Learning Rich Representations for Robot State Estimation

by

Ioan Andrei Bârsan

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Department of Computer Science
University of Toronto

Abstract

Learning Rich Representations for Robot State Estimation

Ioan Andrei Bârsan
Doctor of Philosophy
Department of Computer Science
University of Toronto
2024

Mobile robots, such as self-driving vehicles, need to operate safely and autonomously in a wide range of environments. This requires an autonomy software stack with robustness built into it at every level. At the root of every decision that a robot makes is its ego-state estimation: establishing where the robot is, how its actuators are positioned, and the related dynamics.

As the scale and complexity of real-world robotic systems continue to grow, many research questions remain open: How do we design state estimation systems that are performant, interpretable, and scalable? How sensitive are complex robotic systems to state estimation failures? Can we mirror recent trends in the broader field of deep learning and improve robot localization by learning on large datasets?

This thesis encompasses a line of work which analyzes these questions by studying the robustness of robotic systems at the state estimation level through the lens of robot localization. In the first part of the thesis, we propose a way to formulate localization as an online histogram filtering problem and study the importance of learning task-specific representations and data-driven compression for interpretability and scalability, respectively. In the second part of the thesis, we study state estimation from a systems perspective by analyzing the impact of localization failures on downstream tasks and propose a perception architecture that improves resilience to this family of errors. We continue our analysis by focusing on the impact of large datasets on localization and present Pit30M, a new large-scale localization dataset that helps us draw novel insights into global localization thanks to its highly accurate ground truth.

The approaches discussed in this thesis provide insights into how to build accurate and resilient state estimation systems, how to evaluate them holistically in terms of their impact on overall system performance, and how to design localization and mapping systems that scale effortlessly to nation-sized maps.

Acknowledgements

The journey I undertook during my PhD was challenging yet highly rewarding, and I am deeply grateful to all who supported me throughout.

Of course, I would not be writing this thesis if my advisor, Raquel Urtasun, hadn't had the faith to take me in as a graduate student back in 2017. Her supportive nature, coupled with her unbridled, contagious excitement for research, made the difficult decision to move to a new continent much easier.

I am also thankful to Sanja Fidler and David J Fleet, my other committee members, for the numerous fascinating and inspiring conversations. Sanja's questions always encouraged me to stay up to date with the latest advances in computer vision. David always encouraged me to look at things from new perspectives I had never considered before. Your feedback and support have helped shape the way I look at research and the kinds of questions I ask. I would also like to thank Professors Florian Shkurti and Luca Carlone for serving as external examiners for my PhD, providing me with insightful, actionable feedback, and Professor Marsha Chechik for being a kind, encouraging, and supportive mentor. Candyce, your help has been invaluable in turning all the key milestones of this PhD into concrete events on everyone's calendars; thank you!

I would like to thank Shenlong Wang, in particular, for being my friend and for mentoring me during my first few years at the University of Toronto and Uber ATG. Shenlong has always encouraged and supported me to become an independent scientist, and he has done a fantastic job making me feel welcome in Toronto and our research lab. I will never forget the first time we met during the Fall of 2017 in the Sandford Fleming building and how exciting and fun it was to talk to you about research over lunch as we went to grab some delicious sandwiches at a place near College & Spadina, which has since shut down. I am likewise grateful for all the friendships I built along the way with fellow students and staff alike—Wei-Chiu Ma, Namdar Homayounfar, Jingkang Wang, Joyce Yang, Ze Yang, Yun Chen, Gellért Máttyus, Kelvin Wong, Jason Siefken, Ignacio Tartavull, Luisa San Martin, Davi Frossard, Siva Manivasagam, Min Bai, Inmar Givoni, Ali Athar, Sergio Casas, Katie Luo, Dominic Cheng, Justin Liang, Arnaud Bonnet, Elaine Papa, Renjie Liao, Mengye Ren, Wenjie Luo, Alex Cui, Sean Segal, Thomas Li, Yuwen Xiong, Bin Yang, Wenyuan Zeng, Chris Zhang, James Tu, Frieda Rong, Ben Argo, Jack Fan, Jashan Shewakramani, Xinkai Wei, Sasha Doubov, John Phillips, Can Cui, George Chen, Andrei Pokrovsky, Simon Suo, Annie Zhang, Abbas Sadat, and Quinlan Sykora. I feel blessed to have worked with so many interesting, brilliant people; you are all truly wonderful.

I also cherish the long-lasting friendships I built over the years at ETH Zurich: Taivo Pungas, who taught me about meta-cognition and who motivated me to learn more about how to turn ML into products, Tom Sydney Kerckhove for helping me understand the importance of attention to detail and for helping me not be satisfied with shallow explanations, and Bernhard Kratzwald for his lovely sense of humor and his unrivaled ability to have fun even while working on the most technologically demanding machine learning projects. These friendships have been instrumental in my development as a scientist and as a person. Thank you, likewise, to my amazing Romanian friends, Dragoș, Teo, Alex B., Vlad G., Cristina, Rareș, Cristi, Ionuț, Maria, Paul, Vlad Ț., Alex I., Tudor, Horia, and Codrin, for being by my side all these years and for always making me feel excited to come back and visit Romania!

I am also deeply grateful to Cătălin Tufănaru, who encouraged me to keep pushing when I felt like giving up and who encouraged me to seek and to embody *phronesis*.

Of course, thank you, Julieta, for being by my side for years throughout the process, supporting me day and night! Not a day goes by when I am not grateful for having your lovely sense of humor and amazingly inspiring brains by my side. Last but most certainly not least, I wish to thank my parents, Anca and Lucian,

for being incredible engineering role models and for being patient with me before I could develop my own patience. This thesis is dedicated to them.

Most of the work in this thesis was performed on or in the vicinity of the University of Toronto campus in downtown Toronto. I wish to acknowledge that the land on which the University operates has been the traditional land of the Huron-Wendat, the Seneca, and most recently, the Mississaugas of the Credit River. Today, this meeting place is still the home to many Indigenous people from across Turtle Island and I am grateful to have the opportunity to work on this land.

Contents

List of Tables

List of Figures

1 Introduction

The field of robotics has been undergoing rapid evolution over the last half-century, and the rate of change is only accelerating due in no small part to the massive advances in computer vision we have seen over the last decade.

Powered primarily by explosive progress in machine learning (ML), the rapid advancement of computer vision has brought about sizeable leaps in performance on tasks ranging from image understanding and 3D reconstruction to object detection and motion forecasting. Many of these advances apply to robotics.

Nevertheless, going from a clearly defined computer vision task to a real-world application is non-trivial. How do we transform self-contained tasks like image classification into useful applications? How can semantic segmentation help people with impaired vision? How can we build a better indoor navigation experience using visual odometry? These open questions fall outside the scope of typical computer vision literature.

Robotics is even more challenging because robots typically need to interact with the world physically. In this sense, we can differentiate robotics from other applications of computer vision and machine learning by speaking of *embodiment*. Robotic intelligence is embodied: It interacts with a physical reality directly, while a computer vision service running in the cloud does not. Autonomous mobile robots like drones or self-driving cars are a particular case which is yet more difficult, as such robots require added layers of robustness and safety. Moravec's Paradox[1] comes into play in full force in these applications which need to safely and reliably perform a wide range of tasks which are often trivial to humans but challenging for software: understand their environment, localize within it, plan collision-free trajectories, and execute them successfully in the presence of uncertainty, noise, and modeling errors—all in real-time.

There is tremendous potential in modern machine learning, and its effectiveness is evident in a wide range of applications, from speech

[1] Named after CMU's Prof. Hans Moravec, who first discussed it in his 1988 book "Mind Children," the Paradox states that there is a strong trend for machines to be able to achieve superhuman performance on tasks average humans perceive as difficult (e.g., chess) while struggling with tasks humans perceive as trivial, such as object manipulation and bipedal locomotion (Moravec 1988, 9–11).

recognition (Radford et al. 2023) to playing sophisticated games in partially observable environments (Berner et al. 2019). However, applying it to real-world, open-domain tasks like autonomous robotics requires us to frequently shift our focus to the system rather than to the task level and understand how all algorithmic components interact with each other and how this relates to the end goal of the system in question.

1.1 Current Challenges

In spite of recent advances in machine learning (specifically deep learning), progress in areas like natural language processing has been faster than in autonomous robotics, despite the fact that both fields are equally amenable to improvement through learning.

Robots[2] need the ability to operate continuously in unstructured environments, which, even in the simplest cases, requires the ability to solve many different 3D reasoning problems at once: navigation, localization, perception, and planning (Shenlong Wang 2021). The robot needs to deal with uncertainty and handle failures of individual components gracefully through introspection and redundancy.

[2] Throughout this thesis, whenever we say *robot* we mean *autonomous mobile robot*, which means we avoid limiting ourselves to teleoperated robots or static robots such as manipulator arms.

Additionally, the task of autonomous driving comes with its own set of challenges in addition to typical robotics ones. As autonomous vehicles need to transport humans at high speed through diverse and unstructured environments, resilience and safety become major concerns.

Embodied applications of computer vision, such as autonomous driving, stand to bring massive economic and quality-of-life improvements to the world. Therefore, it is essential to understand and overcome the challenges that prevent the adoption of such transformative technologies. We can group these challenges into several high-level categories.

- **High Performance.** We naturally want systems that excel at their tasks. This means we need the ability to set goals, define metrics which measure how close we are to a goal, and iteratively develop a robotic system to achieve the performance goal.
- **Scalability & Reduced Costs.** We want to achieve our goals at a large scale, for many robots in many environments, while controlling costs. While computational efficiency is an important part of scalability, it is not the only one. Auxiliary data necessary during operations, such as maps, should be represented in formats that lend themselves well to incremental updates and compression, and the pipelines used to train the ML models should be cost-effective and easy to maintain.

- **Interpretability.** Considering the safety requirements of real-world robotics, their actions need to be interpretable to human auditors, both those working on the systems themselves as well as those working in regulatory, insurance, or law enforcement roles. Interpretable methods can help build public trust while also providing insights that can lead to improved performance (Marcu et al. 2023).
- **System resilience.** Fallbacks are needed to recover from failures. The system must identify failure conditions, such as a nonresponsive component, a sensor issue, or invalid model outputs, and react accordingly, such as by deploying a fallback option or, if necessary, triggering a safe shutdown maneuver. Therefore, it is vital to incorporate redundancy and self-healing into the components that make up a safety-critical system[3].
- **Large-Scale Analysis.** In order to demonstrate safety, autonomous systems need to be analyzed and validated under extremely large varieties of conditions that cover the long tail of possible scenarios.

[3] For the scope of this thesis, we limit our attention to failures which manifest themselves as incorrect localization poses, and leave the study of resilience to complete localizer failures (e.g., system crashes) or to completely new operational domains as future work. Please refer to papers like (Reinke et al. 2022; Ebadi et al. 2023) for broader discussions on topics like loose vs. tight coupling, dealing with dust accumulation around sensors, and CPU scheduling of mission-critical tasks.

At the heart of robotic systems lies the task of state estimation, namely understanding where a robot and its actuators are positioned within the operation environment. As we will discuss at length in Chapter 2.3, this task is critical for most downstream components: 3D detection, motion forecasting, routing, and motion planning, which all require some form of map data in order to accomplish their task. For the scope of this thesis, we focus on autonomous driving, where there are no dedicated actuators like arms or manipulators, so the task of state estimation is reduced to localization[4]. As we will elaborate in Chapter 2, most autonomous driving systems rely on prior maps, and knowing their position within these maps is critical for their operation. Because maps can be used in many downstream tasks, localization errors can affect the system in a wide range of ways. This makes localization an interesting problem to study not just by itself but also as part of a broader system.

[4] State estimation for localization does not imply the state only tracks the pose and its dynamics. Typical robot state estimators also keep track of additional quantities, such as the biases of an IMU (Mourikis and Roumeliotis 2007).

Accurately localizing observations within a scene has numerous applications beyond the autonomy software of an SDV. Localization is a requirement in areas as diverse as 3D reconstruction (J. Wang et al. 2022), digital twin creation (Turki et al. 2023), and offline labeling (A. J. Yang et al. 2023; Fan et al. 2023). As elaborated in Chapter 8, simulation is the key to solving many of the problems currently limiting the robustness of robotic systems. It is vital for simulations to use realistic environments that reflect the real ones where the robot is expected to operate. In order to reconstruct such environments from existing observations, it is necessary to aggregate all observations into

a common reference frame, with localization playing an important role in this process.

The goal of this thesis is to propose several approaches toward performant, scalable, interpretable, and resilient robot state estimation while also studying the role of state estimation within the broader robotic system and how learning-based approaches for localization scale as a function of dataset size.

1.2 Key Contributions

This thesis aims to tackle the challenges above within the scope of self-driving vehicles, with a focus on localization. We show that high performance can be achieved with deep LiDAR[5] matching in online localization, and with retrieval in global localization. Scalability can be achieved in a wide range of ways: lightweight semantic maps, a database of low-dimensional vectors, or domain-specific compression. The proposed online localizers operate in a low-dimensional space using histogram filtering, which facilitates interpretability.

[5] Light Detection and Ranging

We also show that LiDAR localization can be integrated as a perception network sub-module, facilitating resilience without requiring a large computational budget. Last but not least, we propose a new dataset called Pit30M to facilitate large-scale analysis of localization techniques. The subsequent chapters are organized as follows.

Chapter 2 provides the background knowledge required for subsequent parts of the thesis, covering notation, a brief primer on recursive Bayesian estimation, and a survey of related work. The related work is grouped in this chapter as the following chapters have significant overlap because they all deal with localization.

Chapter 3 describes a localization method based on lightweight HD[6] maps which leverages powerful deep learning observation models based on camera and LiDAR data, but which is not trained end-to-end. By using lightweight maps, this method scales well to highway-size environments. Thanks to its low-dimensional histogram filter formulation, the method is also interpretable.

[6] High Definition

Chapter 4 builds upon the previous chapter by developing a deep representation for LiDAR-based localization, which is trained end-to-end for the task of localization, and demonstrates improved performance. As in the previous chapter, this representation is integrated into a recursive Bayesian filtering framework. By leveraging 3-DoF cross-correlation, which is end-to-end differentiable, to perform matching, this approach allows the LiDAR representations to be optimized directly for the task of localization, improving performance and efficiency, and maintaining the interpretability of the previous chapter.

While using learned representations can substantially improve the

robustness of state estimation, LiDAR matching still requires dense map imagery to function. This means map storage will take up large amounts of space, limiting scalability. **Chapter 5** addresses this by extending LiDAR-based localization to use a compression-aware map representation, which can outperform general-purpose lossy compression by an order of magnitude while maintaining similar end-task performance. We achieve this by optimizing the compressibility of the map representation, in addition to the task performance.

Chapter 6 moves beyond the task of localization and investigates its role within a full robotics system. We analyze the impact of localization errors on tasks such as motion forecasting and motion planning, and formulate a multi-task solution which allows modules downstream of localization to recover from pose errors. The resulting system maintains high-performance localization and perception while also ensuring that perception is resilient to potential localization errors.

Chapter 7 discusses an open, petabyte-scale dataset and benchmark for localization meant to bring the task of large-scale state estimation into the 2020s, while also enabling countless opportunities for generative modeling and unsupervised learning. We study different approaches to global localization and show that a simple retrieval-based localizer is highly performant and scales to city-scale environments thanks to its ability to leverage a low-dimensional database of geotagged images as a map. We leverage Pit30M to study the behavior of this global localizer at an unprecedented scale.

This chapter also discusses some of the practical considerations of developing and releasing a dataset of this scale, such as handling the anonymization of sensitive information, creating training, validation, and test splits, and designing a cloud-friendly software development kit.

Finally, **Chapter 8** concludes the thesis by summarizing its key insights and discussing limitations and future topics. We summarize which challenges are primarily tackled by which chapter in Tab. 1.1.

Contribution Scope. Most of the contributions in this thesis are studied through the lens of autonomous driving, e.g., assuming large amounts of onboard compute resources for inference and the presence of a LiDAR. However, many insights can be transferred to broader domains. LiDAR matching localization can also be applied to camera-based systems by leveraging known sensor extrinsics (Philion and Fidler 2020) or learned transformations (W. Yang et al. 2021; Z. Li et al. 2022) to map camera images to bird's-eye view. The same applies to the map compression line of work presented in Chapter 5, which is agnostic to the data source used to build a map. Furthermore, the concept of task-specific compression extends beyond localization and

	High Perf.	Scalable	Interpretable	Resilient	Large-Scale Analysis
Lightweight Localization (Chapter 3)		✓	✓		
Deep LiDAR Matching (Chapter 4)	✓		✓		
Learning to Compress (Chapter 5)	✓	✓	✓		
Localization & Perception (Chapter 6)	✓	✓		✓	
The Pit30M Benchmark (Chapter 7)	✓	✓			✓

Table 1.1: **Challenges tackled by each chapter.** Each chapter tackles one or more of the key challenges in autonomous mobile robots, which are highlighted in the intro.

into tasks like memory-augmented perception (You, Luo, Chen, et al. 2022), language modeling (J. Mu, Li, and Goodman 2024), and retrieval-augmented generation (RAG).

