 4581–4590, 2019.

[19] W. Maddern, G. Pascoe, C. Linegar, and P. Newman. 1 Year, 1000km: The Oxford RobotCar Dataset. *IJRR*, 36(1), 2017.

[20] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[21] I. Melekhov, A. Tiulpin, T. Sattler, M. Pollefeys, E. Rahtu, and J. Kannala. Dgc-net: Dense geometric correspondence network. In *WACV*, pages 1034–1042. IEEE, 2019.

[22] R. Mur-Artal, J. M. Montiel, and J. D. Tardos. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE T-RO*, 31(5), 2015.

[23] R. Mur-Artal and J. D. Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE T-RO*, 33(5), 2017.

[24] T. Ort, L. Paull, and D. Rus. Autonomous vehicle navigation in rural environments without detailed prior maps. In *ICRA*, pages 2040–2047. IEEE, 2018.

[25] G. Pascoe, W. Maddern, M. Tanner, P. Piniés, and P. Newman. Nid-slam: Robust monocular slam using normalised information distance. In *CVPR*, pages 1435–1444, 2017.

[26] J. Revaud, C. De Souza, M. Humenberger, and P. Weinzaepfel. R2d2: Reliable and repeatable detector and descriptor. In *NeurIPS*, pages 12405–12415, 2019.

[27] I. Rocco, R. Arandjelovic, and J. Sivic. Convolutional neural network architecture for geometric matching. In *CVPR*, pages 6148–6157, 2017.

[28] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to SIFT or SURF. In *ICCV*, pages 2564–2571. Ieee, 2011.

[29] S. Saeedi, B. Bodin, H. Wagstaff, A. Nisbet, L. Nardi, J. Mawer, N. Melot, O. Palomar, E. Vespa, T. Spink, et al. Navigating the landscape for real-time localization and mapping for robotics and virtual and augmented reality. *Proceedings of the IEEE*, 106(11):2020–2039, 2018.

[30] P.-E. Sarlin, C. Cadena, R. Siegwart, and M. Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *CVPR*, pages 12716–12725, 2019.

[31] P.-E. Sarlin, F. Debraine, M. Dymczyk, R. Siegwart, and C. Cadena. Leveraging deep visual descriptors for hierarchical efficient localization. In *CoRL*, 2018.

[32] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich. Superglue: Learning feature matching with graph neural networks. In *CVPR*, pages 4938–4947, 2020.

[33] T. Sattler, W. Maddern, C. Toft, A. Torii, L. Hammarstrand, E. Stenborg, D. Safari, M. Okutomi, M. Pollefeys, J. Sivic, F. Kahl, and T. Pajdla. Benchmarking 6DOF outdoor visual localization in changing conditions. In *CVPR*, 2018.

[34] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *CVPR*, 2016.

[35] H. Taira, M. Okutomi, T. Sattler, M. Cimpoi, M. Pollefeys, J. Sivic, T. Pajdla, and A. Torii. Inloc: Indoor visual localization with dense matching and view synthesis. In *CVPR*, pages 7199–7209, 2018.

[36] L. von Stumberg, P. Wenzel, Q. Khan, and D. Cremers. GN-Net: The Gauss-Newton Loss for Multi-Weather Relocalization. *RA-L*, 5(2):890–897, 2020.

[37] R. Wang, M. Schwörer, and D. Cremers. Stereo dso: Large-scale direct sparse visual odometry with stereo cameras. In *ICCV*, 2017.

[38] N. Yang, L. v. Stumberg, R. Wang, and D. Cremers. D3VO: Deep depth, deep pose and deep uncertainty for monocular visual odometry. In *CVPR*, pages 1281–1292, 2020.

[39] N. Yang, R. Wang, X. Gao, and D. Cremers. Challenges in monocular visual odometry: Photometric calibration, motion bias, and rolling shutter effect. *RA-L*, 3(4):2878–2885, 2018.

[40] F. Zhang, V. Prisacariu, R. Yang, and P. H. Torr. Ga-net: Guided aggregation net for end-to-end stereo matching. In *CVPR*, pages 185–194, 2019.

[41] X. Zheng, Z. Moratto, M. Li, and A. I. Mourikis. Photometric patch-based visual-inertial odometry. In *ICRA*, pages 3264–3271, 2017.

LM-Reloc: Levenberg-Marquardt Based Direct Visual Relocalization: Supplementary Material

Lukas von Stumberg[1,2*] Patrick Wenzel[1,2*] Nan Yang[1,2] Daniel Cremers[1,2]
[1] Technical University of Munich [2] Artisense

A. Video

As mentioned in the paper, we provide a video of the qualitative relocalization demo, which is available at https://vision.in.tum.de/lm-reloc.

B. Network Architecture

CorrPoseNet. The CorrPoseNet takes 2 images (I and I') as the input and outputs the relative pose R, t between those images. The overall network architecture of the CorrPoseNet is depicted in Figure 1. The convolutional blocks consist of in total 9 convolutional layers followed by ReLU activations. The architectural details of the convolutional blocks are listed in Table 1. The correlation layer which takes the output of the convolutional blocks as input is described in the main paper. The correlation layer is followed by the regression block which regresses the relative pose. The layers of the regression block are listed in Table 2. The output of the network is the rotation R as Euler angles and translation t.

Table 1: Network architecture and parameters of the convolutional blocks. **k** denotes kernel size, **s** stride, and **p** padding.

Convolutional blocks						
layer	**in-chns**	**out-chns**	**k**	**s**	**p**	**activation**
conv0	3	16	16	2	3	ReLU
conv1	16	32	5	2	2	ReLU
conv2	32	64	3	2	1	ReLU
conv3	64	64	3	1	0	ReLU
conv4	64	128	3	2	2	ReLU
conv5	128	128	3	1	1	ReLU
conv6	128	256	3	2	1	ReLU
conv7	256	256	3	1	1	ReLU

LM-Net. We adopt U-Net [1] as the encoder of LM-Net. However, we change the decoder part of the architecture in the following way. Starting from the coarsest level, we upsample (with bilinear interpolation) the feature maps by 2 and concatenate those feature maps with the feature map of the higher level. This is followed by 1×1 convolutional filters. This procedure is repeated 4 times. This results in the feature pyramid maps as described in Table 3.

*Equal contribution.

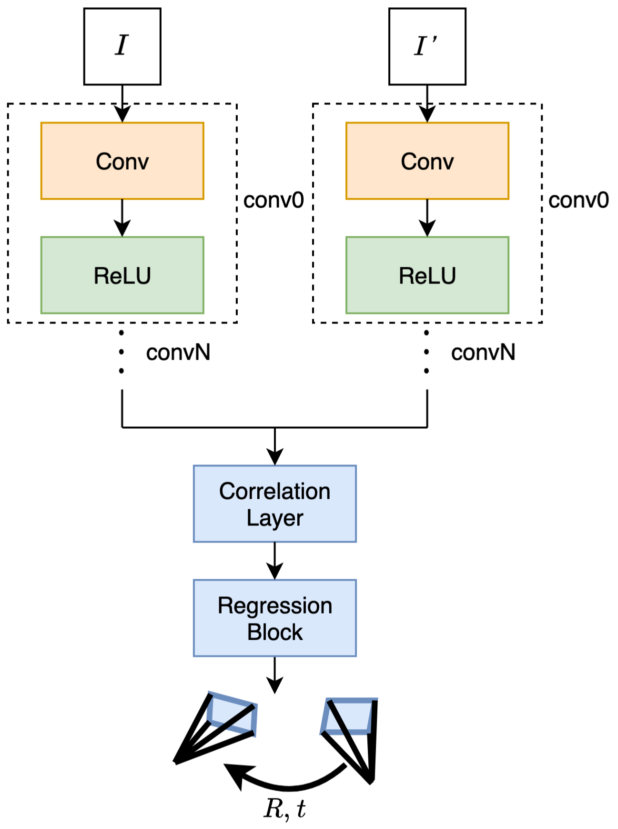
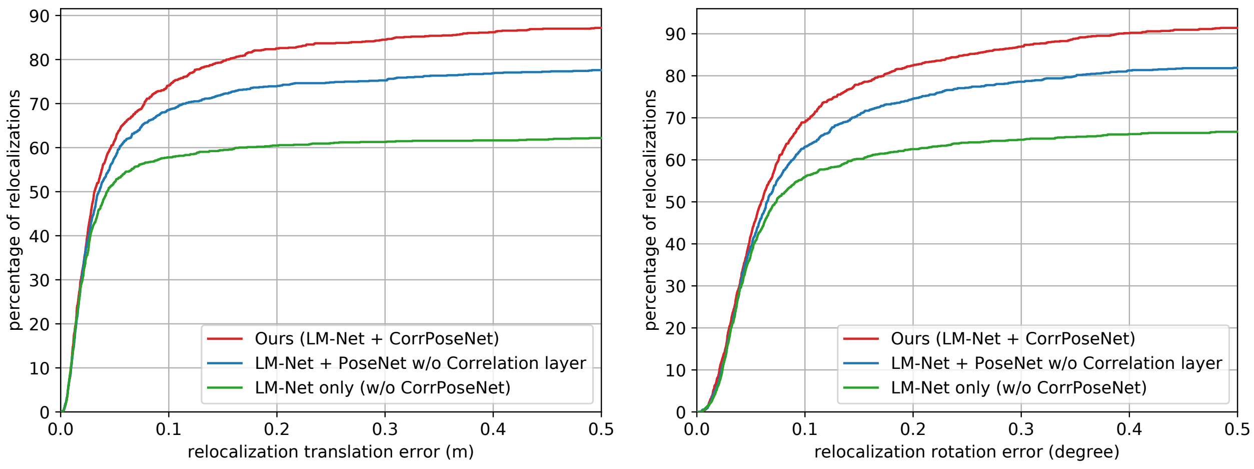


Figure 1: Network architecture of CorrPoseNet.

C. Ablation Study Correlation Layer

We demonstrate the impact of the Correlation layer in the proposed CorrPoseNet. We compare it to a simpler pose estimation network where the correlation and regression layers are replaced with two 1×1 convolutions with 3 output-channels each, which directly regress rotation and Euler angles. This simpler PoseNet has one more convo-



(a) Translation error. **(b)** Rotation error.

Figure 2: Cumulative error plot for relocalization on the CARLA relocalization benchmark validation data [2]. It can be seen that the correlation layer in CorrPoseNet has a large impact on the performance.

Table 2: Network architecture and parameters of the regression block. **k** denotes kernel size, **s** stride, and **p** padding. $N_c = 256$ denotes the input channels for the CARLA model, and $N_o = 260$ denotes the input channels for the Oxford model, respectively.

Regression block						
layer	**in-chns**	**out-chns**	**k**	**s**	**p**	**activation**
conv0	N_c / N_o	128	7	1	0	ReLU
BN	128	128	-	-	-	
conv1	128	64	5	1	0	ReLU
BN	64	64	-	-	-	
FC	2304	6	-	-	-	

Table 3: Output of the decoder of LM-Net. H, and W denote height and width of the feature maps.

Decoder layer	**Output size**
F_1	$16 \times H/8 \times W/8$
F_2	$16 \times H/4 \times W/4$
F_3	$16 \times H/2 \times W/2$
F_4	$16 \times H \times W$

lutional block conv8 with 512 output channels, kernel size 3, stride 2, and padding 1. Otherwise the network architecture and parameters are the same as for CorrPoseNet. The results on the CARLA validation data are shown in Figure 2. Even the simpler pose estimation network (PoseNet w/o Correlation layer) improves the result over using identity as an initialization for the direct image alignment (LM-Net only). However, utilizing the correlation layer significantly boosts the performance.