The system-level analysis in Chapter 6 is likewise sensor-agnostic and can apply to other approaches, such as pure end-to-end driving using HD maps (Sadat et al. 2020). Multi-task learning is not limited to correcting localization errors and can also tackle, for example, calibration errors that may arise during the operation of a multi-sensor system (Kanai et al. 2023).

The Pit30M benchmark is itself multi-sensor, and the original paper already focused primarily on camera data, making its insights amenable to drones and mobile phones, in addition to ground robots. The dataset also includes camera, GPS, wheel speed, and IMU[7] data. Its long sequences can also be applied to study tasks like unsupervised learning and generalizable novel view synthesis (NVS).

[7] inertial measurement unit

1.3 Relation to Published Work

This thesis encompasses several research papers on which I worked during my time at the University of Toronto. All of them have been published in top-tier international robotics and computer vision venues. The * denotes equal contribution.

- Chapter 3 is based on: Wei-Chiu Ma*, Ignacio Tartavull*, **Ioan Andrei Bârsan***, Shenlong Wang*, Min Bai, Gellért Máttyus, Namdar Homayounfar, Shrinidhi Kowshika Lakshmikanth, Andrei Pokrovsky, and Raquel Urtasun. "Exploiting sparse semantic HD maps for self-driving vehicle localization." *International Conference on Intelligent Robots and Systems* (**IROS**). 2019.

- Chapter 4 is based on: **Ioan Andrei Bârsan***, Shenlong Wang*, Andrei Pokrovsky, and Raquel Urtasun. "Learning to Localize Using a LiDAR Intensity Map." *Conference on Robot Learning* (**CoRL**). 2018.

- Chapter 5 is based on: Xinkai Wei*, **Ioan Andrei Bârsan***, Shenlong Wang*, Julieta Martinez, and Raquel Urtasun. "Learning to localize through compressed binary maps." *International Conference on Computer Vision and Pattern Recognition* (**CVPR**). 2019.

- Chapter 6 is based on: John Phillips*, Julieta Martinez*, **Ioan Andrei Bârsan***, Sergio Casas, Abbas Sadat, and Raquel Urtasun. "Deep Multi-Task Learning for Joint Localization, Perception, and Prediction" *International Conference on Computer Vision and Pattern Recognition* (**CVPR**), 2021.

- Chapter 7 is based on: Julieta Martinez, Sasha Doubov, Jack Fan, **Ioan Andrei Bârsan**, Shenlong Wang, Gellért Máttyus, and Raquel Urtasun, "Pit30m: A benchmark for global localization in the age of self-driving cars", *International Conference on Robots and Systems* (**IROS**), 2020. (**Best Application Paper Finalist**)

  – This chapter also includes new material detailing the process and challenges of releasing a helpful, easy-to-use large-scale dataset: finding a hosting provider to sponsor the dataset, anonymization, and SDK design.

1.4 Other Research During My PhD Study

Throughout my PhD, I have had the privilege of working with many talented collaborators on a broad range of topics whose scope lies beyond this thesis. This section gives a brief overview of this work.

1.4.1 Simultaneous Localization and Mapping

It is often infeasible for all of a robot's sensors to record at the same cadence. In some cases, high sampling rates are a requirement for nominal operation (e.g., IMU), while other sensors, such as low-end GNSS receivers, may only provide measurements at 1Hz. For many sensors, such as most LiDARs and cameras, any notion of an instantaneous or near-instantaneous measurement ceases to exist due to the rolling shutter. It is, therefore, vital for robots to employ designs which are aware of asynchronous sensors. In AMV-SLAM (A. J. Yang et al. 2021), we studied asynchronous sensors from the perspective of multi-camera simultaneous localization and mapping. By modeling the robot trajectory using B-splines, the system was

able to incorporate asynchronous observations coming from multiple cameras in all stages of the SLAM pipeline: tracking, mapping, and place recognition. Additionally, given the lack of publicly available asynchronous multi-camera datasets at the time of publication, we curated our own as a subset of the Pit30M dataset presented in Chapter 7. The dataset is available for download on its project page: `https://www.cs.toronto.edu/~ajyang/amv-slam/`.

1.4.2 Sensor Simulation and Domain Gap Analysis

While the construction of simulated worlds is one of the many applications of mapping and localization, discussing specific approaches for simulation is beyond the scope of this thesis. A simulator, just like a video game, consists of multiple components, including asset creation, sensor simulation ("graphics"), behavior simulation ("NPCs"), and the simulation infrastructure ("the engine"). In CADSim (J. Wang et al. 2022), we focused on automated asset creation, and proposed an approach for part-aware object reconstruction from in-the-wild data. By leveraging mesh deformation, the method was able to produce realistic assets which could be readily imported and rendered into any rasterization-based graphics engine, and used to synthesize both camera and LiDAR sensor data.

More recently, in LidarDG (Manivasagam et al. 2023), we explored the topic of LiDAR simulation realism in more detail and proposed a framework for ablating the impact of asset quality and various sensor effects such as asset quality, rolling shutter, unreturned rays, and multi-echo returns, on realism.

1.5 About This Thesis

I typeset this thesis using a template based on Edward R. Tufte's books. Professor Tufte is well-known for his pioneering research and writing on statistics and data visualization. He has written several books on the intersection of these two fields, such as The Visual Display of Quantitative Information (Tufte 2001). Prof. Tufte advocates for layouts that facilitate focusing on the main flow of ideas in the main text, making ample use of side notes for additional but non-critical details.

I found the format makes reading long technical texts easier and more pleasant, especially in digital formats, so I used it in my thesis by leveraging the MIT-licensed `tufte-markdown` project available on GitHub at github.com/duzyn/tufte-markdown.

2
Background and Related Work

This chapter presents the key mathematical building blocks required to understand the thesis, together with a survey of related work in robot localization and related tasks. The background assumes familiarity with basic probability and graphical models used in robotics. For readers who are not yet familiar with these concepts or those interested in a refresher, textbooks such as (Thrun, Burgard, and Fox 2005), (Siegwart, Nourbakhsh, and Scaramuzza 2011), and (Barfoot 2024) provide excellent introductions to key topics ranging from robot and sensor basics, all the way to sophisticated mathematical techniques for localization, 3D registration, SLAM, and more.

The chapter is structured as follows. We begin by introducing the fundamental notation and key mathematical concepts used throughout the thesis. We also introduce the key components that make up the majority of autonomous driving systems and the fundamentals of LiDAR, the sensor used in most applications from this thesis. Then, in Chapter 2.4, we discuss the theoretical foundations of online histogram filtering and their advantages in ground robot localization. We apply this well-established framework multiple times throughout this thesis, to the task of localizing in lightweight HD maps (Chapter 3), learning representations for LiDAR localization (Chapter 4), learning compressible maps (Chapter 5), and localizing as part of a multi-task system (Chapter 6).

Afterward, Chapter 2.5 frames the thesis by providing an extensive overview of the prior work in robot localization, as well as briefly covering related research in SLAM, mapping, compression, and multi-task learning.

2.1 3D Geometry Notation

We represent 3D coordinates as 3D column vectors, such as:

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \in \mathbb{R}^3, \tag{2.1}$$

where $x, y, z \in \mathbb{R}$.

We can express a 3D rotation as an orthogonal matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$ of determinant 1, and a 3D translation as a vector $\mathbf{t} \in \mathbb{R}^3$. They can be assembled together to form a rigid transformation $\mathbf{T} \in \mathbb{R}^{4\times4}$ as

$$\mathbf{T} = \begin{pmatrix} \mathbf{R} & \mathbf{x} \\ \mathbf{0}^T & 1 \end{pmatrix} \in \mathbb{R}^{4\times4}. \tag{2.2}$$

This transform has six degrees of freedom. We can also use letters, typically capital letters, to denote sequences of points. For example, $\mathbf{X} \in \mathbb{R}^{4\times N}$ will denote N 4D vectors, stacked horizontally. This is convenient when reasoning about multiple points at once for a 3D map or a LiDAR sweep.

While structurally, the $\mathbf{R}$ and $\mathbf{T}$ are real 3×3 and 4×4 matrices, their unique constraints (orthogonality and determinant $= 1$ for $\mathbf{R}$, and the added structural constraints for $\mathbf{T}$) limit their degrees of freedom (DoF), meaning the matrices are not arbitrary elements of these matrix groups. Instead, these quantities represent members of the Special Orthogonal group of $\mathbb{R}^{3\times3}$, and the Special Euclidean group of $\mathbb{R}^{4\times4}$. These groups are Lie groups, which impacts how their elements are interpreted, e.g., during interpolation and optimization (Gallier and Quaintance 2020).

The rigid transformation can be applied to a point $\mathbf{x}$ by rotating, then translating it, as follows:

$$\mathbf{x}' = \mathbf{R}\mathbf{x} + \mathbf{t}. \tag{2.3}$$

However, it is often more convenient to apply $\mathbf{T}$ directly as a linear operator. For this, we need the notion of homogeneous coordinates.

Homogeneous coordinates. It is beneficial to define an extended notation for 3D vectors such that rigid transformations like those in Eq. 2.2 can be applied directly as linear operators. We denote

$$\hat{\mathbf{x}} = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}, \tag{2.4}$$

as the homogeneous version of $\mathbf{x}$. Homogeneous points have the property that all points of format $\hat{\mathbf{x}} = (kx, ky, kz, k)$ are equivalent for any

$k \neq 0$[1]. In some equations, we may abuse the notations and simply use $\mathbf{x}$ itself to denote a point with 2D or 3D homogeneous coordinates.

We can also spell out the source and destination coordinate frames for a 3D transformation. This can also be done for quantities such as point 3D coordinates and vector directions in order to improve clarity. Specifically, we can use

$$\mathbf{X}_w = {}^w\mathbf{T}_r\mathbf{X}_r \tag{2.5}$$

to denote a 3D operation $\mathbf{T}$ which transforms points $\mathbf{X}_r$ expressed in some robot reference frame r into a world reference frame denoted w.

[1] In practice, when $k = 0$ the point is still valid, but it is a *point at infinity*. This has important ramifications in projective geometry, but a detailed treatment of these geometric entities is beyond the scope of this chapter. Please refer to (Hartley and Zisserman 2003) for further details.

2.2 LiDAR Sensors

In addition to cameras, many ground robots and most autonomous vehicles leverage Light Detection And Ranging (LiDAR) sensors. LiDAR sensor typically leverage time-of-flight measurements for infrared laser pulses to perceive the 3D structure of an environment. For the scope of applications in this thesis, we focus on **spinning 3D LiDAR sensors**, such as the Velodyne HDL-64E.

Spinning LiDAR sensors consist of a column of lasers, 64 in the case of the HDL-64E, which spins at a pre-defined frequency, taking readings of the surrounding environment. At precise azimuth angles along its 360° spin, each laser from the spinning LiDAR's laser column emits a laser pulse. The reflections of these pulses are measured by a series of photodiodes typically rigidly attached close to the emitters. By measuring the time-of-flight for the pulse, each photodiode can infer the range of the object that produced the reflection using the speed of light. The range information, coupled with the known beam pitch and azimuth, can be used to infer the precise 3D coordinates of the perceived object in the sensor frame. We depict a simplified diagram of a spinning LiDAR in Fig. 2.1.

In addition to the range (and, by extension, cartesian coordinates) of points, laser scanners can also return an intensity measurement. While the definition of "intensity'' can vary across the literature and manufacturers (Kashani et al. 2015), it can generally be interpreted as a measure of the reflectivity of the object hit by the laser. Thus, darker objects will yield low-intensity returns, while lighter and highly reflective ones will return very high intensities. Nevertheless, many factors can influence the actual reported intensity of a LiDAR return, including incidence angle and environmental factors such as weather (Kashani et al. 2015; Manivasagam et al. 2023). This can pose a problem for robotic systems that rely on LiDAR intensity measurements for tasks such as localization, especially when different LiDAR units, or even different models are used in mapping versus localization. Chapter 4 explores this topic in detail, proposing an efficient and

effective learning-based method to overcome it.

Unlike cameras, LiDARs are a form of active sensor, which makes them mostly invariant to ambient lighting. They also perceive 3D structure directly, bypassing the ambiguities of doing so with stereo or monocular cameras.

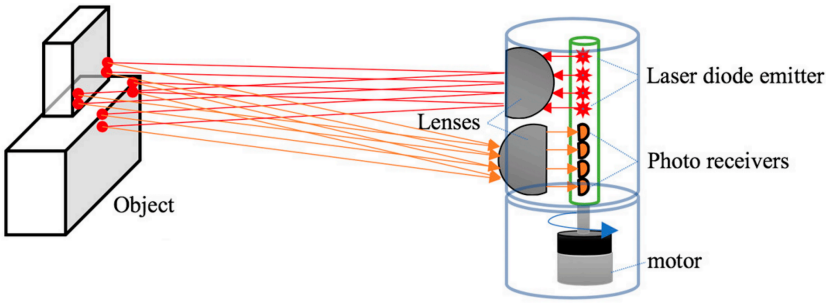


Figure 2.1: **3D LiDAR Operating Principle.** Illustration from (Rosique et al. 2019).

While their cost and complexity are higher than those of typical cameras, the industry has made great strides in the last five years to address them, often in the form of "solid-state'' LiDARs, which can scan an environment without physically moving the entire laser and lens assembly (Luminar Technologies 2024).

2.3 Self-Driving Vehicles

Self-Driving Vehicles (SDVs) represent a particularly challenging application of mobile ground robots. With applications in taxi services and freight transportation, SDVs have the potential to bring about substantial economic benefits and improve overall road safety. Nevertheless, the need to operate safely and autonomously alongside human drivers across a wide range of high-speed environments brings with it a host of challenges.

In this thesis, we focus on autonomous vehicles designed to share the road with human drivers and forego discussing industrial applications like warehouse automation, mining, and agriculture. In this section, we will give a brief overview of the main tasks an average self-driving vehicle must perform for safe, efficient, and effective operation. We depict a high-level diagram of the structure of such a system in Fig. 2.2, noting that in many approaches, some or all of the depicted modules may be fused—e.g., perception and prediction may be formulated as a unified task, prediction and planning may be unified in a reactive framework, or a single neural network may replace all components between the sensors and the controls.

System Inputs. The inputs to a typical autonomous driving system considered in this thesis are (i) its high-level mission, (ii) a map,

and (iii) the sensor data. The high-level mission specifies the task that the robot is assigned, such as the pick-up and drop-off locations for a passenger or cargo, in the case of robotaxis or shipping, respectively.

The map may be a plain navigation map, like those used by humans in GPS navigation, or an HD map. An HD map may encompass multiple layers. Layers may contain sparse information, such as the road centerlines, lane centerlines, traffic signs, and their topology, or they may contain dense information, such as satellite imagery, ground LiDAR intensity imagery, or a digital elevation model. Recent work has also explored learning what to store in the map in order to maximize performance on tasks such as object detection (You, Luo, Chen, et al. 2022), though such representations are still limited to research projects at the time of writing.

Sensor inputs can be very diverse. In the applications covered by this thesis, we consider multiple cameras, a single spinning LiDAR, an IMU, GNSS, and wheel speed sensor data, though many mobile robots also utilize ultrasonic sensors, RADAR, event cameras, or thermal cameras.

In some cases, system inputs may also come from a human. While it is common for a human safety driver to be present behind the wheel in earlier stages of SDV development, remote monitoring is common for mature autonomy systems, typically for safety reasons. Remote assistance inputs typically take the form of direct driving actuation or waypoints (Kettwich, Schrank, and Oehl 2021; Waymo LLC 2024, C.VI.A), though in recent research, natural language has also demonstrated potential in helping autonomous vehicles navigate challenging areas (Mao et al. 2023).