Table 4: This table shows the AUC until 0.5 meters / 0.5 degrees for the relocalization error on the Oxford validation sequences. Our data augmentation (which warps the images using random poses) improves both rotation and translation error.

Method	t_{AUC}	R_{AUC}
Ours	**80.45**	**65.11**
Ours w/o data augmentation	80.15	64.58

D. Ablation Study Oxford Data Augmentation

We show the impact of the data augmentation for the Oxford RobotCar Relocalization benchmark, where we warp the images to different poses using dense depths in Table 4. It can be seen that the proposed augmentation improves translation and rotation error on the validation data.

References

[1] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *MICCAI*, 2015. 1

[2] L. von Stumberg, P. Wenzel, Q. Khan, and D. Cremers. GN-Net: The Gauss-Newton Loss for Multi-Weather Relocalization. *RA-L*, 5(2):890–897, 2020. 2

B ABSTRACTS OF ADDITIONAL PUBLICATIONS

B.1 SELF-SUPERVISED STEERING ANGLE PREDICTION FOR VEHICLE CONTROL USING VISUAL ODOMETRY

Qadeer Khan, Patrick Wenzel, and Daniel Cremers

Self-Supervised Steering Angle Prediction for Vehicle Control Using Visual Odometry

International Conference on Artificial Intelligence and Statistics (AISTATS), 2021

PMLR 130:3781-3789

Abstract

Vision-based learning methods for self-driving cars have primarily used supervised approaches that require a large number of labels for training. However, those labels are usually difficult and expensive to obtain. In this paper, we demonstrate how a model can be trained to control a vehicle's trajectory using camera poses estimated through visual odometry methods in an entirely self-supervised fashion. We propose a scalable framework that leverages trajectory information from several different runs using a camera setup placed at the front of a car. Experimental results on the CARLA simulator demonstrate that our proposed approach performs at par with the model trained with supervision.

B.2 VISION-BASED MOBILE ROBOTICS OBSTACLE AVOIDANCE WITH DEEP REINFORCEMENT LEARNING

Patrick Wenzel, Torsten Schön, Laura Leal-Taixé, and Daniel Cremers

Vision-Based Mobile Robotics Obstacle Avoidance With Deep Reinforcement Learning

IEEE International Conference on Robotics and Automation (ICRA), 2021

DOI: 10.1109/ICRA48506.2021.9560787

Abstract

Obstacle avoidance is a fundamental and challenging problem for autonomous navigation of mobile robots. In this paper, we consider the problem of obstacle avoidance in simple 3D environments where the robot has to solely rely on a single monocular camera. In particular, we are interested in solving this problem without relying on localization, mapping, or planning techniques. Most of the existing work consider obstacle avoidance as two separate problems, namely obstacle detection, and control. Inspired by the recent advantages of deep reinforcement learning in Atari games and understanding highly complex situations in Go, we tackle the obstacle avoidance problem as a data-driven end-to-end deep learning approach. Our approach takes raw images as input and generates control commands as output. We show that discrete action spaces are outperforming continuous control commands in terms of expected average reward in maze-like environments. Furthermore, we show how to accelerate the learning and increase the robustness of the policy by incorporating predicted depth maps by a generative adversarial network.

OWN PUBLICATIONS

[1] Qadeer Khan, Patrick Wenzel, and Daniel Cremers. “Self-Supervised Steering Angle Prediction for Vehicle Control Using Visual Odometry.” In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2021 (cited on p. 10).

[2] Qadeer Khan, Patrick Wenzel, Daniel Cremers, and Laura Leal-Taixé. “Towards Generalizing Sensorimotor Control Across Weather Conditions.” In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019 (cited on pp. 7, 9–11, 45, 115, 135).

[3] Lukas von Stumberg, Patrick Wenzel, Qadeer Khan, and Daniel Cremers. “GN-Net: The Gauss-Newton Loss for Multi-Weather Relocalization.” In: *IEEE Robotics and Automation Letters (RA-L)* 5.2 (2020), pp. 890–897 (cited on pp. 7, 9–11, 61, 80, 89, 93, 94, 99–104, 115, 143).

[4] Lukas von Stumberg, Patrick Wenzel, Nan Yang, and Daniel Cremers. “LM-Reloc: Levenberg-Marquardt Based Direct Visual Relocalization.” In: *Proceedings of the International Conference on 3D Vision (3DV)*. 2020 (cited on pp. 7, 9, 10, 12, 91, 115, 169).

[5] Patrick Wenzel, Qadeer Khan, Daniel Cremers, and Laura Leal-Taixé. “Modular Vehicle Control for Transferring Semantic Information Between Weather Conditions Using GANs.” In: *Conference on Robot Learning (CoRL)*. 2018 (cited on pp. 7, 9, 10, 31, 47, 48, 54–57, 115).

[6] Patrick Wenzel, Torsten Schön, Laura Leal-Taixé, and Daniel Cremers. “Vision-Based Mobile Robotics Obstacle Avoidance With Deep Reinforcement Learning.” In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2021 (cited on p. 10).

[7] Patrick Wenzel, Rui Wang, Nan Yang, Qing Cheng, Qadeer Khan, Lukas von Stumberg, Niclas Zeller, and Daniel Cremers. “4Seasons: A Cross-Season Dataset for Multi-Weather SLAM in Autonomous Driving.” In: *Proceedings of the German Conference on Pattern Recognition (GCPR)*. 2020 (cited on pp. 7, 9, 10, 12, 79, 115, 153).

[8] Hatem Alismail, Brett Browning, and Simon Lucey. "Photometric Bundle Adjustment for Vision-Based SLAM." In: *Proceedings of the Asian Conference on Computer Vision (ACCV)*. 2016 (cited on pp. 62, 64, 92).