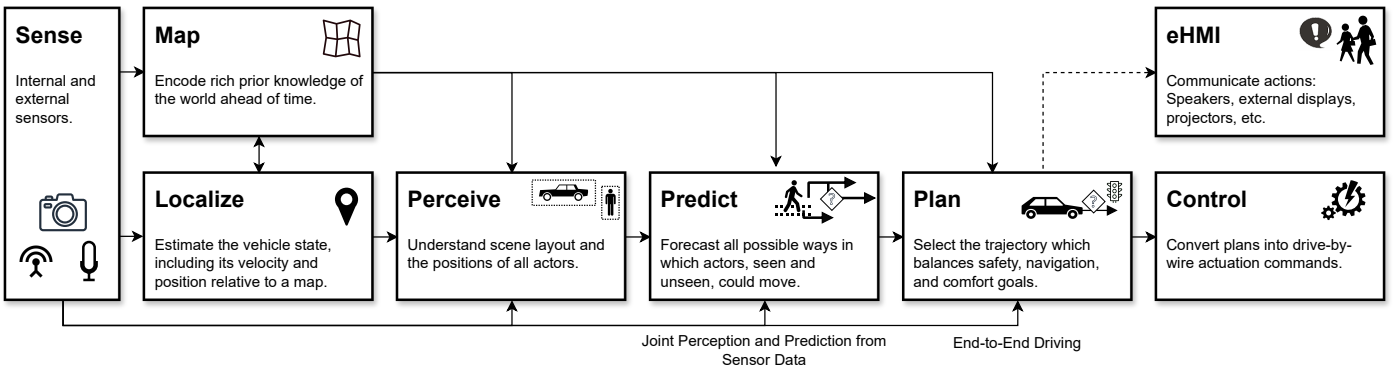


Figure 2.2: **Data flow diagram in typical self-driving software systems.** This figure depicts an approximation of the components which encompass an SDV. *eHMI* stands for external human-machine interface, and encompasses all SDV-specific methods that a robotic driver can use to communicate with other traffic participants, like specific sound alerts, custom lights, displays, etc.

System Outputs. The key outputs of an SDV typically include steering, throttle, and braking commands generated by a controller, which are sent through a drive-by-wire interface to operate the vehicle

(Ort et al. 2019). A high-reliability computer may further validate these commands before they are actuated to ensure they are within a safe pre-validated range.

In multi-agent settings (V2X), an SDV may also send intermediate outputs, such as object detections or even feature maps, to other robots to improve the effective field of view of nearby agents (T.-H. Wang et al. 2020).

Another interesting family of outputs relates to communication with non-autonomous traffic participants. While humans often use hand gestures, eye contact, honking, or even voice to communicate with other traffic participants, autonomous vehicles cannot use most of these channels. Nevertheless, these channels are important in navigating complex scenarios, as well as in improving safety and building public trust. For example, while it can be easy for a pedestrian to tell that a human driver has seen them by making eye contact, this is not possible with autonomous vehicles. A wide body of research at the intersection of robotics and human-computer interaction (HCI) aims to study these kinds of challenges through the development of various external human-machine interfaces (eHMIs). Proposed solutions range from light strips and speakers (Carmona et al. 2021) to installing animatronic eyes on vehicles which are actuated to make eye contact with nearby pedestrians (Chang et al. 2022) and even laser projectors which illuminate the planned trajectory of a vehicle (Daimler AG 2015, refer to the Photos section).

System Modules. While it is possible to develop camera-only autonomy systems, the inability of cameras to sense 3D structure increases complexity and computational costs as these aspects need to be modeled in software, either implicitly or explicitly. Thus, we describe autonomy systems which can leverage LiDAR sensors to directly infer the environment structure. Nevertheless, most of this discussion also applies to camera-only SDVs.

We discuss the system components in a modular way for ease of understanding, but this does not mean each such component operates standalone. For example, it is common to unify perception and prediction into a single task (W. Luo, Yang, and Urtasun 2018; Y. Hu et al. 2023). This can also be extended to motion planning, in which case we speak of *reactive planning* (Rhinehart et al. 2021).

- **Mission Planning** dictates the overarching goal of a mobile robot operating in an environment. For example, for warehouse automation robots, this may be the module which dispatches a robot to move a specific pallet, while for SDVs, this typically takes the form of high-level route planning from point A to point B based on customer input.

- **Sensor Preprocessing** involves decoding and aggregating sensor data into a format that an autonomy system can handle. In most robotic systems, this begins with decoding raw sensor data from the wire format and applying fundamental transformations such as image debayering and tone mapping, or LiDAR motion compensation.
- **Localization** computes vehicle poses with respect to some reference frame. As we will elaborate later in this chapter, localization can be performed using almost any sensor available on an SDV platform: IMUs, wheel speed sensors, LiDARs, cameras, RADARs, GNSS, etc. Leveraging a combination of sensors, such as LiDAR and IMU, is common for ensuring robust results. In practice, localizers typically compute two kinds of poses.

  1. A local ("continuous") pose can be used for reasoning about things in the immediate vicinity of the robot. This is used by components that require smoothness and local consistency, like the controller or LiDAR motion compensation. While the local pose may drift, it is updated at a high frequency and is guaranteed to be smooth.
  2. A global ("map-relative") pose is used for reasoning about things on the map. In contrast to the local pose, the global pose should be allowed to jump around in small amounts as new evidence is perceived without causing issues with autonomy. While continuous poses are typically updated very frequently, often in sync with an IMU, global pose updates often rely on heavier computations like point cloud registration and thus run at 10 Hz or slower. The focus of this thesis is on map-relative localization.

- **Perception** reasons about the location and type of the various objects in the scene surrounding the SDV. This encompasses both (potentially) dynamic objects like cars and pedestrians, as well as static objects like lane lines, traffic signs, etc. Perception can run with or without map input. If used, maps may be sparse structured maps, as those described in Chapter 3, or unstructured fully learned "hindsight features" as proposed by (You, Luo, Chen, et al. 2022). Perception can leverage a wide range of sensors such as cameras, LiDARs, RADAR, ultrasonics, etc. Multi-modal inputs can be beneficial because of the different strengths and limitations of different sensors. For example, RADAR may be noisy on its own but coupled with LiDAR it can improve robustness in adverse weather conditions (B. Yang et al. 2020).
- **Tracking** involves inferring correspondences between objects detected in the individual frames of a data stream, such as a video. In other words, tracking organizes temporally disjoint perception outputs into consistent tracks (Frossard and Urtasun 2018). This

can improve the quality of motion forecasting (W. Luo, Yang, and Urtasun 2018) and help reason about long occlusions (Luiten et al. 2021).

- **Prediction** (or motion forecasting) is tasked with understanding where the actors in the scene are likely to go in the near future. Prediction is challenging due to its inherent ambiguity. This ambiguity occurs at two levels: first, outputs need to be multi-modal and probabilistic, and their probabilities need to be well-calibrated. Second, at training time, only a single future unfolds from any given moment in the real data, limiting the amount of supervision, even with perfect labels. Prediction can be a standalone module, or it can be coupled with perception, planning, or both (W. Luo, Yang, and Urtasun 2018; Zeng et al. 2019). Its outputs may take a parametric form (Casas, Gulino, Suo, Luo, et al. 2020), in the shape of a sequence of waypoints for each actor at pre-defined times in the future, or a nonparametric one (A. Hu et al. 2021; Agro et al. 2023, 2024), in which case the model outputs probabilistic occupancy maps at different times in the future. The occupancy maps may be represented explicitly (A. Hu et al. 2021) or implicitly (Agro et al. 2023, 2024).
- **Motion Planning** is tasked with leveraging inferred information about current and future actor states and computing a safe, compliant, physically realistic, and comfortable trajectory for the vehicle which makes progress towards its high-level goal (Sadat et al. 2019). The planner needs information about the map around the SDV to ensure its outputs comply with the rules of the road; the map may be pre-computed (Sadat et al. 2019), or it may be the output of an online mapping algorithm that runs as part of the perception stage (Casas, Sadat, and Urtasun 2021).
- **The Controller** leverages local poses and other ego-state parameters together with the active motion plan to compute high-frequency actuation commands for braking, acceleration, and steering in order to follow the plan (Ort et al. 2019).

2.4 Localizing Ground Robots using Histogram Filtering

As motivated in Chapter 1, this thesis will primarily focus on the localization task, the cornerstone of all modules discussed above. Depending on the prior assumptions, we can divide localization into *global* and *online* localization.

- **Global localization** aims to position a robot in a map based on its sensory information without any prior knowledge about where the robot is. GPS-based methods and place recognition belong to this family, for example.

- **Online localization** positions the robot based on a strong prior of where it was on a previous time-step: this can enable efficient centimeter-level localization precision, but relies on having a good guess of where the robot is located. These methods are often unable to recover from an incorrect initialization without the help of a global localizer. Online localization methods include the inertial part of an RTK[2] system, as well as the LiDAR matching methods covered in this thesis.

[2] Real-Time Kinematic

In this thesis, we mainly focus on online localization. In every chapter except Chapter 7, we perform online localization against pre-built maps. The maps are constructed from multiple passes through the same area, which allows us to perform additional post-processing steps, such as dynamic object removal. The accumulation of multiple passes also produces maps that contain denser information than individual LiDAR sweeps.

The maps may contain different kinds of information, or *layers*, such as lane lines, traffic signs (see Chapter 3), orthographic bird's-eye view (BEV) images of the ground, or pre-computed neural network features (see Chapter 4).

Considering our focus on SDVs, which operate in structured human-built environments, we can restrict the task of computing the robot's 6-DoF 3D pose to 3-DoF by assuming its height, pitch, and roll can be inferred from the map itself. We assume that our sensors are calibrated and neglect the effects of suspension, unbalanced tires, and vibration, which allows us to simplify inference and efficiently leverage histogram filtering. This brings benefits in terms of simplicity and robustness. Since most chapters of this thesis involve performing online localization within a prior 2D map using LiDAR data in a histogram filtering framework, this section will give a primer on the mathematical elements shared among all methods.

The section is structured as follows. We first formulate localization as a recursive Bayesian estimation problem and discuss each probabilistic term. We then present a real-time inference algorithm and specify different observation sources which can be integrated, as will be detailed in upcoming chapters:

- Chapter 3 discusses a localization system which leverages energy terms for matching against lane lines and traffic signs.
- Deep LiDAR Localization (Chapter 4) uses an energy term that matches map and online LiDAR images processed using fully convolutional networks.
- The system in Chapter 5 uses the same framework and does not add new inputs to localization. Instead, it focuses on reducing the maps' storage requirements.

- Finally, Chapter 6 integrates the LiDAR matching observation model alongside perception into a multi-task framework. As with the past two chapters, the matching here is likewise performed against a dense LiDAR intensity map.

2.4.1 Recursive Bayesian Filtering

For the scope of this thesis, we will focus our attention onto a particular family of graphical models, namely recursive Bayesian filters, as they have numerous properties that are convenient when modeling robotic applications.

A histogram filter is an instance of recursive Bayesian inference which uses a discrete grid to make the inference problem tractable. While this approach also suffers from the same Curse of Dimensionality as Particle Filters, it has the benefit of dense support over the entire problem domain, eliminating the problem of particle exhaustion and reducing (though not eliminating) the risk of mode collapse (Thrun, Burgard, and Fox 2005).

2.4.2 Online 3-DoF Localization for Ground Robots

Let us now describe our 3-DoF inference algorithm mathematically within the well-established framework of recursive Bayesian filtering. Our 3-DoF pose consists of a 2D translation and a heading angle. We denote the SDV pose as $\mathbf{x} = \{x, y, \theta\}$, where $x, y \in \mathbb{R}$ and $\theta \in (-\pi, \pi]$, and use Bel to denote a probability distribution over the vehicle pose.

At each time step t, the localizer takes as input the previous most likely estimate of the pose $\mathbf{x}^*_{t-1}$ and distribution $\text{Bel}_{t-1}(\mathbf{x})$, the vehicle dynamics $\dot{\mathbf{x}}_t$, the online observation $\mathcal{Z}$, the vehicle dynamics observation $\mathcal{X}$, and the pre-built map $\mathcal{M}$.

$\mathcal{Z}$ may include camera, GPS, or aggregated LiDAR data. Aggregated LiDAR, denoted as $\mathcal{I}$, can be accumulated from the k most recent LiDAR sweeps using the IMU and wheel odometry. This produces denser online LiDAR images than just using the most recent sweep, helping localization. Since k is small, drift is not an issue.

We then formulate localization as a recursive Bayesian inference problem. We encode the fact that the online observation should be consistent with the map at the vehicle's location and that the belief updates should be consistent with the motion model. Thus,

$$\text{Bel}_t(\mathbf{x}) = \eta \cdot P_{\text{obs}}(\mathcal{Z}_t \,|\, \mathbf{x}; \mathcal{M}, \mathbf{w})\, \text{Bel}_{t|t-1}(\mathbf{x} \,|\, \mathcal{X}_t) \tag{2.6}$$

where P_{obs} represents the current frame observation model, which measures the probability of a sensor observation model (GPS, LiDAR, or cameras) given the map $\mathcal{M}$. The exact sensors depend on the specific formulation of the observation term, e.g., a pure LiDAR

matcher would not need cameras. The map $\mathcal{M}$ may also be omitted, for example, in the case of a GPS observation term. The observation model may include a set of parameters used in interpreting the observations, which we denote as $\mathbf{w}$. In practice, this term is formulated as an elementwise product of multiple specific terms, for example, the multiplication of a GPS energy and a BEV matching model. $\text{Bel}(\mathbf{x}_t)$ is the posterior distribution of the vehicle pose at time t given all the sensor observations until step t; η is a normalization factor. We do not need to calculate it explicitly because we discretize the belief space, so normalization is trivial.

2.4.3 Energy Terms Used in This Thesis

This section will focus on the specific energy terms used in the localizers discussed in this thesis.

Vehicle motion model. The vehicle motion model is encoded in the prediction term of our Bayesian filter, $\text{Bel}_{t|t-1}(\mathbf{x}|\mathcal{X}_t)$. It encodes the fact that the inferred state estimates should be consistent over time, and consistent with the current vehicle ego-motion $\mathcal{X}_t$, as perceived by its wheel encoders and inertial measurement unit (IMU). This motion model is encoded as

$$\text{Bel}_{t|t-1}(\mathbf{x} \,|\, \mathcal{X}_t) = \sum_{\mathbf{x}_{t-1} \in \mathcal{R}_{t-1}} P(\mathbf{x} \,|\, \mathcal{X}_t, \mathbf{x}_{t-1}) \text{Bel}_{t-1}(\mathbf{x}_{t-1}). \tag{2.7}$$

In practice, the wheel encoder and IMU information are fed into a separate Kalman Filter which estimates the ego velocity. This estimate is treated as a velocity observation $\mathcal{X}_t$ to our system, following a loose coupling philosophy.

Specifically, given an observation of the vehicle motion $\mathcal{X}_t$, the motion model is computed by marginalizing out the previous pose, where the likelihood is a Gaussian probability model

$$P(\mathbf{x}|\mathcal{X}_t, \mathbf{x}_{t-1}) \propto \mathcal{N}((\mathbf{x} \ominus (\mathbf{x}_{t-1} \oplus \mathcal{X}_t)), \Sigma) \tag{2.8}$$

with Σ the covariance matrix. Note that $\oplus$ and $\ominus$ are the standard 2D pose composition and inverse pose composition operators described by (Kümmerle et al. 2009).

GPS Energy. The GPS term measures the probability of the current GPS observation for a given vehicle pose $\mathbf{x}$. We approximate it using a simple 2D Gaussian distribution, whose variance σ^2_{GPS} is computed empirically:

$$P_{\text{GPS}}(\mathbf{g} \,|\, \mathbf{x}) \propto \exp\left(-\frac{(g_x - x)^2 + (g_y - y)^2}{\sigma^2_{\text{GPS}}}\right). \tag{2.9}$$

Here, $\mathbf{g} = [g_x, g_y]^T$ are the GPS observations, and $\mathbf{x} = [x, y]^T$ represents the 2-DoF vehicle pose, both expressed as easting and northing

in the Universal Transverse Mercator (UTM) coordinate system[3]. Our models do not make use of the GPS heading estimate.

[3] While it is possible to define these terms in a geodetic reference frame like WGS 84, the modeling task would be made more complex due to their non-Euclidean nature.

BEV Matching Model. Given a candidate pose $\mathbf{x}$, a bird's-eye view (BEV) matching model of the form P_{BEV} encodes the agreement between a current observation, expressed in BEV, and a BEV map indexed at the hypothesized 3-DoF pose $\mathbf{x}$. We warp the online observation according to each pose hypothesis and compute the cross-correlation between the warped online observation and the map. Formally, this can be written as:

$$P_{\text{BEV}}\left(\mathcal{Z}_t \,|\, \mathbf{x}; \mathbf{w}\right) \propto s\left(\pi\left(f_o(\mathcal{Z}_t; \mathbf{w}_o), \mathbf{x}\right), f_m(\mathcal{M}; \mathbf{w}_m)\right), \tag{2.10}$$

where f_o and f_m are the BEV representations of the online sensor data and the map, respectively. $\mathcal{Z}_t$ denotes the current online observation, and $\mathbf{w} = (\mathbf{w}_o, \mathbf{w}_m)$ encodes any additional parameters of the online and map terms[4]. π represents a 2D rigid warping function meant to transform the online BEV observation into the map's coordinate frame according to the given pose hypothesis $\mathbf{x}$[5]. Finally, s represents an affinity function, typically a cross-correlation operation.