[9] Hatem Alismail, Michael Kaess, Brett Browning, and Simon Lucey. "Direct Visual Odometry in Low Light Using Binary Descriptors." In: *IEEE Robotics and Automation Letters (RA-L)* 2.2 (2017), pp. 444–451 (cited on pp. 64, 68, 94).

[10] Peter Anderson et al. "On Evaluation of Embodied Navigation Agents." In: *arXiv preprint arXiv:1807.06757*. 2018 (cited on pp. 48, 53, 58).

[11] Adrien Angeli, David Filliat, Stéphane Doncieux, and Jean-Arcady Meyer. "Fast and Incremental Method for Loop-Closure Detection Using Bags of Visual Words." In: *IEEE Transactions on Robotics (T-RO)* 24.5 (2008), pp. 1027–1037 (cited on p. 89).

[12] Relja Arandjelović, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. "NetVLAD: CNN architecture for weakly supervised place recognition." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 (cited on pp. 63, 89, 94, 105).

[13] Relja Arandjelović and Andrew Zisserman. "All about VLAD." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2013 (cited on p. 89).

[14] José-Luis Blanco-Claraco, Francisco-Ángel Moreno-Dueñas, and Javier González-Jiménez. "The Málaga urban dataset: High-rate stereo and LiDAR in a realistic urban scenario." In: *International Journal of Robotics Research (IJRR)* 33.2 (2014), pp. 207–214 (cited on p. 81).

[15] Mariusz Bojarski et al. "End to End Learning for Self-Driving Cars." In: *arXiv preprint arXiv:1604.07316*. 2016 (cited on pp. 4, 33, 46, 53).

[16] Marcus A. Brubaker, Andreas Geiger, and Raquel Urtasun. "Map-Based Probabilistic Visual Self-Localization." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 38.4 (2016), pp. 652–665 (cited on p. 92).

[17] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W. Achtelik, and Roland Siegwart. "The EuRoC micro aerial vehicle datasets." In: *International Journal of Robotics Research (IJRR)* 35.10 (2016), pp. 1157–1163 (cited on pp. 63, 70, 76, 82).

[18] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. "nuScenes: A multimodal dataset for autonomous driving." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020 (cited on pp. 80, 82).

[19] Che-Han Chang, Chun-Nan Chou, and Edward Y. Chang. "CLKN: Cascaded Lucas-Kanade Networks for Image Alignment." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 (cited on pp. 64, 94).

[20] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. "DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving." In: *Proceedings of the International Conference on Computer Vision (ICCV)*. 2015 (cited on pp. 33, 48).

[21] Bi-ke Chen, Chen Gong, and Jian Yang. "Importance-Aware Semantic Segmentation for Autonomous Vehicles." In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 2017 (cited on p. 34).

[22] Christopher B. Choy, JunYoung Gwak, Silvio Savarese, and Manmohan Chandraker. "Universal Correspondence Network." In: *Neural Information Processing Systems (NIPS)*. 2016 (cited on p. 64).

[23] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2016 (cited on p. 25).

[24] Felipe Codevilla, Antonio M. López, Vladlen Koltun, and Alexey Dosovitskiy. "On Offline Evaluation of Vision-based Driving Models." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018 (cited on pp. 48, 53).

[25] Felipe Codevilla, Matthias Müller, Alexey Dosovitskiy, Antonio López, and Vladlen Koltun. "End-to-end Driving via Conditional Imitation Learning." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2018 (cited on p. 48).

[26] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. "The Cityscapes Dataset for Semantic Urban Scene Understanding." In: *Proceedings of the IEEE*

Conference on Computer Vision and Pattern Recognition (CVPR). 2016 (cited on pp. 34, 47, 48, 80, 82).

[27] Mark Cummins and Paul Newman. "FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance." In: *International Journal of Robotics Research (IJRR)* 27.6 (2008), pp. 647–665 (cited on pp. 65, 92).

[28] Jan Czarnowski, Stefan Leutenegger, and Andrew Davison. "Semantic Texture for Robust Dense Tracking." In: *Proceedings of the International Conference on Computer Vision Workshops (ICCVW)*. 2017 (cited on pp. 65, 68).

[29] Andrew Davison, Ian Reid, Nicholas Molton, and Olivier Stasse. "MonoSLAM: Real-Time Single Camera SLAM." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 29.6 (2007), pp. 1052–1067 (cited on pp. 64, 92, 93).

[30] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. "SuperPoint: Self-Supervised Interest Point Detection and Description." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2018 (cited on pp. 73, 94, 101–103).

[31] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio López, and Vladlen Koltun. "CARLA: An Open Urban Driving Simulator." In: *Conference on Robot Learning (CoRL)*. 2017 (cited on pp. 34, 48, 52, 64, 93, 100).

[32] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization." In: *Journal of Machine Learning Research (JMLR)* 12.7 (2011), pp. 2121–2159 (cited on p. 23).

[33] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. "D2-Net: A Trainable CNN for Joint Detection and Description of Local Features." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019 (cited on pp. 73, 80, 86, 94, 101–103).

[34] Jakob Engel, Vladlen Koltun, and Daniel Cremers. "Direct Sparse Odometry." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 40.3 (2017), pp. 611–625 (cited on pp. 5, 62, 64, 68, 70, 80, 92–96, 101).

[35] Jakob Engel, Thomas Schöps, and Daniel Cremers. "LSD-SLAM: Large-Scale Direct Monocular SLAM." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2014 (cited on pp. 62, 64, 80, 92, 93).

[36] Jakob Engel, Jörg Stückler, and Daniel Cremers. "Large-Scale Direct SLAM with Stereo Cameras." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2015 (cited on p. 80).

[37] Jakob Engel, Vladyslav Usenko, and Daniel Cremers. "A Photometrically Calibrated Benchmark For Monocular Visual Odometry." In: *arXiv preprint arXiv:1607.02555*. 2016 (cited on pp. 63, 70, 82, 88).

[38] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. "Virtual Worlds as Proxy for Multi-Object Tracking Analysis." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 (cited on p. 47).

[39] Dorian Gálvez-López and Juan Domingo Tardós. "Bags of Binary Words for Fast Place Recognition in Image Sequences." In: *IEEE Transactions on Robotics (T-RO)* 28.5 (2012), pp. 1188–1197 (cited on p. 89).

[40] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. "Vision meets Robotics: The KITTI Dataset." In: *International Journal of Robotics Research (IJRR)* 32.11 (2013), pp. 1231–1237 (cited on pp. 80–82, 86, 90).

[41] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012 (cited on pp. 6, 47, 48, 88).

[42] Ruben Gomez-Ojeda, Zichao Zhang, Javier Gonzalez-Jimenez, and Davide Scaramuzza. "Learning-based Image Enhancement for Visual Odometry in Challenging HDR Environments." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2018 (cited on p. 64).

[43] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016 (cited on p. 18).

[44] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative Adversarial Nets." In: *Neural Information Processing Systems (NIPS)*. 2014 (cited on p. 21).

[45] Albert Gordo, Jon Almazán, Jerome Revaud, and Diane Larlus. "Deep Image Retrieval: Learning Global Representations for Image Search." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2016 (cited on p. 89).

[46] Lei Han, Mengqi Ji, Lu Fang, and Matthias Nießner. "RegNet: Learning the Optimization of Direct Image-to-Image Pose Registration." In: *arXiv preprint arXiv:1812.10212*. 2018 (cited on p. 65).

[47] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 (cited on p. 89).

[48] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. "Distilling the Knowledge in a Neural Network." In: *arXiv preprint arXiv:1503.02531*. 2015 (cited on p. 48).

[49] Geoffrey E. Hinton and Ruslan R. Salakhutdinov. "Reducing the Dimensionality of Data with Neural Networks." In: *Science* 313.5786 (2006), pp. 504–507 (cited on p. 21).

[50] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A. Efros, and Trevor Darrell. "CyCADA: Cycle-Consistent Adversarial Domain Adaptation." In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2018 (cited on p. 33).

[51] Hao Hu and Gerard de Haan. "Low Cost Robust Blur Estimator." In: *Proceedings of the IEEE International Conference on Image Processing (ICIP)*. 2006 (cited on p. 84).

[52] Xinyu Huang, Xinjing Cheng, Qichuan Geng, Binbin Cao, Dingfu Zhou, Peng Wang, Yuanqing Lin, and Ruigang Yang. "The ApolloScape Dataset for Autonomous Driving." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2018 (cited on p. 82).

[53] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. "Multimodal Unsupervised Image-to-Image Translation." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018 (cited on pp. 48, 52).

[54] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. "Image-to-Image Translation with Conditional Adversarial Networks." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 (cited on pp. 39, 62).

[55] Joel Janai, Fatma Güney, Aseem Behl, and Andreas Geiger. "Computer Vision for Autonomous Vehicles: Problems, Datasets and State of the Art." In: *arXiv preprint arXiv:1704.05519*. 2017 (cited on p. 47).

[56] Carlos Jaramillo, Yuichi Taguchi, and Chen Feng. "Direct Multichannel Tracking." In: *Proceedings of the International Conference on 3D Vision (3DV)*. 2017 (cited on pp. 65, 80).

[57] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. "Aggregating local descriptors into a compact image representation." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2010 (cited on p. 89).

[58] Eunah Jung, Nan Yang, and Daniel Cremers. "Multi-Frame GAN: Image Enhancement for Stereo Visual Odometry in Low Light." In: *Conference on Robot Learning (CoRL)*. 2019 (cited on p. 80).

[59] Juho Kannala and Sami S. Brandt. "A Generic Camera Model and Calibration Method for Conventional, Wide-Angle, and Fish-Eye Lenses." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 28.8 (2006), pp. 1335–1340 (cited on p. 84).

[60] Ioannis Kapsouras and Nikos Nikolaidis. "A Vector of Locally Aggregated Descriptors Framework for Action Recognition on Motion Capture Data." In: *Proceedings of the European Signal Processing Conference (EUSIPCO)*. 2018 (cited on p. 65).

[61] Levent Karacan, Zeynep Akata, Aykut Erdem, and Erkut Erdem. "Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts." In: *arXiv preprint arXiv:1612.00215*. 2016 (cited on p. 33).

[62] Elia Kaufmann, Antonio Loquercio, Rene Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. "Deep Drone Racing: Learning Agile Flight in Dynamic Environments." In: *Conference on Robot Learning (CoRL)*. 2018 (cited on p. 46).

[63] Alex Kendall, Matthew Grimes, and Roberto Cipolla. "PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization." In: *Proceedings of the International Conference on Computer Vision (ICCV)*. 2015 (cited on p. 70).

[64] Christian Kerl, Jürgen Sturm, and Daniel Cremers. "Dense visual SLAM for RGB-D cameras." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2013 (cited on pp. 5, 92, 93).

[65] Diederik P. Kingma and Jimmy Lei Ba. "Adam: A Method for Stochastic Optimization." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2015 (cited on p. 23).

[66] Georg Klein and David Murray. "Parallel Tracking and Mapping for Small AR Workspaces." In: *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*. 2007 (cited on pp. 62, 64, 92, 93).

[67] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." In: *Neural Information Processing Systems (NIPS)*. 2012 (cited on pp. 25, 89).

[68] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. "g2o: A General Framework for Graph Optimization." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2011 (cited on p. 85).

[69] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. "Autoencoding beyond pixels using a learned similarity metric." In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2016 (cited on p. 35).

[70] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." In: *Nature* 521.7553 (2015), pp. 436–444 (cited on p. 94).