[4] For example, if evaluating the term involves inference in a neural network, as is the case, e.g., in Chapter 4, $\mathbf{w}_o$ and $\mathbf{w}_m$ would encompass the network weights.

[5] The warp operation is linear and can be implemented efficiently by transforming the coordinates of the pixels in the target image back into the original and computing their values using bilinear interpolation. This is the resampling principle used by (Jaderberg et al. 2015), and its implementation is readily available in PyTorch. Thanks to the bilinear interpolation, the entire operation is differentiable, allowing it to be integrated into the optimization process for $\mathbf{w}$.

While the first two terms are either standard components of recursive Bayesian filtering (the motion model), or mathematically straightforward (the GPS energy), the BEV energy is where most of the novelty in the next four chapters is concentrated. The bird's-eye view observation $\mathcal{Z}_t$ can encode many types of information. In Chapter 3, it will take the form of a binary map denoting lane lines and traffic signs, mapped into 2D locations by use of LiDAR data, while in Chapters 4, 5, and 6, we will discuss BEV observation terms which directly leverage LiDAR data.

2.4.4 Efficient Inference

The recursive formulation expressed in Eq. (2.6) is generally intractable. In practice, efficient inference requires approximating the objective. This can typically be achieved by approximating the quantities with weighted samples, yielding a particle filter, or by discretizing the state space, which leads to a histogram filter.

The two approaches have different strengths and weaknesses: Particle filtering tends to have the maximum flexibility as particles can dynamically adjust the support of the distribution. On the other hand, histogram filtering provides dense support throughout the domain. However, histogram filtering also provides computational advantages when it comes to evaluating certain observation terms.

Both approaches can be used for local or global localization by selecting the scale of the support region—modeling the entire state space (e.g., the whole map) yields global localization, while modeling

it locally around the current likeliest pose estimate is equivalent to online localization. Effectively performing global localization at scale is not the focus of this thesis, and we instead focus on online localization for most of this thesis. We revisit global localization for our large-scale dataset analysis in Chapter 7, but we do not leverage any probabilistic model to filter observations.

The computational advantage of dense histogram filtering becomes apparent when we consider how BEV observation terms are modeled in this setting. Evaluating an observation term at a particular candidate location is computationally equivalent to a dot product. However, over a dense (x, y) region, this becomes a convolution[6], bringing numerous computational advantages.

[6] If we were using particle filtering, then evaluating each particle would involve a dot product at a completely different map location, which prevents any computation reuse.

In practice, we always evaluate a dense grid of $\mathbf{x}$, resulting in a 3D probability volume over (x, y, θ). We factorize the computation of this probability volume by first enumerating over n_θ discrete values of θ, and computing a convolution over (x, y) for each one. n_θ need not be large in practice, and the convolution at every step can be evaluated very fast even for large (x, y) search ranges, as we will see next.

Accelerating Correlation Enumerating the full translational search range with the inner product when evaluating a particular observation term is equivalent to a correlation filter with a large kernel. Motivated by the fact that the kernel is very large, we perform this operation in the spectral domain (i.e., "FFT-conv"), which accelerates it by a factor of 20 over the state-of-the-art GEMM-based spatial GPU correlation implementations.

This is enabled by the convolution theorem, which states that the Fourier Transform of two convolved signals is equal to the inverse of the dot product of the FFTs of the individual signals:

$$f * g = \mathcal{F}^{-1}\{\mathcal{F}\{f\} \cdot \mathcal{F}\{g\}\}. \tag{2.11}$$

In our case, f denotes the current BEV observation, while g denotes the areas of the map within which we are searching for a match. This area is typically on the order of several tens of meters at most. Since we deal with discrete tensors, we leverage the Fast Fourier Transform algorithm.

The run time of a spatial convolution between the two signals is $\mathcal{O}(n^2)$, where n is the total number of pixels. Leveraging the convolution theorem empowered by FFT, which is $\mathcal{O}(n \log n)$ results in an operation run-time of $\mathcal{O}(n + n \log n)$, which is dominated by the $n \log n$ term[7].

[7] In most convolutional neural network architectures, the dimensions of a convolution filter are small (often 3×3 or 5×5), which means that the overhead of FFT is not worth it. In our case, since our "filter" is the online BEV observation, hundreds of pixels in each dimension, the speedup is substantial, making the run-time penalty of the FFT worthwhile.

While extending the rotational component of the matching to the yaw dimension is not straightforward, it is typically acceptable to simply enumerate a few yaw offset candidates and perform the above procedure for each. By doing this, we end up with an x, y, n_θ distribu-

tion over poses.[8]

Point Estimation. At any time during the operation of state estimation, it may be necessary to extract a point estimate of the pose. For example, such an estimate is necessary for fusing map and online LiDAR information to pass them as input to a neural network which performs map-aided object detection. Unlike the MAP inference which simply takes the configuration that maximizes the posterior belief, we adopt a center-of-mass-based soft-argmax (Levinson and Thrun 2010) to better incorporate the uncertainty of our model and encourage smoothness in our localization. We thus define

$$\mathbf{x}_t^* = \frac{\sum_{\mathbf{x}} \mathrm{Bel}_t(\mathbf{x})^\alpha \cdot \mathbf{x}}{\sum_{\mathbf{x}} \mathrm{Bel}_t(\mathbf{x})^\alpha}, \tag{2.12}$$

where $\alpha \geq 1$ is a temperature hyperparameter[9]. This gives us an estimate that takes the uncertainty of the prediction into account.

[8] Matching over the rotational dimension can be performed using a Fourier-Mellin transform (X. Guo et al. 2005). However, this approach is less practical for online localization, where we are only interested in small relative offsets—in this case, simply enumerating a few offsets and performing FFT-convs for each of them is faster.

[9] Intuitively, for $\alpha = 1$ we recover a weighted average of the values in our grid, while for $\alpha \to \infty$, we return the mode of the posterior.

2.5 Related Work

Robot state estimation, especially localization, is one of the most widely studied areas of robotics. With applications ranging from navigation, shipping, and construction to autonomous robots and virtual reality, this task has been approached with nearly every sensing technique developed by humans.

While the task of localization has a history dating back to the very start of modern civilization, this section will mostly cover localization as it relates to modern robotic systems: starting from the mid-20th-century tower-based predecessors of GPS, such as LORAN-C, and up to the latest LiDAR registration methods using keypoints learned with deep learning (Du, Wang, and Cremers 2020; H. Wang et al. 2022). We will also briefly touch on related work relevant to other aspects of this thesis, such as perception, compression, and multi-task learning.

The 1980s and early 90s saw large innovations both in terms of absolute positioning, thanks to the development of systems such as GPS, as well as in terms of the newly emerged field of simultaneous localization and mapping (SLAM). SLAM aims to address the task of precise localization in environments which lack both prior maps as well as access to beacon-based systems like GPS, or where access to GPS is insufficient for the desired precision.

This section covers the major areas of global and online map-based localization, followed by brief overviews of SLAM and other work related to the remaining parts of the thesis, such as compression, multi-task learning, and large-scale localization.

2.5.1 Reference-Based Localization

The earliest techniques humans have employed for localizing themselves have been reference-based. Lakes, mountains, valleys, and celestial bodies such as the moon, the sun, and the stars have served as references for globally localizing travelers for many millennia (Reid et al. 2020). As highlighted in Fig. 2.3, the ranging accuracy tends to improve by an order of magnitude every decade.

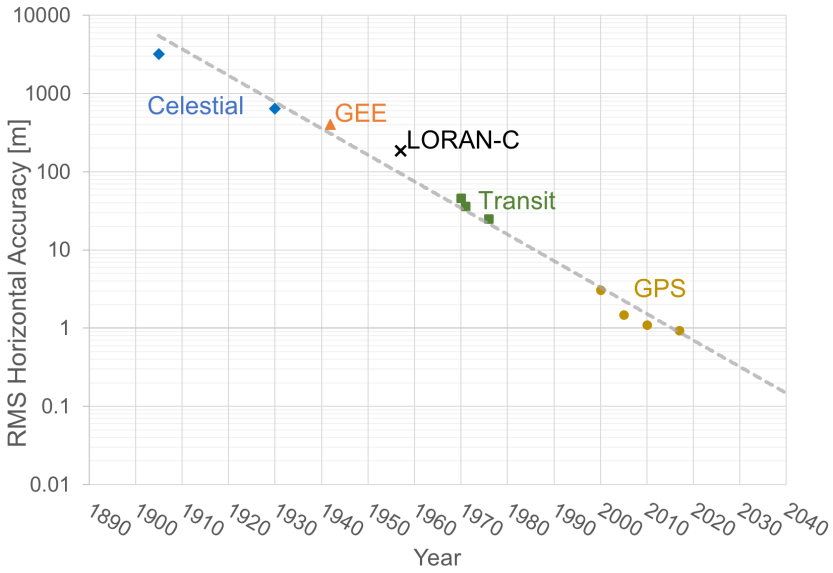


Figure 2.3: The evolution of reference-based localization error in the 20th century, from leveraging stars to the modern GPS. Figure from Reid et al. (Reid et al. 2020)

Following the development of radio technology, the 20th century witnessed many iterations of its application in navigation. The earliest systems were the UK's GEE and the United States LORAN (Wikipedia Contributors 2023), which relied on triangulating airplane positions from radio signals.

The development of GPS in the 1970s and 1980s made most radio navigation systems mostly obsolete[10]. GPS was soon joined by equivalent satellite navigation systems from China (BeiDou), Russia (GLONASS), the EU (GALILEO), and India (NavIC).

The operating principle of GNSS involves solving for the receiver's position on the surface of the earth by trilateration, using estimated ranges to at least three satellites which are in view[11]. Please refer to textbooks such as (P. Misra and Enge 2011) for additional details.

While not directly capable of the centimeter-level precision necessary for safe autonomous driving, GNSS systems can be enhanced in a variety of ways to achieve this goal. GNSS error reduction techniques can be divided into two major categories: Infrastructure-based and measurement-based. Techniques from the former category rely on ad-

[10] In spite of the large number of positive applications of GNSS, ranging from search and rescue all the way to how it has revolutionized mapping and navigation that penetrate our daily lives, it is worth remembering the dark history of the first system from this family: GPS. GPS reached full operational status in 1995, and its primary applications were military, with an emphasis on guiding ICBMs, as highlighted by one of the core mission statements of the GPS JPO: "[D]rop five bombs in the same hole." (Parkinson et al. 1996)

[11] While in principle we need four pseudo-ranges to solve for four unknowns, the receiver's 3D position plus its clock bias, given that we can assume the receiver is not located deep under the surface of the earth, three readings are enough.

ditional infrastructure, such as nearby base stations, to correct their readings to enhance precision, while those from the latter use more sophisticated methods to measure satellite ranges more accurately, leading to improved receiver position estimates. Techniques from both categories can be combined, as is the case with RTK, which leverages ground stations (first category) as well as phase shift measurements (second category) to improve positioning precision. We now present further details on the major error correction methods.

Differential GNSS (DGNSS). Differential positioning systems rely on a nearby ground station to correct satellite readings. The receiver uses code-based positioning, which is a simple and effective method for estimating satellite ranges. Consumer-grade GNSS receivers also use code-based positioning. While differential GNSS can be used on its own to enhance GNSS precision, differential readings are typically leveraged as part of an RTK system, which further leverages carrier wave phase shift measurements to significantly improve precision over code-based positioning alone.

Precise Point Positioning (PPP). PPP (Hofmann-Wellenhof, Lichtenegger, and Wasle 2007) is a protocol which uses live correction data pertaining to the satellites to integrate measurements over tens of minutes in order to account for various types of errors, such as due to atmospheric effects. PPP can typically achieve an accuracy of roughly 10cm (Joubert, Reid, and Noble 2020; Novatel Inc. 2015), without requiring a nearby reference station[12].

[12] While PPP does require additional data in the form of orbit corrections (ephemeris data) this information is typically available worldwide with relatively low latency over a cellular connection.

While not directly amenable to real-time mobile robotic localization due to the long convergence time, this technique can be leveraged in mapping by precisely localizing beacons that robots can later use. PPP can also be used as an inexpensive way to provide pose ground truth with reasonable accuracy within a global frame, such as UTM.

PPP implementations typically use an EKF which models the receiver position, clock errors, tropospheric delay, and carrier-phase ambiguities as its state (Novatel Inc. 2015).

Real-Time Kinematic (RTK). RTK systems extend code-based differential GNSS positioning with the carrier phase measurements also used by PPP, which can significantly increase the accuracy of range measurement, reaching centimeter-level accuracy under ideal conditions (satellite line-of-sight, close to a base station) (Novatel Inc. 2015). Unlike PPP, RTK systems require a base station to be within a few dozen kilometers of the roving receiver (e.g., the SDV). This means that, in contrast to PPP, RTK rovers no longer need to model atmospheric / multi-path errors themselves, which enables much faster convergence times.

Compared to basic DGNSS, RTK is much more accurate but requires more expensive equipment. Furthermore, DGNSS is able to

operate at a much larger distance from the base station compared to RTK (Novatel Inc. 2015).

The discussed GNSS error correction techniques are summarized in the table below.

Table 2.1: Different satellite-based positioning solutions and their main traits for mobile systems. Based on a Table from (Joubert, Reid, and Noble 2020).

	GPS	PPP	RTK	PPK
Accuracy Order of Magnitude	1.00m	0.10m	0.02m	0.02m
Convergence Time	20s	>10min	<20s	n/A
Requires Correction Service		yes	yes	yes
Requires Nearby Base Station			yes	yes
Cost	$	$$	$$$	$$
Needs Satellite LOS	yes	yes	yes	yes
Robust to Temporary Outages				yes

Post-Processed Kinematics (PPK). PPK is a less standardized term than RTK or PPP. PPK generally denotes a solution computed in a similar matter to RTK, but offline, after an entire dataset was collected, typically using non-causal information or global optimization. PPK can be leveraged in map building or as ground truth for evaluating state estimation algorithms quantitatively.

PPK follows a similar formulation to RTK, fusing IMU data, GNSS readings, and base station information with a kinematic model. Unlike RTK, PPK runs offline after a full sequence of data has been collected. PPK is typically implemented using a forward-backward filtering approach, though the implementation details of most commercial software are not public (R. Li 2023).

PPK can produce higher-accuracy solutions than RTK. Since it is computed offline, after the fact, it doesn't require the rover (SDV, drone, etc.) to have a continuous connection for receiving correction information, like RTK does. PPK may be used in map building and for ground truth trajectory estimation, but since it does not run online, it cannot be used directly for autonomous robotics[13].

[13] Commercial solutions (e.g., Novatel) can run best-effort RTK online while logging all the necessary data for running PPK later. This way, any outages which may cause temporary RTK inaccuracies online can be addressed later with post-processing.

Next-Generation GNSS. As of 2021, the deployment of the next generation of navigation satellites is well underway. Unlike previous GNSS systems, which leverage small constellations of satellites in geosynchronous Medium Earth Orbit (MEO), the next generation aims to leverage large constellations of micro-satellites in Low Earth Orbit (LEO). Xona Space Systems is one of the companies currently developing this technology (Reid et al. 2020). The advantages of this new paradigm are numerous, and include:

1. reduced cost,
2. higher precision,
3. improved integrity monitoring, and
4. increased robustness, encryption, and robustness to spoofing attacks[14].

The reduction in cost is achieved by using much smaller satellites. Deploying them in LEO[15] is cheaper, as the distance to Earth is at least 20× smaller compared to MEO.

The dramatic increase in the number of satellites visible at any given moment has lead to improvements in precision. Xona aims to launch 300 satellites in LEO. While this is a high number compared to the 31 GPS satellites, it is still dwarfed by the constellation sizes of LEO-based internet providers such as Amazon Kuiper or StarLink, which aim to deploy constellations of over 3,000 and 42,000 satellites, respectively. In addition to their increased number, the faster motion of the satellites with respect to the receiver can also improve the positioning precision (Reid et al. 2020).

Rebuilding a navigation system from scratch allows bypassing the need for backward compatibility and other legacy systems issues. This allows encryption and authentication to be built as a core consideration of the involved protocols, leading to improved robustness and security.