[71] Yann LeCun, Urs Muller, Jan Ben, Eric Cosatto, and Beat Flepp. "Off-Road Obstacle Avoidance through End-to-End Learning." In: *Neural Information Processing Systems (NIPS)*. 2005 (cited on p. 33).

[72] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. "Keyframe-Based Visual-Inertial SLAM Using Nonlinear Optimization." In: *International Journal of Robotics Research (IJRR)* 34.3 (2015), pp. 314–334 (cited on p. 5).

[73] Kenneth Levenberg. "A Method for the Solution of Certain Non-Linear Problems in Least Squares." In: *Quarterly of Applied Mathematics* 2.2 (1944), pp. 164–168 (cited on pp. 17, 93, 95).

[74] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. "End-to-End Training of Deep Visuomotor Policies." In: *Journal of Machine Learning Research (JMLR)* 17.39 (2016), pp. 1–40 (cited on p. 46).

[75] Minjun Li, Haozhi Huang, Lin Ma, Wei Liu, Tong Zhang, and Yu-Gang Jiang. "Unsupervised Image-to-Image Translation with Stacked Cycle-Consistent Adversarial Networks." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018 (cited on pp. 48, 52).

[76] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. "Unsupervised Image-to-Image Translation Networks." In: *Neural Information Processing Systems (NIPS)*. 2017 (cited on pp. 48, 62).

[77] David G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoint." In: *International Journal of Computer Vision (IJCV)* 60.2 (2004), pp. 91–110 (cited on p. 94).

[78] Stephanie Lowry, Niko Sünderhauf, Paul Newman, John J. Leonard, David Cox, Peter Corke, and Michael J. Milford. "Visual place recognition: A survey." In: *IEEE Transactions on Robotics (T-RO)* 32.1 (2015), pp. 1–19 (cited on p. 88).

[79] Bruce D. Lucas and Takeo Kanade. “An Iterative Image Registration Technique with an Application to Stereo Vision.” In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 1981 (cited on p. 68).

[80] Zhaoyang Lv, Frank Dellaert, James Rehg, and Andreas Geiger. “Taking a Deeper Look at the Inverse Compositional Algorithm.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019 (cited on pp. 65, 73, 94).

[81] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. “Rectifier Nonlinearities Improve Neural Network Acoustic Models.” In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2013 (cited on p. 25).

[82] Will Maddern, Geoffrey Pascoe, Chris Linegar, and Paul Newman. “1 year, 1000 km: The Oxford RobotCar Dataset.” In: *International Journal of Robotics Research (IJRR)* 36.1 (2017), pp. 3–15 (cited on pp. 6, 64, 82, 86, 90, 93, 100).

[83] Donald W. Marquardt. “An Algorithm for Least-Squares Estimation of Nonlinear Parameters.” In: *Journal of the Society for Industrial and Applied Mathematics* 11.2 (1963), pp. 431–441 (cited on pp. 17, 93, 96).

[84] Iaroslav Melekhov, Aleksei Tiulpin, Torsten Sattler, Marc Pollefeys, Esa Rahtu, and Juho Kannala. “DGC-Net: Dense Geometric Correspondence Network.” In: *Proceedings of the IEEE Workshop on Applications of Computer Vision (WACV)*. 2019 (cited on p. 99).

[85] Matthias Müller, Vincent Casser, Jean Lahoud, Neil Smith, and Bernard Ghanem. “Sim4CV: A Photo-Realistic Simulator for Computer Vision Applications.” In: *International Journal of Computer Vision (IJCV)* 126.9 (2018), pp. 902–919 (cited on p. 48).

[86] Matthias Müller, Alexey Dosovitskiy, Bernard Ghanem, and Vladlen Koltun. “Driving Policy Transfer via Modularity and Abstraction.” In: *Conference on Robot Learning (CoRL)*. 2018 (cited on pp. 47, 48).

[87] Raúl Mur-Artal, José María Martínez Montiel, and Juan Domingo Tardós. “ORB-SLAM: A Versatile and Accurate Monocular SLAM System.” In: *IEEE Transactions on Robotics (T-RO)* 31.5 (2015), pp. 1147–1163 (cited on pp. 5, 62, 64, 80, 92, 93).

[88] Raúl Mur-Artal and Juan Domingo Tardós. “ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras.” In: *IEEE Transactions on Robotics (T-RO)* 33.5 (2017), pp. 1255–1262 (cited on pp. 65, 73, 80, 92, 93).

[89] Ashvin Nair, Dian Chen, Pulkit Agrawal, Phillip Isola, Pieter Abbeel, Jitendra Malik, and Sergey Levine. “Combining Self-Supervised Learning and Imitation for Vision-Based Rope Manipulation.” In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2017 (cited on p. 46).

[90] Vinod Nair and Geoffrey E. Hinton. “Rectified Linear Units Improve Restricted Boltzmann Machines.” In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2010 (cited on p. 25).

[91] Richard Newcombe, Steven Lovegrove, and Andrew Davison. “DTAM: Dense Tracking and Mapping in Real-Time.” In: *Proceedings of the International Conference on Computer Vision (ICCV)*. 2011 (cited on pp. 5, 64, 80).

[92] Teddy Ort, Liam Paull, and Daniela Rus. “Autonomous Vehicle Navigation in Rural Environments Without Detailed Prior Maps.” In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2018 (cited on p. 92).

[93] Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. “A Survey of Motion Planning and Control Techniques for Self-driving Urban Vehicles.” In: *IEEE Transactions on Intelligent Vehicles (T-IV)* 1.1 (2016), pp. 33–55 (cited on p. 4).

[94] Seonwook Park, Thomas Schöps, and Marc Pollefeys. “Illumination Change Robustness in Direct Visual SLAM.” In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2017 (cited on p. 64).

[95] Geoffrey Pascoe, Will Maddern, Michael Tanner, Pedro Piniés, and Paul Newman. “NID-SLAM: Robust Monocular SLAM Using Normalised Information Distance.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 (cited on pp. 47, 94).

[96] Matthew Pitropov, Danson Evan Garcia, Jason Rebello, Michael Smart, Carlos Wang, Krzysztof Czarnecki, and Steven Waslander. “Canadian Adverse Driving Conditions Dataset.” In: *International Journal of Robotics Research (IJRR)* 40.4-5 (2021), pp. 681–690 (cited on p. 6).

[97] Dean A. Pomerleau. “ALVINN: An Autonomous Land Vehicle in a Neural Network.” In: *Neural Information Processing Systems (NIPS)*. 1988 (cited on pp. 4, 33, 48).

[98] Horia Porav, Will Maddern, and Paul Newman. “Adversarial Training for Adverse Conditions: Robust Metric Localisation using Appearance Transfer.” In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2018 (cited on p. 62).

[99] Filip Radenović, Giorgos Tolias, and Ondřej Chum. “Fine-tuning CNN Image Retrieval with No Human Annotation.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 41.7 (2018), pp. 1655–1668 (cited on p. 89).

[100] Joern Rehder, Janosch Nikolic, Thomas Schneider, Timo Hinzmann, and Roland Siegwart. “Extending kalibr: Calibrating the Extrinsics of Multiple IMUs and of Individual Axes.” In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2016 (cited on p. 84).

[101] Jerome Revaud, Philippe Weinzaepfel, César Roberto de Souza, and Martin Humenberger. “R2D2: Repeatable and Reliable Detector and Descriptor.” In: *Neural Information Processing Systems (NeurIPS)*. 2019 (cited on pp. 86, 94, 101–103).

[102] Stephan R. Richter, Zeeshan Hayder, and Vladlen Koltun. “Playing for Benchmarks.” In: *Proceedings of the International Conference on Computer Vision (ICCV)*. 2017 (cited on p. 47).

[103] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. “Playing for Data: Ground Truth from Computer Games.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2016 (cited on p. 47).

[104] Ignacio Rocco, Relja Arandjelović, and Josef Sivic. “Convolutional neural network architecture for geometric matching.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 (cited on pp. 93, 94, 99).

[105] Ignacio Rocco, Mircea Cimpoi, Relja Arandjelović, Akihiko Torii, Tomas Pajdla, and Josef Sivic. “Neighbourhood Consensus Networks.” In: *Neural Information Processing Systems (NeurIPS)*. 2018 (cited on p. 64).

[106] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation.” In: *Proceedings of the International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*. 2015 (cited on p. 66).

[107] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M. Lopez. “The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 (cited on p. 47).

[108] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. “ORB: An efficient alternative to SIFT or SURF.” In: *Proceedings of the International Conference on Computer Vision (ICCV)*. 2011 (cited on p. 94).

[109] Sajad Saeedi et al. “Navigating the Landscape for Real-Time Localization and Mapping for Robotics and Virtual and Augmented Reality.” In: *Proceedings of the IEEE* 106.11 (2018), pp. 2020–2039 (cited on p. 92).

[110] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. “From Coarse to Fine: Robust Hierarchical Localization at Large Scale.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019 (cited on p. 94).

[111] Paul-Edouard Sarlin, Frédéric Debraine, Marcin Dymczyk, Roland Siegwart, and Cesar Cadena. “Leveraging Deep Visual Descriptors for Hierarchical Efficient Localization.” In: *Conference on Robot Learning (CoRL)*. 2018 (cited on p. 94).

[112] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. “SuperGlue: Learning Feature Matching with Graph Neural Networks.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020 (cited on pp. 94, 101–103).

[113] Paul-Edouard Sarlin, Mihai Dusmanu, Johannes L Schönberger, Pablo Speciale, Lukas Gruber, Viktor Larsson, Ondrej Miksik, and Marc Pollefeys. “LaMAR: Benchmarking Localization and Mapping for Augmented Reality.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2022 (cited on p. 6).

[114] Torsten Sattler, Tobias Weyand, Bastian Leibe, and Leif Kobbelt. “Image Retrieval for Image-Based Localization Revisited.” In: *Proceedings of the British Machine Vision Conference (BMVC)*. 2012 (cited on p. 89).

[115] Torsten Sattler et al. “Benchmarking 6DOF Outdoor Visual Localization in Changing Conditions.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018 (cited on pp. 6, 63, 65, 70, 82, 87, 90, 94).

[116] Tanner Schmidt, Richard Newcombe, and Dieter Fox. “Self-Supervised Visual Descriptor Learning for Dense Correspondence.” In: *IEEE Robotics and Automation Letters (RA-L)* 2.2 (2017), pp. 420–427 (cited on p. 64).

[117] Johannes Lutz Schönberger and Jan-Michael Frahm. “Structure-from-Motion Revisited.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 (cited on p. 102).

[118] David Schubert, Thore Goll, Nikolaus Demmel, Vladyslav Usenko, Jörg Stückler, and Daniel Cremers. “The TUM VI Benchmark for Evaluating Visual-Inertial Odometry.” In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018 (cited on p. 82).

[119] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. "AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles." In: *Field and Service Robotics (FSR)*. 2017 (cited on p. 48).

[120] Jonathan Shen, Noranart Vesdapunt, Vishnu N. Boddeti, and Kris M. Kitani. "In Teacher We Trust: Learning Compressed Models for Pedestrian Detection." In: *arXiv preprint arXiv:1612.00478*. 2016 (cited on p. 48).

[121] David Silver, J. Andrew Bagnell, and Anthony Stentz. "Learning from Demonstration for Autonomous Navigation in Complex Unstructured Terrain." In: *International Journal of Robotics Research (IJRR)* 29.12 (2010), pp. 1565–1592 (cited on p. 33).

[122] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2015 (cited on p. 89).