For a more detailed overview of GNSS for autonomous driving, please refer to the evaluations by (Joubert, Reid, and Noble 2020) and (Reid et al. 2020).

[14] Given the military applications of GNSS, spoofing is a common method of defense against satellite-guided weaponry. An interesting example of GPS spoofing was discovered in Moscow during the Pokemon Go fever of 2016 when players noticed large inaccuracies as "the fake signal, which seems to center on the Kremlin, relocated anyone nearby to Vnukovo Airport, 32 km away." (Hoffer 2017)

[15] Low Earth Orbit

2.5.2 Geometry-Based Localization

In spite of recent advances in GNSS, these technologies still fail to universally yield the sub-meter-accurate positioning required to effectively leverage HD maps and the centimeter-accurate positioning required to build digital twins for simulation. This is especially the case in areas without a clear view of the sky, such as urban canyons, inside buildings, tunnels, etc. Furthermore, achieving cm-level RTK localization typically requires expensive hardware as well as costly subscriptions to correction services. The time required to acquire a fix is sometimes also a problem, often taking between a few seconds and a few minutes.

To this end, a wide range of alternate approaches to localization have been developed. The first family of such methods tackles localization by geometrically registering an observation to a 3D model of a scene, typically using a perspective-n-points (P-n-P) approach. The idea is to extract local features such as SIFT (Lowe 2004) from images and find correspondences with the prior geo-registered point sets (Y. Li et al. 2012, 2012; Dusmanu et al. 2019; Irschara et al. 2009;

Sattler, Leibe, and Kobbelt 2011; Sattler et al. 2015; L. Liu, Li, and Dai 2017). (Y. Li et al. 2012) pre-stored point clouds along with SIFT features for this task, while (L. Liu, Li, and Dai 2017) proposed to use branch-and-bound to solve the exact 2D-3D registration.

As a prerequisite, these methods require a 3D model of the world to be built in advance. Cameras perceive a 2D projection of the 3D scene, so research in this area has focused on building consistent 3D maps of the world (Agarwal et al. 2009), often using techniques based on bundle adjustment (Triggs et al. 1999) or multi-view stereo (MVS) (Yao et al. 2019).

Geometry-based localization can also be performed using LiDAR. The wide range of work developed for general-purpose point cloud registration can be leveraged to align observations to a known map. This includes both local methods such as iterative closest point (ICP) (Yoneda et al. 2014) and Normal Distribution Transform (NDT) (Biber and Straßer 2003) which require a coarse initialization in the rough vicinity of the true location, as well as global ones, which register overlapping point clouds without the need for an initial estimate of the relative transform. This family includes both correspondence-based methods, such as Fast Global Registration (Q.-Y. Zhou, Park, and Koltun 2016), Deep Global Registration (Choy, Dong, and Koltun 2020), DH3D (Du, Wang, and Cremers 2020), 3DRegNet (Pais et al. 2020), or Deep Closest Point (Y. Wang and Solomon 2019), as well as correspondence-free methods, like PHASER (Bernreiter et al. 2021).

While these approaches can be very accurate, several drawbacks remain. Scalability (maintaining a very large 3D database), for example, remains challenging. While fine vocabularies (Sattler et al. 2015; Havlena and Schindler 2014) and model compression (Camposeco et al. 2019; Lynen et al. 2020) provide ways to accelerate matching in large scenes, accuracy suffers, and building and storing 3D models at city scale requires large engineering efforts. Building systems that are robust to long-term changes (Sattler et al. 2018) also remains an active area of research.

2.5.3 Place Recognition

Another prevailing approach in self-localization is place recognition (M. Bansal and Daniilidis 2014; Hays and Efros 2008; Y. Li et al. 2012; Moosmann and Stiller 2013; Wolcott and Eustice 2014; Arandjelovic et al. 2016; Zamir, Hakeem, and Szeliski 2016; Jégou et al. 2011; Gálvez-López and Tardós 2012; Sarlin et al. 2019; Lindenberger, Sarlin, and Pollefeys 2023; S. Zhu et al. 2023). Place recognition aims to bypass the need for reference stations or an explicit map, opting to formulate localization as a similarity search problem.

In general, methods from this line of work encode sensor readings (e.g., camera images) with known positions into a low-dimensional representation and store them in a compact database[16]. At query time, a new sensor reading is received, encoded, and looked up. The pose for the most similar database entry is retrieved. Given the small size of the sensor encoding, this setting allows billions of encoded sensor readings to fit in RAM, with efficient indexing allowing approximate nearest neighbor queries in milliseconds (Johnson, Douze, and Jégou 2017; Martinez-Covarrubias 2018; Douze et al. 2024).

[16] For example, a 4 million pixel RGB image could be projected down to 128 32-bit floating point numbers—a 100,000× compression for 4MP 8-bit RGB compressed to 1024 32-bit float.

By designing a sensor encoding process that preserves enough discriminative information, we can achieve robust results. Robustness can be further improved by geometric checks (Gálvez-López and Tardós 2012). As most of the features used to describe the scene (e.g., 3D line segments (M. Bansal and Daniilidis 2014) or 3D point clouds (Y. Li et al. 2012; Moosmann and Stiller 2013; Wolcott and Eustice 2014)) are highly correlated with the appearance of the world, one needs to update the database frequently. With this problem in mind, (Cummins and Newman 2008; Nelson et al. 2015; Linegar, Churchill, and Newman 2015) proposed image-based localization techniques that are, to some degree, invariant to appearance changes. Deep learning approaches have brought large improvements in computing discriminative descriptors, for example, by training fully convolutional networks to predict descriptor feature maps and keypoint heatmaps using combinations of self-supervision and supervised learning (DeTone, Malisiewicz, and Rabinovich 2018).

Image retrieval-based localization. Classical methods extract local invariant features, such as SIFT (Lowe 2004) or SURF (Bay et al. 2008). and aggregate them into a global descriptor such as visual bag-of-words (Filliat 2007) or VLAD (Jégou et al. 2011; Torii, Arandjelovic, et al. 2015). Candidate re-ranking and geometric verification are sometimes used as a second stage to boost performance further (Knopp, Sivic, and Pajdla 2010; Sattler et al. 2017). Recent work has used deep convolutional neural networks (CNNs) to learn compact visual representations (Arandjelovic et al. 2016; Gordo et al. 2016; Radenović, Tolias, and Chum 2016; Tolias, Sicre, and Jégou 2016). For instance, NetVLAD (Arandjelovic et al. 2016) uses a CNN and differentiable VLAD pooling to learn global image representations for retrieval in an end-to-end manner, and RMAC (Tolias, Sicre, and Jégou 2016) builds a compact deep feature vector with Region of Interest (RoI) pooling. (Gordo et al. 2016) extended this work by leveraging a ranking framework to learn features, as well as the region pool scheme itself. This process can be enhanced by leveraging coarse 3D models from the internet (Panek, Kukelova, and Sattler 2023).

Hybrid approaches combining place recognition with local feature

detection have also been proposed. One example is HF-Net (Sarlin et al. 2019), which trains a joint network to compute a NetVLAD descriptor as well as SuperPoint (DeTone, Malisiewicz, and Rabinovich 2018) local features based on a shared feature backbone, saving computation.

Recent approaches such as SuperGlue (Sarlin et al. 2020) and LightGlue (Lindenberger, Sarlin, and Pollefeys 2023) cast feature matching as inference in a graph neural network or a transformer, respectively, learning the process end-to-end by supervising with ground truth matches. These methods can refine the retrieval result of place recognition to achieve state-of-the-art localization accuracy on challenging benchmarks such as HPatches (Balntas et al. 2017) and MSLS (Warburg et al. 2020).

Many newer approaches move yet another step forward, and extend learning to the ranking process itself. (Hausler et al. 2021; S. Zhu et al. 2023) learn to re-rank a set of top candidates obtained through retrieval. For example, R^2Former (S. Zhu et al. 2023) uses a transformer to attend to patches from the top-k retrieved candidates in order to re-rank them and identify the best one with improved accuracy.

LiDAR retrieval-based localization. While handcrafted 3D descriptors have been used for 3D registration and recognition tasks (Tombari, Salti, and Di Stefano 2010; Rusu, Blodow, and Beetz 2009), we are not aware of classical global pooling techniques applied to LiDAR retrieval-based localization.

Recently, following the success of deep learning in extracting data-driven features, work has concentrated on learning deep descriptors from 3D point clouds (L. He, Wang, and Zhang 2016; Dewan, Caselitz, and Burgard 2018; Klokov and Lempitsky 2017). PointNetVLAD (Uy and Lee 2018) uses PointNet (Qi, Su, et al. 2017) to generate local per-point features, which are then aggregated by a VLAD (Arandjelovic et al. 2016) layer. PCAN (W. Zhang and Xiao 2019) improves upon PointNetVLAD by learning an attention map for aggregation, using an architecture inspired by PointNet++ (Qi, Yi, et al. 2017). LPD-Net (Z. Liu et al. 2019) achieves improved retrieval results using a graph neural network to leverage local structure when learning global descriptors. While these methods yield excellent results, they operate directly on raw point clouds, which is computationally expensive in general. Recent works have tackled the challenge of efficient inference either by using specialized sparse 3D convolution operators (e.g., MinkLoc3D (Komorowski 2021)) or by operating on LiDAR range view images and exploiting much faster 2D network architectures (e.g., OverlapNet (Xieyuanli Chen et al. 2022) and OverlapTransformer (J. Ma et al. 2022)).

In closing, the accuracy of retrieval methods is limited by the den-

sity and coverage of the underlying database, and finding compact yet discriminative representations remains difficult.

2.5.4 Pose Regression Methods

Moving one step beyond map-less methods like place recognition, pose regression methods approach the task of global localization by implicitly encoding the environment structure, typically in the weights of a neural network.

Scene coordinate regression methods, such as the influential work by Shotton et al., which also introduced the widely used "7 Scenes" dataset (Shotton et al. 2013), aim to localize a camera in a known scene by learning to regress the 3D coordinates of observed 2D pixels directly. This allows the camera pose to be inferred using a direct solver inside a RANSAC loop, without explicitly storing the map—the coordinate regression model implicitly encodes it.

(Brachmann et al. 2017) continue this line of work and present Differentiable SAmple Consensus (DSAC), a soft relaxation of RANSAC which enables end-to-end training of hypothesis generation through the otherwise non-differentiable sampling procedure, demonstrating its efficiency and robustness in the task of scene coordinate regression. This work demonstrates the importance of end-to-end training for this task, with the proposed end-to-end method significantly outperforming prior methods such as (Shotton et al. 2013), which do not optimize the feature representations for the end task of camera pose estimation.

However, methods from this line of work are not without their limitations. Since the regression model needs to encode the mapping between observations and *absolute* scene coordinates, it needs to be re-trained for every scene, limiting the scalability of methods from this family. Moreover, these methods struggle to generalize to large environments, with most evaluations limited to indoor datasets, or datasets covering around one or two city blocks (e.g., Cambridge Landmarks (Kendall, Grimes, and Cipolla 2015)).

Recent innovations such as ACE (Brachmann, Cavallari, and Prisacariu 2023) have nevertheless made great strides in addressing the generalization limitations by reusing the backbone across all scenes, and by accelerating scene-specific training by several orders of magnitude. Scalability to city-sized scenes, however, remains an open research question.

Direct regression methods such as PoseNet (Kendall, Grimes, and Cipolla 2015) represent a similar but streamlined line of work which aims to directly regress the pose of a query observation in a known environment (Shotton et al. 2013; Kendall, Grimes, and Cipolla 2015; Brachmann et al. 2016; Brachmann and Rother 2018;

Kendall and Cipolla 2016). Unlike scene coordinate methods, approaches such as PoseNet bypass the fine-grained coordinate regression stage and instead directly predict the 6-DoF camera pose from the raw input pixels using a convolutional neural network. These methods have two major advantages. First, they are conceptually simple and perform well in small and medium-sized environments. Second, just like screen coordinate regression, direct regression methods enable global localization without the need for an external database, leading to low memory usage and fast inference.

While promising, these methods have been shown to not generalize well to city-scale localization (Sattler et al. 2018, 2019). Their inability to work in areas on which they have not been trained makes transitions to new environments computationally demanding, as it would require collecting sensor data and ground truth pose information from a novel environment, then training a neural network before deploying a robot to the environment.

Multi-Task Localization. Recent work has explored the complementarity of semantics, temporal information, and global pose regression (Valada, Radwan, and Burgard 2018; Radwan, Valada, and Burgard 2018; Schönberger et al. 2018; P. Wang et al. 2018). VLocNet++ (Radwan, Valada, and Burgard 2018) combines the pose regression of PoseNet with visual odometry and semantic segmentation in a multi-task setting, showing that jointly solving these tasks can be mutually beneficial while also improving run-time efficiency by sharing computation between the tasks.

2.5.5 Lightweight Map-Based Localization

While retrieval, regression, and structure-based localization methods have achieved promising performance on a wide range of benchmarks, they either suffer from large costs associated with storing image databases or lack the accuracy required for safe autonomous driving in safety-critical scenarios.

Likewise, pure reference-based methods like GPS can achieve automotive-grade accuracy but only by leveraging base station corrections in the form of RTK, which limits scalability, as the improvements are conditioned on the availability of such nearby stations (Wan et al. 2018; P. Misra and Enge 2011). Scalability is further limited by the subscription costs required to access base station corrections in real-time.

As highlighted in the introduction, one main reason to perform localization is to leverage various kinds of data already encoded in a map, such as a navigation road graph or detailed lane information. Many approaches have, therefore, been developed to bypass the

aforementioned issues and instead leverage the semantic information already encoded in the map required by the autonomy system. Both high-level information, like a routing graph, as well as low-level information, like the exact lane structure, can be leveraged to achieve high-precision localization superior to that of GPS without reliance on dense 3D models, sparse geo-referenced databases, or black-box neural networks.

One line of work uses "lightweight'' maps like OpenStreetMap and Google Maps for precise localization. (Brubaker, Geiger, and Urtasun 2013) demonstrated a lightweight and scalable solution based on particle filtering which leveraged crowdsourced OpenStreetMap data to globally localize vehicles with sub-meter accuracy in city-sized maps using cues such as road curvature. (W.-C. Ma et al. 2017) built upon this framework to leverage additional cues, such as the road type and the position of the sun, to further improve robustness. Given an initial estimate of the vehicle position, (Floros, Zander, and Leibe 2013) exploited the local shape of the ego-trajectory to self-localize within a small region.

These works are appealing since they only require a cartographic map. However, the localization accuracy is strongly limited by the performance of odometry. The semantic cues are only used to resolve ambiguous modes and speed up the inference procedure. Second, the computational complexity is a function of the uncertainty in the map, which remains fairly large when dealing with maps that have repetitive structures.

LaneLoc (Schreiber, Knöppel, and Franke 2013) proposed to use lane lines as localization cues. Towards this goal, the authors manually annotated lane markings over the LiDAR intensity map. The lane markings are then detected online using a stereo camera and matched against the ones in the map. (Welzel, Reisdorf, and Wanielik 2015) and (Qu, Soheilian, and Paparoditis 2015) utilize traffic signs to assist image-based vehicle localization. Specifically, their approach detects traffic signs from images and matches them against a geo-referenced sign database, after which it conducts local bundle adjustment to estimate a fine-grained pose. However, the effectiveness of such methods depends on perception performance and does not work for regions where such cues are absent. More recently, (Schönberger et al. 2018) built dense semantic maps using image segmentation and conducted localization by matching both semantic and geometric cues.

2.5.6 Dense High-Definition Maps

While visual place recognition and lightweight localization can disambiguate the robot's location and provide a good estimate of its

position within a map, methods belonging to this category are still unable to achieve the centimeter-level precision necessary for tasks like safe autonomous driving or 3D reconstruction.

In order to achieve performance of this magnitude, the use of high-definition maps (HD maps) has gained attention in recent years (Ziegler et al. 2014; Levinson, Montemerlo, and Thrun 2007; Levinson and Thrun 2010; Wolcott and Eustice 2015, 2014; Wan et al. 2018). The general idea is to build an accurate map of the environment offline by aligning multiple sensor passes over the same area. In the online stage, the system is able to achieve sub-meter-level accuracy by matching the sensory input against the HD map.

While map matching has been studied in broader contexts since the 70s (Kuglin 1975), it was the landmark paper by (Levinson, Montemerlo, and Thrun 2007) that popularized the use of map matching for self-driving vehicle localization. The authors proposed building 2D bird's-eye view LiDAR intensity maps offline with Graph-SLAM (Thrun and Montemerlo 2006) and used particle filtering and Pearson product-moment correlation to localize against them. Subsequent work (Levinson and Thrun 2010; Wolcott and Eustice 2015) highlights the importance of encoding variance in the map data, in order to robustly predict which online observations are likely unreliable, e.g., due to originating in areas which are often occluded during mapping.