[123] Noah Snavely, Steven M. Seitz, and Richard Szeliski. "Photo tourism: Exploring photo collections in 3D." In: *Proceedings of SIGGRAPH*. 2006 (cited on p. 65).

[124] Jaime Spencer, Richard Bowden, and Simon Hadfield. "Same Features, Different Day: Weakly Supervised Feature Learning for Seasonal Invariance." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020 (cited on pp. 86, 87).

[125] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." In: *Journal of Machine Learning Research (JMLR)* 15.1 (2014), pp. 1929–1958 (cited on p. 22).

[126] Lukas von Stumberg, Vladyslav Usenko, and Daniel Cremers. "Direct Sparse Visual-Inertial Odometry using Dynamic Marginalization." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2018 (cited on p. 84).

[127] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. "A Benchmark for the Evaluation of RGB-D SLAM Systems." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2012 (cited on pp. 82, 88).

[128] Xun Sun, Yuanfan Xie, Pei Luo, and Liang Wang. "A Dataset for Benchmarking Image-based Localization." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 (cited on p. 6).

[129] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. "On the importance of initialization and momentum in deep learning." In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2013 (cited on p. 23).

[130] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going Deeper with Convolutions." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015 (cited on pp. 49, 89).

[131] Hajime Taira, Masatoshi Okutomi, Torsten Sattler, Mircea Cimpoi, Marc Pollefeys, Josef Sivic, Tomas Pajdla, and Akihiko Torii. "InLoc: Indoor Visual Localization with Dense Matching and View Synthesis." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018 (cited on pp. 6, 94).

[132] Chengzhou Tang and Ping Tan. "BA-Net: Dense Bundle Adjustment Network." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2019 (cited on p. 65).

[133] Sebastian Thrun et al. "Stanley : The Robot that Won the DARPA Grand Challenge." In: *Journal of Field Robotics* 23.9 (2006), pp. 661–692 (cited on p. 48).

[134] Tijmen Tieleman, Geoffrey Hinton, et al. "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude." In: *COURSERA: Neural networks for machine learning* 4.2 (2012), pp. 26–31 (cited on p. 23).

[135] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. "Particular object retrieval with integral max-pooling of CNN activations." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2016 (cited on p. 89).

[136] Rui Wang, Martin Schwörer, and Daniel Cremers. "Stereo DSO: Large-Scale Direct Sparse Visual Odometry with Stereo Cameras." In: *Proceedings of the International Conference on Computer Vision (ICCV)*. 2017 (cited on pp. 73, 80, 84, 101, 105).

[137] Shenlong Wang, Min Bai, Gellert Mattyus, Hang Chu, Wenjie Luo, Bin Yang, Justin Liang, Joel Cheverie, Sanja Fidler, and Raquel Urtasun. "TorontoCity: Seeing the World With a Million Eyes." In: *Proceedings of the International Conference on Computer Vision (ICCV)*. 2017 (cited on p. 82).

[138] Paul Wohlhart and Vincent Lepetit. "Learning Descriptors for Object Recognition and 3D Pose Estimation." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015 (cited on p. 64).

[139] Huazhe Xu, Yang Gao, Fisher Yu, and Trevor Darrell. "End-to-end Learning of Driving Models from Large-scale Video Datasets." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 (cited on pp. 47, 53).

[140] Chenglin Yang, Lingxi Xie, Siyuan Qiao, and Alan L. Yuille. "Training Deep Neural Networks in Generations: A More Tolerant Teacher Educates Better Students." In: *Proceedings of the National Conference on Artificial Intelligence (AAAI)*. 2019 (cited on p. 48).

[141] Nan Yang, Lukas von Stumberg, Rui Wang, and Daniel Cremers. "D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020 (cited on p. 94).

[142] Nan Yang, Rui Wang, Xiang Gao, and Daniel Cremers. "Challenges in Monocular Visual Odometry: Photometric Calibration, Motion Bias, and Rolling Shutter Effect." In: *IEEE Robotics and Automation Letters (RA-L)* 3.4 (2018), pp. 2878–2885 (cited on pp. 92, 94).

[143] Nan Yang, Rui Wang, Jörg Stückler, and Daniel Cremers. "Deep Virtual Stereo Odometry: Leveraging Deep Depth Prediction for Monocular Direct Sparse Odometry." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018 (cited on p. 80).

[144] Yurong You, Xinlei Pan, Ziyan Wang, and Cewu Lu. "Virtual to Real Reinforcement Learning for Autonomous Driving." In: *Proceedings of the British Machine Vision Conference (BMVC)*. 2017 (cited on pp. 33, 47, 48).

[145] Matthew D Zeiler. "ADADELTA: An Adaptive Learning Rate Method." In: *arXiv preprint arXiv:1212.5701*. 2012 (cited on p. 23).

[146] Feihu Zhang, Victor Prisacariu, Ruigang Yang, and Philip H. S. Torr. "GA-Net: Guided Aggregation Net for End-to-end Stereo Matching." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019 (cited on p. 100).

[147] Jiakai Zhang and Kyunghyun Cho. "Query-Efficient Imitation Learning for End-to-End Autonomous Driving." In: *arXiv preprint arXiv:1605.06450*. 2016 (cited on p. 33).

[148] Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Ken Goldberg, and Pieter Abbeel. "Deep Imitation Learning for Complex Manipulation Tasks from Virtual Reality Teleoperation." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2018 (cited on p. 46).

[149] Xing Zheng, Zack Moratto, Mingyang Li, and Anastasios I. Mourikis. "Photometric Patch-based Visual-Inertial Odometry." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2017 (cited on p. 92).

[150] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. "Open3D: A Modern Library for 3D Data Processing." In: *arXiv preprint arXiv:1801.09847*. 2018 (cited on p. 72).

[151] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks." In: *Proceedings of the International Conference on Computer Vision (ICCV)*. 2017 (cited on pp. 39, 48, 52, 54).