These methods are capable of real-time operation on a CPU using map resolutions up to 5 cm/px (Levinson, Montemerlo, and Thrun 2007). However, a quasi-manual calibration procedure is needed to meaningfully compare LiDAR intensities between runs, which severely limits their scalability to large fleets and environments. In a similar fashion, (Wan et al. 2018) use BEV LiDAR intensity images in conjunction with differential GPS and an IMU to robustly localize against a pre-built map, using a Kalman filter to track the uncertainty of the fused measurements over time.

Finally, (Y. Zhou et al. 2020) demonstrate that LiDARs may not be necessary for accurate and robust localization in LiDAR maps. Instead, the authors train a convolutional neural network to match between visual features perceived online and features stored in a LiDAR map. Instead of explicitly matching features like in, e.g., SLAM, the method performs online localization by an exhaustive search: for a given initial pose guess, such as one predicted by a pose filter, the system generates a grid of nearby candidates in the nearby (x, y, yaw) space. It then evaluates matching scores between projected map features and online-computed visual features at each candidate pose, aggregating per-pixel scores to produce one score per pose hypothesis. These scores are then used as observation weights in order to yield a belief over the vehicle's current pose.

2.5.7 Simultaneous Localization and Mapping (SLAM)

In many applications, it is not possible to build a map of an environment in advance. For example, many robots in domains such as space exploration or search-and-rescue need to operate autonomously in a new environment while mapping it. In this case, robot poses, as well as the environment map, need to be estimated jointly, resulting in the Simultaneous Localization and Mapping (SLAM) task. SLAM has been one of the pillars of robotics research since the 1980s (Bailey and Durrant-Whyte 2006; Engel, Schöps, and Cremers 2014; Mur-Artal, Montiel, and Tardós 2015).

More formally, given a sequence of images, point clouds, or other sensor readings, SLAM is tasked with jointly estimating an accurate and consistent robot trajectory together with a map of the environment without any prior information on its structure.

This can be accomplished by estimating relative robot poses between consistent frames through feature correspondences followed by pose estimation, which is typically alternated with periodic optimizations of the entire scene structure.

Unfortunately, since the estimation error is usually biased, accumulated errors cause gradual estimation drift as the robot moves, resulting in large errors. *Loop closure* is a common technique used to fix this issue. Typically based on some form of place recognition (cf. Chapter 2.5.3), loop closure constrains the robot trajectory, enabling accumulated drift to be reduced by propagating information from an already-visited part of the environment.

Visual SLAM. Cameras have been widely studied in SLAM applications. While inexpensive and widely available, cameras are unable to perceive direct scale, which adds additional challenges to the SLAM problem in the form of scale ambiguities. Examples include monoSLAM (Davison et al. 2007), PTAM (Klein and Murray 2007), ORB-SLAM (Mur-Artal and Tardós 2017), and Kimera (Rosinol et al. 2020).

While a detailed treatment of visual SLAM and visual bundle adjustment lies beyond the scope of this thesis, the foundational overview by (Triggs et al. 1999) provides a detailed overview of the nature of the problem, main challenges, common formulations, and efficient solvers, while also discussing background knowledge for and the history of large-scale least-squares optimization.

LiDAR SLAM. Finally, in recent years, LiDAR SLAM methods such as ICP-SLAM, LeGO-LOAM, and LIO-SAM (Shan et al. 2020) have demonstrated extremely accurate map building, with front-end odometry drifting less than a meter per km (Shan et al. 2020). Multi-sensor approaches such as LIC-Fusion (Zuo et al. 2020) fuse LiDAR

odometry estimates with visual estimates obtained by matching sparse camera features, further boosting accuracy and increasing robustness to sensor failures.

Neural SLAM. Given the explosion in research on neural 3D representations in the last four years, a growing body of work has explored these avenues for SLAM. The envisioned benefits include more robust 3D maps thanks to the neural representations' natural ability to in-paint, as well as improved camera tracking thanks to (meta-)learned registration and alignment priors.

Earlier papers like NeRF-- (Z. Wang et al. 2021) and BARF (Lin et al. 2021) extended neural rendering (Mildenhall et al. 2021) to optimize over poses, in addition to the radiance field itself. iMapping (Sucar et al. 2021) proposed to do so incrementally, describing several techniques to ensure efficient operation, such as information-guided pixel sampling. Later methods such as (Z. Zhu et al. 2022) extend the approach to larger scenes using multi-resolution voxel hashing (Müller et al. 2022), or to multi-sensor calibration settings by also incorporating sensor intrinsics into the optimization procedure (Herau et al. 2024; Z. Yang et al. 2024).

While the goal of SLAM is to build globally consistent maps and trajectories, a subset of papers simply focus on estimating a good camera trajectory without worrying about loop closure. These methods are known as visual (Engel, Koltun, and Cremers 2018) or LiDAR (J. Zhang and Singh 2014) odometry.

2.5.8 Visual and LiDAR Odometry

Unlike SLAM, pure odometry methods build incremental estimates of the trajectory without enforcing global consistency (Geiger, Ziegler, and Stiller 2011; Engel, Koltun, and Cremers 2018; Sen Wang et al. 2017; Xu et al. 2022). This may reduce the accuracy of the final trajectory and maps; however, for many tasks the resulting accuracy is sufficient and its simplified computation also reduces the cost of backend operations (no loop closure checks, no PGO, simple or non-existent map management, etc.).

Visual odometry may be computed using keypoint features (Geiger, Ziegler, and Stiller 2011) or by exploiting photometric optimization to achieve fast and effective estimates without relying on keypoint extraction, which may be unreliable in textureless environments (Engel, Koltun, and Cremers 2018). Recent approaches such as DeepVO (Sen Wang et al. 2017) have explored the use of deep learning to replace traditional keypoint matching or photometric optimization. Nevertheless, given their reliance on prior information encoded in HD maps, it is common for aaaa to localize in existing maps rather than perform

SLAM or visual odometry.

2.5.9 Matching Networks

Many localization approaches, such as (Levinson, Montemerlo, and Thrun 2007), rely on measuring the affinity between an observation and a particular location in the map. While papers such as this one leverage traditional functions from the signal processing literature, like normalized cross-correlation, recent papers have explored ways of learning these functions from data. Many modern approaches train CNNs[17] to measure similarity between patches across images and volumes (Long, Zhang, and Darrell 2014; W. Luo, Schwing, and Urtasun 2016; Zhuoyuan Chen et al. 2015; Zbontar and LeCun 2015; Zagoruyko and Komodakis 2015). The pioneering work of (Long, Zhang, and Darrell 2014) directly transfers low-level CNN features pre-trained on ImageNet to feature matching. In (Zbontar and LeCun 2015), a siamese network is trained to match local patches, with applications in stereo matching. (W. Luo, Schwing, and Urtasun 2016) improve the matching efficiency by eliminating the patch-based inference, while also producing confidence estimates for the computed disparities. GC-Net (Kendall et al. 2017) learns not just the representations but the matching operation itself, which is modeled as a sequence of stacked hourglass CNNs. EpicFlow (Revaud et al. 2015) extends this framework by densifying the sparse matches using a novel interpolation scheme before performing variational energy minimization.

[17] Convolutional Neural Networks

More recently, RAFT (Teed and Deng 2020) demonstrated robust optical flow estimation and promising sim2real generalization by learning to iteratively update flow estimates with a recurrent network, mirroring the steps of a first-order optimization algorithm. In addition to winning the best paper award at ECCV 2020, RAFT's success has been demonstrated by the richness of works that build on its structure to improve scene flow (Teed and Deng 2021b) or SLAM (Teed and Deng 2021a). The rapid advances in learned affinity functions highlight the importance of exploiting data-driven priors for matching tasks.

2.5.10 Compression

Compression has been widely studied in computer science ever since Claude Shannon (Shannon 1948) first proposed the source coding theorem. This has led to a wide array of applications in the field, ranging from the development of error correction codes like turbo codes (Berrou, Glavieux, and Thitimajshima 1993) to fast and effective compression algorithms such as DEFLATE (Wikipedia Contributors

2024a) (the foundation of formats such as `zip`) and Snappy (Wikipedia Contributors 2024b).

Compression methods can be categorized in two ways. They can be classified based on their ability to perfectly recover the original data into lossy and lossless, and based on their target domain into general-purpose and domain-specific. General-purpose algorithms such as DEFLATE are applicable to any kind of data, from text to multimedia, and they are typically lossless. Special-purpose algorithms are tailored to a specific kind of data, such as images, and exploit the particular structure of this data in order to achieve results superior to those of general-purpose algorithms. In this thesis, we focus on the domain-specific task of compressing map data encoded as images, using both lossless and lossy approaches.

Image Compression. Image compression is a classical subfield of computer vision and signal processing. It has seen a great deal of progress in the past few years thanks to the development of hardware acceleration and the advent of deep learning (Yibo Yang et al. 2023). In most modern incarnations, a learning-based compression method consists of an encoder network, a quantization mechanism (e.g., binary codes), and a decoder network which reconstructs the input from the quantized codes. Recent learning-based approaches (Toderici et al. 2017; Rippel and Bourdev 2017; Mentzer et al. 2018) consistently outperform classic compression methods like JPEG2000 and BPG. While earlier works on learned compression typically used a standard autoencoder architecture (Balle, Laparra, and Simoncelli 2016), thereby imposing a fixed code size for all images, (Toderici et al. 2016) overcome this limitation by using a recurrent neural network as an encoder. Subsequently, (Toderici et al. 2017) extended the previous results, which were typically presented on resolutions of 64×64 pixels or less due to performance considerations, demonstrating state-of-the-art compression rates on full-resolution images. (Mentzer et al. 2018) proposed a pipeline that obtained results on par with the state of the art while also being trainable end-to-end (encoder, quantizer, and decoder). Nevertheless, in spite of their superior compression rates, it is very difficult for these approaches to compete with the speed, energy efficiency, broad compatibility, and ubiquity of modern codecs such as JPEG and AVIF.

Map Compression. In spite of their advantages in providing a strong signal for accurate localization, maps can end up with huge storage requirements, particularly when dense representations are used, like in the case of LiDAR maps (Yoneda et al. 2014; Levinson, Montemerlo, and Thrun 2007).

Compression has also been studied in the task of localization and mapping. (Camposeco et al. 2019) proposed a hybrid approach con-

sisting of downsampling 3D map points and compressing their descriptors, while (Q. Zhou et al. 2022) demonstrated promising visual localization results without the need to store point descriptors at all. Similar techniques can also be applied to dense maps. (Bârsan et al. 2018) showed that even simplistic pruning of uncertain parts in volumetric maps can dramatically reduce memory usage with negligible costs in map density. More recently, (Wiesmann et al. 2022) and (Cai et al. 2024) have proposed learning-based approaches for online and global localization, respectively, using compressed LiDAR maps.

2.5.11 Perception, Prediction, and Planning

Object Detection and Motion Forecasting. Detecting actors and predicting their future motion from sensor data is one of the core tasks in autonomous driving. While object detection (B. Yang, Luo, and Urtasun 2018; Shi, Wang, and Li 2019; Meyer et al. 2019) and motion forecasting (Zheng, Yue, and Hobbs 2016; Cui et al. 2019; Chai et al. 2020; Gao et al. 2020; Phan-Minh et al. 2020; N. Mu et al. 2024) can be modeled as independent tasks, models that jointly perform both tasks (W. Luo, Yang, and Urtasun 2018; Casas, Luo, and Urtasun 2018; Liang et al. 2020; Zhishuai Zhang et al. 2020; Y. Hu et al. 2023) have been shown to provide a number of benefits, such as fast inference, uncertainty propagation, and overall improved performance.

Multi-task learning for perception. Compared to end-to-end approaches for autonomous agents that learn to directly map sensor readings to control output (Bojarski et al. 2016; M. Bansal, Krizhevsky, and Ogale 2019; Kendall et al. 2019), multi-task modular approaches have been shown to perform better empirically (B. Zhou, Krähenbühl, and Koltun 2019), while also being more interpretable thanks to human-readable intermediary representations like semantic segmentation (B. Zhou, Krähenbühl, and Koltun 2019), object detections (Zeng et al. 2020), occupancy forecasts (Sadat et al. 2020), and planning cost maps (Zeng et al. 2019). Furthermore, (Liang et al. 2019) have shown that tasks like mapping, object detection, and depth completion can be mutually beneficial when formulated under a unified architecture.

The wide range of recent multi-task learning approaches can be divided into two major areas. One line of work is focused primarily on developing and understanding network architectures, such as those leveraging a common backbone with task-specific heads (Collobert and Weston 2008; Kaiser et al. 2017; Kendall, Gal, and Cipolla 2018; Teichmann et al. 2018; W. Luo, Yang, and Urtasun 2018; Sarlin et al. 2019; Du, Wang, and Cremers 2020; Cao, Araujo, and Sim 2020),

cascaded approaches where some tasks rely on the outputs of others (Dai, He, and Sun 2016; Hashimoto et al. 2017; K. He et al. 2017; Zeng et al. 2019, 2020), or *cross-talk* networks such as Cross-Stitch (I. Misra et al. 2016), which have completely separate per-task networks, but share activation information.

Modular learning approaches, such as Modular Meta-Learning (Alet, Lozano-Pérez, and Kaelbling 2018), aim to construct reusable modular architectures which can be re-combined to solve new tasks. Side-Tuning (J. O. Zhang et al. 2020) proposes an incremental approach where new tasks are added to existing neural networks in the form of additive side-modules that are easy to train, and have the advantage of leaving the weights of the original network unchanged, bypassing issues such as catastrophic forgetting. Recently, approaches such as EmerNeRF (J. Yang et al. 2023) have proposed formulations which unify perception with novel view synthesis by integrating volumetric rendering objectives into perception pre-training.

Another line of work is concerned with the optimization process itself. The most straightforward approach is sub-task weighting, which may be based on uncertainty scores (Kendall, Gal, and Cipolla 2018), learning speed (Zhao Chen et al. 2018), or performance (M. Guo et al. 2018). Other methods have explored multi-task learning by performing multi-objective optimization explicitly (Sener and Koltun 2018), by regularizing task-specific networks through soft parameter sharing (Yongxin Yang and Hospedales 2017), or through knowledge distillation (Buciluǎ, Caruana, and Niculescu-Mizil 2006; Clark et al. 2019). Please see (Crawshaw 2020) for a detailed survey of multi-task deep learning.

Planning Under Pose Uncertainty. The task of planning robust trajectories under pose uncertainty has been studied in the past, with previous methods formulating it as a continuous POMDP which can be solved with an iterative linear-quadratic-Gaussian method (van den Berg, Patil, and Alterovitz 2012), or as an optimal control problem solved using model-predictive control (Indelman, Carlone, and Dellaert 2015). (van den Berg, Patil, and Alterovitz 2012) have studied motion planning under uncertainty, and formulated the problem as a continuous POMDP, solving it using an iterative Linear-Quadratic-Gaussian method. (Indelman, Carlone, and Dellaert 2015) likewise operate in a continuous domain, casting motion planning as an optimal control problem and solving it using a model-predictive control method. More recently, (Artuñedo et al. 2020) focus on autonomous vehicles and incorporate the pose uncertainty in a probabilistic map representation that is then leveraged by a sampling-based planner, while (Zichao Zhang and Scaramuzza 2020) propose an efficient way of estimating visual localization accuracy for use in motion planning.

Finally, (Amini et al. 2019) achieve robust planning in the face of coarse topometric maps and inaccurate localization by learning to predict distributions over control commands using a variational formulation. However, none of these approaches model other dynamic actors and the uncertainty in their own motion, and they do not study the complex interplay between pose uncertainty and state-of-the-art perception systems.

System-Level Analysis. A number of recent papers have studied the correlations between task-level metrics, such as object detection and system performance (Philion, Kar, and Fidler 2020). This line of work has shown that while task-level metrics serve as good predictors of overall system performance, they are unable to differentiate between similar errors that can, however, lead to very different system behaviors. A related line of work analyzed the impact of sensor and inference latency on object detection in images (M. Li, Wang, and Ramanan 2020) and LiDAR (Han et al. 2020; Frossard et al. 2020). At the same time, the simultaneous localization and mapping (SLAM) community (Nardi et al. 2015; Davison 2018; Bujanca et al. 2019) has recently proposed extending SLAM evaluation beyond trajectory accuracy (Geiger et al. 2013), toward system-level metrics like latency, power usage, and computational cost.

3
Localization with Sparse Semantic Maps

3.1 Introduction

High-definition maps (HD maps) are a fundamental component of most self-driving cars, as they contain useful information about the static part of the environment. The locations of lanes, traffic lights, and crosswalks, as well as the associated traffic rules are often encoded in the maps. Such maps encode the prior knowledge about any scene the autonomous vehicle may encounter.

In order to be able to leverage HD maps, self-driving cars have to localize themselves with respect to the map. The accuracy requirements in localization are very strict, and only a few centimeters of error are typically tolerated in such safety-critical scenarios. As discussed extensively in Chapter 2.5, over the past few decades, a wide range of localization systems has been developed based on different sensors, ranging from GPS to visual place recognition and map matching.

To overcome the limitations of GPS and IMU, place recognition techniques have been developed. These approaches store what the world looks like either in terms of geometry (e.g., LiDAR point clouds), visual appearance (e.g., SIFT features, LiDAR intensity), or semantics (e.g., semantic point cloud), and formulate localization as a retrieval task. Extensions of classical methods such as iterative closest point (ICP) are typically employed for geometry-based localization (Yoneda et al. 2014; Aghili and Su 2016). Unfortunately, geometric approaches suffer in the presence of repetitive patterns that arise frequently in scenarios such as highways, tunnels, and bridges. Visual recognition approaches (Cummins and Newman 2008) pre-record the scene and encode the "landmark'' visual features. They then perform localization by matching perceived landmarks to stored ones. However, they often require capturing the same environment for multiple seasons and times of the day. Recent work (Schönberger et al. 2018) builds dense semantic maps of the environment and combines both

semantics and geometry to conduct localization. However, this method requires a large amount of dense map storage and cannot achieve centimeter-level accuracy.

While place recognition approaches are typically fairly accurate, the costs associated with ensuring the stored representations are up to date can often be prohibitive. They also require substantial amounts of storage on board. Several approaches have been proposed to provide affordable solutions to localization by exploiting coarse maps that are freely available on the web (Brubaker, Geiger, and Urtasun 2013; W.-C. Ma et al. 2017). Despite demonstrating promising results, the accuracy of such methods is still in the order of a few meters, which does not meet the requirements of safety-critical applications such as autonomous driving.

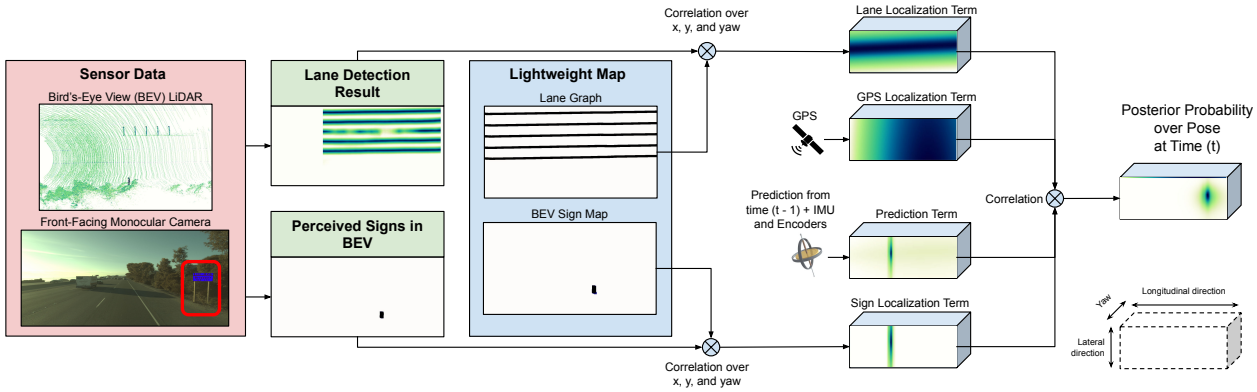


Figure 3.1: **System architecture.** Given the camera image and LiDAR sweep as input, we first detect lanes in the form of a truncated inverse distance field, and detect signs as a bird's-eye view (BEV) probability map. The detection output is then passed through a differentiable rigid transform layer (Jaderberg et al. 2015) under multiple rotational angles. Finally, the inner-product score is measured between the inferred semantics and the map. The probability score is merged with GPS and vehicle dynamics observations, and the inferred pose is computed from the posterior using soft-argmax. The camera image on the left contains an example of a sign used in localization, highlighted with the red box.

With these challenges in mind, in this chapter, we propose a lightweight localization method that does not require detailed knowledge of the appearance of the world (e.g., dense geometry or texture). Instead, we exploit a semantic map containing lane graphs and the locations of traffic signs together with vehicle dynamics. Traffic signs provide information in the longitudinal direction, while lanes help avoid lateral and angular drift. These cues are complementary to each other, and the resulting maps can be stored in a fraction of the memory necessary for dense HD maps, which is important as self-driving cars need to operate in large environments such as entire metropolitan areas.

Following the framework presented in Chapter 2.4.2, we formulate the localization problem as a Bayes filter which leverages observations based on matching online semantic cues to semantic information encoded in the map. We demonstrate the effectiveness of our approach on North American highways, which are challenging for current place recognition approaches as repetitive patterns are common and driving speeds are high. Our experiments on more than 300 km of testing trips showcase that we are able to achieve 0.05m median lateral ac-

curacy and 1.12m median longitudinal accuracy, while using roughly three orders of magnitude less storage than previous map-based approaches (0.55MiB/km^2 vs. the 1.4GiB/km^2 required for dense point clouds).

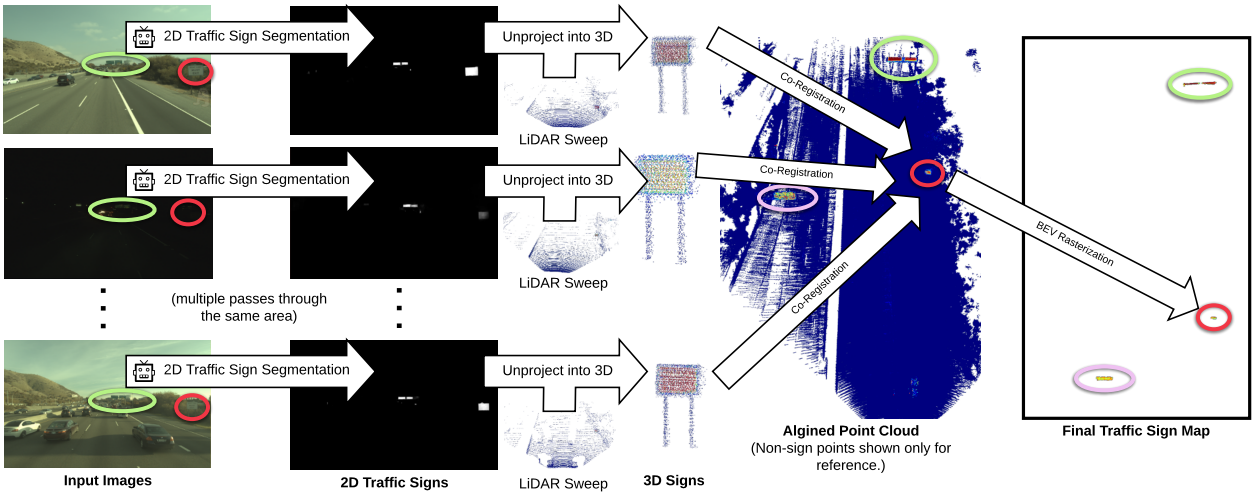


Figure 3.2: **The process used to construct our traffic sign maps.** We first detect signs in 2D using semantic segmentation in the camera frame and then use the LiDAR points to localize the signs in 3D. Mapping can aggregate information from multiple passes through the same area using the ground truth pose information and can function even in low light, as highlighted in the middle row, where the signs are correctly segmented even at night time. We use this information to build the traffic sign map automatically.

3.2 Lightweight HD Mapping

In order to conduct efficient and accurate localization, a compressed yet informative representation of the world needs to be constructed. Ideally, our HD maps should be easy to (automatically) build and maintain at scale, while also enabling real-time broadcasting of map changes between a central server and the fleet. This places stringent storage requirements that traditional dense HD maps fail to satisfy.

In this chapter, we tackle these challenges by building sparse HD maps containing just the lane graph and the locations of traffic signs. These modalities provide complementary semantic cues for localization. Notably, the storage needs for our maps are three orders of magnitude smaller than traditional LiDAR intensity maps (Levinson and Thrun 2010; Wolcott and Eustice 2015; Levinson, Montemerlo, and Thrun 2007) or 3D geometric maps (Yoneda et al. 2014).

Lane Graph. Most roads have visually distinctive lanes that determine the expected trajectory of vehicles and are compliant with the traffic rules. Most self-driving cars store this prior knowledge as lane graphs $\mathcal{L}$.

A lane graph is a structured representation of the road network defined as a set of polygonal chains (polylines), each of which represents a lane boundary. We refer the reader to Fig. 3.1 for an illustration of

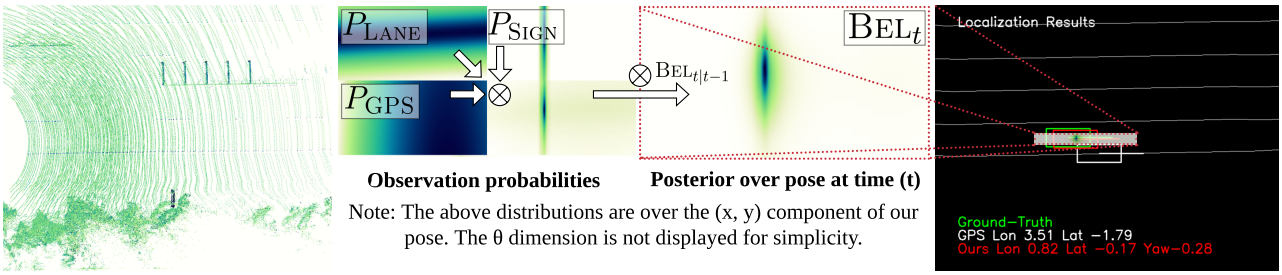


Figure 3.3: **Qualitative results.** A bird's-eye view of the last five LiDAR sweeps (left), which are used for the lane detection, together with the observation probabilities and the posterior (middle), followed by a comparison between the localization result, the ground truth pose, and GPS (right). The (x, y)-resolution of each probability distribution is 1.5m laterally (vertical) and 15m longitudinally (horizontal).

a lane graph. Lane graphs provide useful cues for localization, particularly in the lateral position and the vehicle heading.

Traffic Signs. Traffic signs are common semantic landmarks which are sparsely yet systematically present in cities, rural areas, and highways. Their presence provides useful cues that can be employed for accurate longitudinal localization. Our method relies on automatically built sparse HD maps that include traffic signs. Toward this goal, we exploit multiple passes of our vehicles over the same region and identify the signs by exploiting image-based semantic segmentation followed by 3D sign localization using LiDAR via unprojection from pixel to 3D space.

Note that in our map, we only store points that are estimated to be traffic signs above a certain confidence level. After that, we rasterize the sparse points to create the traffic sign presence probability map $\mathcal{T}$ in bird's-eye view (BEV) at 5cm per pixel. This is a very sparse representation containing all the traffic signs. The full process is conducted without any human intervention. Fig. 3.2 depicts the traffic sign map-building process and an example of its output.

3.3 Localization as Bayes Inference with Deep Semantics

We follow the histogram filtering approach introduced in Chapter 2.4.2 to localize against lightweight HD maps.

This family of methods performs map-based localization in the framework of Bayesian filtering, using either a histogram or particles to represent the state distribution. The key to both approaches is observation modeling: given the known map, an observation, and a pose *hypothesis* (a particle in the case of particle filtering, or a cell in the histogram case), any algorithm from this family must determine the likelihood that the current observation was made at that hypothesis.

3.3.1 Probabilistic Pose Filter Formulation

Our localization system exploits a wide variety of sensors: GPS, IMU, wheel encoders, LiDAR, and cameras. These sensors are available in most self-driving vehicles. The GPS provides a coarse location with several meters of accuracy; an IMU captures vehicle dynamic measurements; the wheel encoders measure the total travel distance; the LiDAR accurately perceives the geometry of the surrounding area through a sparse point cloud; images capture dense and rich appearance information. We assume our sensors are calibrated and neglect the effects of suspension, unbalanced tires, and vibration. As shown in our experiments, the influence of these factors is negligible and other aspects such as sloped roads (e.g., on highway ramps), do not have an impact on our localizer. Therefore, the vehicle's pose can be parametrized with only three degrees of freedom (instead of six) consisting of a 2D translation and a heading angle w.r.t. the map coordinate's origin, i.e., $\mathbf{x} = \{\mathbf{t}, \theta\}$, where $\mathbf{t} \in \mathbb{R}^2$ and $\theta \in (-\pi, \pi]$, since the heading is parallel to the ground plane.

Following the framework discussed Chapter 2.4.2, we factorize the posterior distribution over the vehicle pose into components corresponding to each modality, as shown in Eq. (3.2).

Let $\mathcal{G}_t$ be the GPS readings at time t and let $\mathcal{L}$ and $\mathcal{T}$ represent the lane graph and traffic sign maps respectively, with $\mathcal{M} = \{\mathcal{L}, \mathcal{T}\}$ encompassing all map data. We estimate the vehicle dynamics $\mathcal{X}_t$ from both IMU and the wheel encoders smoothed through an extended Kalman filter, which is updated at 100Hz. This component is separate from the main localizer, and its running estimate is treated as an input, following a loosely coupled philosophy.

The localization task is formulated as a histogram filter aiming to maximize the agreement between the observed and mapped lane graphs and traffic signs while respecting vehicle dynamics. Following Eq. (2.6), we have

$$\text{Bel}_t(\mathbf{x}) \;=\; \eta \cdot P_{\text{obs}}(\mathcal{Z}_t|\mathbf{x}; \mathcal{M}, \mathbf{w})\text{Bel}_{t|t-1}(\mathbf{x}|\mathcal{X}_t), \tag{3.1}$$

where in the current implementation, P_{obs} encompasses three components: two BEV terms based on Eq. (2.10), one for the lanes and one for the signs, and one GPS term based on Eq. (2.9).

Putting it together, this results in the following equation for computing the posterior over the pose at time t:

$$\begin{aligned}\text{Bel}_t(\mathbf{x}) \;= \eta \cdot\; & P_{\text{Lane}}(\mathcal{Z}_t|\mathbf{x}; \mathcal{L}, \mathbf{w}_{\text{Lane}})P_{\text{Sign}}(\mathcal{Z}_t|\mathbf{x}; \mathcal{T}, \mathbf{w}_{\text{Sign}}) \\ & P_{\text{GPS}}(\mathcal{G}_t|\mathbf{x})\text{Bel}_{t|t-1}(\mathbf{x}|\mathcal{X}_t).\end{aligned} \tag{3.2}$$

$\text{Bel}_t(\mathbf{x})$ is the posterior probability of the vehicle pose at time t; η is a normalizing factor to ensure the sum of all probability is equal

to one; $\mathbf{w} = \{\mathbf{w}_{\text{Lane}}, \mathbf{w}_{\text{Sign}}\}$ are sets of learnable parameters, and $\mathcal{Z}_t = \{\mathcal{I}_t, \mathcal{C}_t\}$ is the sensory measurement tuple composed of LiDAR $\mathcal{I}_t$ and camera $\mathcal{C}_t$. This probabilistic equation is depicted visually in Fig. 3.3. Note that by recursively solving Eq. (3.2), we can localize the vehicle at every step with an uncertainty measure that can propagate to the next step. We now describe each new energy term in more detail. Please refer to Chapter 2.4.2 for descriptions of P_{GPS} and $\text{Bel}_{t|t-1}(\cdot)$.

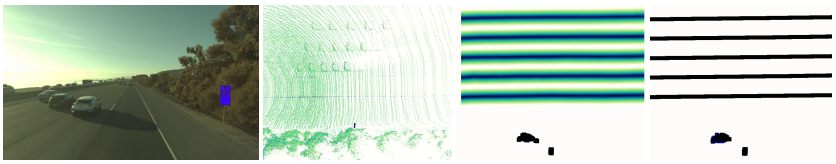


Figure 3.4: **Dataset sample and inference results.** Our system detects signs in the camera images (note the blue rectangle on the right side of the first image) and projects the sign's points in a top-down view using LiDAR (second image). It uses this result in conjunction with the lane detection result (third image) to localize against a lightweight map consisting of just signs and lane boundaries (fourth image).

Lane Observation Model. We define our matching energy to encode the agreement between the lane observation from the sensory input and the map. Our probability is computed by a normalized matching score function that utilizes the existing lane graph and compares it to detected lanes. To detect lanes we use a real-time multi-sensor convolutional network (Bai et al. 2018) which was state-of-the-art at the time of publication. The inputs to the network are a front-view camera image and the raw LiDAR intensity measurements projected onto BEV. The network output is the inverse truncated distance function to the lane graph in the overhead view. Specifically, each pixel in the overhead view encodes the Euclidean distance to the closest lane marker, up to a truncation threshold of 1m. We refer the reader to Fig. 3.4 for an illustration of the neural network's input and output.

To compute the probability, we first orthographically project the lane graph $\mathcal{L}$ onto overhead view such that the lane detection output and the map are under the same coordinate system. The overhead view of the lane graph is also represented using a truncated inverse distance function. Given a vehicle pose hypothesis $\mathbf{x}$, we rotate and translate the lane detection prediction accordingly and compute its matching score against the lane graph map. The matching score is an inner product between the lane detection and the lane graph map

$$P_{\text{Lane}} \propto s\left(\pi\left(f_{\text{Lane}}(\mathcal{Z}_t; \mathbf{w}_{\text{Lane}}), \mathbf{x}\right), \mathcal{L}\right), \tag{3.3}$$

where f_{Lane} is the deep lane detection network and $\mathbf{w}_{\text{Lane}}$ are the network's parameters. π is a 2D rigid transform function to transform the online lane detection to the map's coordinate system given a pose hypothesis $\mathbf{x}$; $s(\cdot, \cdot)$ is a cross-correlation operation between two images. Please refer to Chapter 2.4.3 for more details.

Traffic Sign Observation Model. This model encodes the consistency between perceived online traffic signs and the map. Specifically, we run an image-based semantic segmentation algorithm that performs dense semantic labeling of traffic signs. We adopt the PSPnet structure (Zhao et al. 2017) to our task, which was state-of-the-art at the time of publication. The encoder architecture is a ResNet50 backbone, and the decoder is a pyramid spatial pooling network. Two additional convolutional layers are added in the decoder stage to further boost performance. We train the model jointly with an instance segmentation loss following (Bai and Urtasun 2017). Fig. 3.4 depicts examples of the network input and output. The estimated image-based traffic sign probabilities are converted onto the overhead view to form our online traffic sign probability map. This is achieved by associating each LiDAR with a pixel in the image by projection. We then read the softmax probability of the pixel's segmentation as our estimate. Only high-confidence traffic sign pixels are unprojected to 3D and rasterized in BEV. Given a pose proposal $\mathbf{x}$, we define the sign-matching probability analogously to the lane-matching one as

$$P_{\text{Sign}} \propto s\left(\pi\left(f_{\text{Sign}}(\mathcal{Z}_t; \mathbf{w}_{\text{Sign}}), \mathbf{x}\right), \mathcal{T}\right), \tag{3.4}$$

where f_{Sign} is the sign segmentation network and $\mathbf{w}_{\text{Sign}}$ are the networks parameters. Both the perceived signs and the map to which they are matched are encoded as pixel-wise occupancy probabilities.

3.3.2 Inference and Learning

Inference. Inference is performed in real-time following the framework previously covered in Chapter 2.4.4.

Method	GPS std	Dyn Angle std	Dyn Lat std	Dyn Lon std	Argmax Temp	Lane Temp	Sign Temp
Lane	-	1.0	2.0	5.0	1.0	1.0	-
Lane+GPS	20.0	1.0	2.0	5.0	1.0	1.0	-
Lane+Sign	-	1.0	2.0	5.0	1.0	1.0	2.5
All	20.0	1.0	2.0	5.0	1.0	1.0	2.5

Figure 3.5: Best hyperparameters for each method

Learning. The lane detection and the traffic sign segmentation networks are trained separately through back-propagation using ground-truth annotated data. The lane detection is trained with a regression loss that measures the ℓ_2 distance between the predicted inverse truncated distance transform and the ground-truth one (Bai et al. 2018). The semantic segmentation network is trained with cross-entropy (Bai and Urtasun 2017). Hyperparameters for the Bayes filter

(e.g., σ^2_{GPS}, softmax temperature α, etc.) are searched through cross-validation.

3.4 Experimental Evaluation

We validate the effectiveness of our localization system on a highway dataset of 312 km. We evaluate our model in terms of its localization accuracy and runtime.

3.4.1 Dataset

Our goal is to perform fine-grained localization on highways. Unfortunately, at the time of publication, there were no publicly available datasets which provided ground truth localization at the centimeter-level precision required for safe autonomous driving. We therefore collected a dataset of highways by driving over 300km in North America at different times of the year, covering over 100km of roads. The dataset encompasses 64-beam LiDAR sweeps and images from a front-facing global shutter camera with a resolution of 1900×1280, both captured at 10Hz, as well as IMU and GPS sensory data. The dataset also includes a lane graph map. The extrinsic calibration between the camera and LiDAR is conducted using a set of calibration targets (Hartley and Zisserman 2003). The ground truth 3D localization is estimated by a high-precision ICP-based offline Graph-SLAM using high-definition pre-scanned scene geometry. Fig. 3.4 shows a sample from our dataset together with the inferred and ground truth lane graphs.

Our dataset is partitioned into 'snippets,' each consisting of roughly 2km of driving. The training, validation, and test splits are conducted at the snippet level, where training snippets are used for map building and training the lane detection network, and validation snippets are used for hyperparameter tuning. The test snippets are used to compute the final metrics. An additional 5,000 images are annotated with pixel-wise traffic sign labels which are used to train the sign segmentation network.

3.4.2 Implementation Details

Network Training. To train the lane detection network, we uniformly sample 50K frames from the training region based on their geographic coordinates. The ground truth can be generated automatically, given the vehicle pose and the lane graph. We use a mini-batch size of 16 and employ Adam (Kingma and Ba 2015) as the optimizer. We set the learning rate to 10^{-4}. The network was trained entirely from scratch with Gaussian initialization and converged roughly after

Method	Properties			Travelling Dist = $2km$					
				Longitudinal Error (m)			Lateral Error (m)		
	Lane	GPS	Sign	Median	95%	99%	Median	95%	99%
Lane	yes	no	no	13.45	37.86	51.59	0.20	1.08	1.59
Lane+GPS	yes	yes	no	1.53	5.95	6.27	0.06	0.24	0.43
Lane+Sign	yes	no	yes	6.23	31.98	51.70	0.10	0.85	1.41
All	yes	yes	yes	**1.12**	**3.55**	**5.92**	**0.05**	**0.18**	**0.23**

Table 3.1: Ablation study on the impact of each component of our lightweight localization system.

ten epochs. We visualize some results in Fig. 3.4.

We train our traffic sign segmentation network separately over four GPUs with a total mini-batch size of 8. Synchronized batch normalization is utilized for multi-GPU batch normalization. The learning rate is set to be 10^{-4} and the network is trained from scratch. The backbone of the model is fine-tuned from a DeepLab v2 network pre-trained over the Pascal VOC dataset.

Hyper-parameter Search. We choose the hyperparameters through grid search over a mini-validation dataset consists of 20 snippets of 2km driving. The hyperparameters include the temperatures of the final pose soft-argmax, the lane probability softmax, and the sign probability softmax, as well as the observation noise parameters for GPS and the dynamics. The best configuration is chosen by the failure rate metric. In the context of hyperparameter search, the failure rate is a snippet-level metric which counts a test snippet as failed if the total error becomes greater than 1m at any point. We therefore picked the hyperparameter configuration which minimized this metric on our validation set, and kept it fixed at test time. As noted in Chapter 2.4.2, we restrict our search range to a small area centered at the dead reckoning pose and neglect the probability outside the region. We notice in practice that thanks to the consistent presence of the lanes in self-driving scenarios, there is less uncertainty along the lateral direction than along the longitudinal. The presence of traffic signs helps reduce uncertainty along the longitudinal direction, but signs could be as sparse as every 1km, during which INS drift could be as large as 7 meters. Based on this observation and with the potential drift in mind, we choose a very conservative search range $B = B_x \times B_y \times B_\theta = [-0.75m, 0.75m] \times [-7.5m, 7.5m] \times [-2°, 2°]$ at a spatial resolution of 5cm and an angular resolution of 1°.

We performed an exhaustive grid search to find the set of hyperparameters that minimize the failure rate in the mini-validation set. The search grid consisted of soft-argmax temperature from the set $\{1., 8.\}$, lane temperature from $\{.5, 1., 1.5\}$, sign temperature from $\{.5, 1., 1.5\}$,

GPS observation's standard deviation at $\{0, 10, 20\}$ meters, dynamics observation's standard deviation from $\{1., 2.\}$ degrees, and dynamics observation's longitudinal standard deviation from $\{5., 10.\}$ meters. This describes a grid of 216 points for which metrics were computed exhaustively. The best set of hyperparameters can be seen in Tab. 3.5.

3.4.3 Localization

Metrics. We adopt several key metrics to measure the localization performance of the algorithms evaluated in this Section.

Method Name	Smoothness			
	Mean	95%	99%	Max
Dynamics	0.2	0.4	0.6	1.2
GPS	**0.1**	0.2	0.3	8.5
INS	**0.1**	**0.1**	**0.2**	3.7
Ours	**0.1**	0.2	0.3	**0.9**

Table 3.2: Quantitative results for lightweight localization using smoothness metrics.

In order to safely drive from a certain point to another without any human intervention, an autonomous vehicle must be aware of where it is w.r.t. the map. Lateral error and longitudinal error have different meanings for self-driving since a small lateral error could result in localizing in the wrong lane, while ambiguities about the longitudinal position of the vehicle are more tolerable. As localization is the first stage in self-driving pipeline, it is critical that it stays robust enough with a minimal failure rate; therefore, understanding worst-case performance is critical.

Moreover, localization results should reflect the vehicle dynamics as well, which ensures the smoothness of decision making since sudden jumps in localization might cause downstream components to fail. To this end, we also measure the prediction smoothness of our methods. We define smoothness as the difference between the temporal gradient of the ground truth pose and that of the predicted pose. We estimate the gradients using first-order finite differences, i.e., by simply taking the differences between poses at times (t) and $(t-1)$. As such, we define smoothness as

$$s = \frac{1}{T}\sum_{t=1}^{T} \left\| (\mathbf{x}_t^* - \mathbf{x}_{t-1}^*) - (\mathbf{x}_t^{\mathrm{GT}} - \mathbf{x}_{t-1}^{\mathrm{GT}}) \right\|^2 . \tag{3.5}$$

Baselines. We compare our results with two baselines: dynamics and dynamics+GPS. The first baseline builds on top of the dynamics of the vehicle. It takes as input the IMU data and wheel odometry, and use the measurements to extrapolate the vehicle's motion. The second baseline employs histogram filters to fuse information between

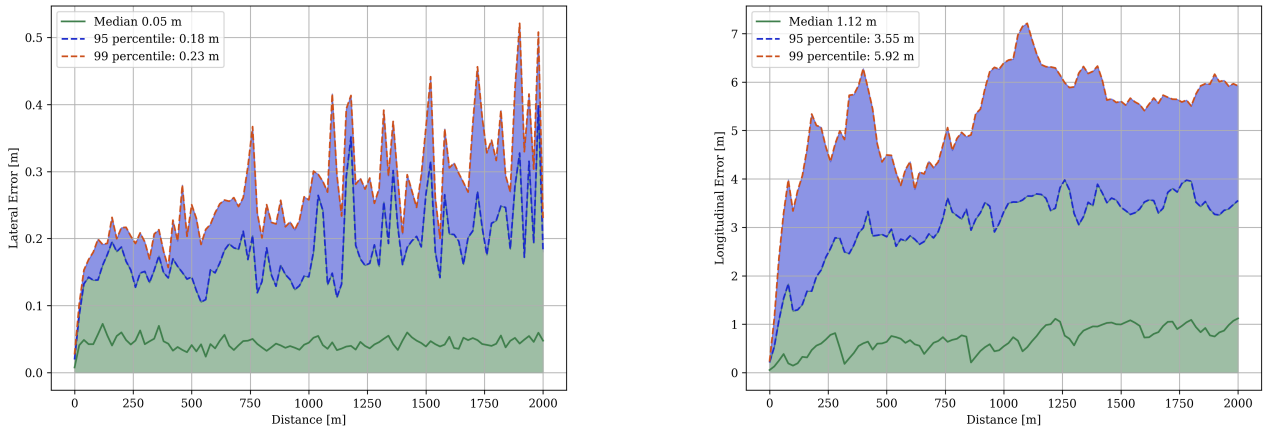


Figure 3.6: Localization Error as a function of travel distance.

IMU readings and GPS sensory input, which combines motion and absolute position cues.

Methods	Longitudinal Error (m)			Lateral Error (m)		
	Median	95%	99%	Median	95%	99%
Dynamics	24.85	128.21	310.50	114.46	779.33	784.22
GPS	1.16	5.78	6.76	1.25	8.56	9.44
INS	1.59	6.89	13.62	2.34	11.02	42.34
Ours	**1.12**	**3.55**	**5.92**	**0.05**	**0.18**	**0.23**

Table 3.3: Quantitative results on localization accuracy. Here, 'Ours' refers to the model proposed in this paper using dynamics, GPS, lanes, and signs, in a probabilistic framework.

Quantitative Analysis. As shown in Tables 3.2 and 3.3, our method significantly outperforms the baselines across all metrics. To be more specific, our model has a median longitudinal error of 1.12m and a median lateral error of 0.05m; both are much smaller than other competing methods, with lateral error one order of magnitude lower. Our method greatly improves performance in worst-case scenarios in terms of longitudinal error, lateral error, and smoothness.

Error vs. Travel Distance. Fig. 3.6 depicts the localization error as a function of travel distance. We can see that our approach is relatively stable across travel distance without catastrophic failures. Particularly, the median lateral error is maintained to be around 5cm with a worst-case of around 23cm.

Qualitative Results. We show the localization results of our system as well as those of the baselines in Fig. 3.3. Through lane observations, our model is able to consistently achieve centimeter-level lateral localization accuracy. When signs are visible, the traffic sign model helps push the prediction toward the location where the observation and map have agreement, bringing the pose estimate to

Inference	Travelling Dist = $2km$						Smoothness			
	Longitudinal Error (m)			Lateral Error (m)						
	Median	95%	99%	Median	95%	99%	Mean	95%	99%	Max
Deterministic	1.29	3.65	**5.16**	0.08	0.26	0.50	0.11	**0.19**	1.78	5.27
Probabilistic	**1.12**	**3.55**	5.92	**0.05**	**0.18**	**0.23**	**0.07**	**0.19**	**0.24**	**0.98**

Table 3.4: Ablation studies on inference settings with full observations (Lane+GPS+Sign)

the correct longitudinal position. In contrast, GPS tends to produce noisy results, but helps substantially improve worst-case performance.

Runtime Analysis. To further demonstrate that our localization system is of practical usage, we benchmark the runtime of each component in the model during inference using an NVIDIA GTX 1080 GPU. A single step of our inference takes 153ms in total on average, with 32ms on lane detection, 110ms on semantic segmentation and 11ms on matching, which is roughly 7 FPS. We note that the real-time performance is made possible largely by the FFT convolutions discussed in Chapter 2.4.4.

Map Storage Analysis. We compare the size of our HD map against other commonly used representations: LiDAR intensity map and 3D point cloud map. For a fair comparison, we store all data losslessly and measure the storage requirements. While the LiDAR intensity and 3D point cloud maps consume 177 MiB/km^2 and 1,447 MiB/km^2, respectively, our HD map only requires **0.55 MiB** per square kilometer. This is only 0.3% of the size of the LiDAR intensity map and 0.03% of that of a 3D point cloud map.

Ablation Study. To better understand the contribution of each component of our model, we respectively compute the longitudinal and lateral error under diverse settings. As shown in Tab. 3.1, each term (GPS, lane, sign) has a positive contribution to the localization performance. Specifically, the lane observation model greatly increases lateral accuracy, while sign observations increase longitudinal accuracy. We also compare our probabilistic histogram filter formulation with a deterministic model. Compared to our histogram filter approach, the non-probabilistic one performs a weighted average between each observation without carrying over the uncertainty of the previous step. As shown in Tab. 3.4, by combining all the observation models, the non-probabilistic model can achieve reasonable performance but still remains less accurate than the probabilistic formulation. Moreover, due to the fact that no uncertainty history is carried over, prediction smoothness over time is not guaranteed.

3.5 Conclusion

In this chapter we proposed a system capable of robustly localizing an autonomous vehicle against a map, while requiring roughly three orders of magnitude less storage than traditional methods. This has the potential to substantially improve the scalability of self-driving technologies by reducing storage costs, while also enabling map updates to be delivered to vehicles in real-time over inexpensive mobile networks.

We approached the task by identifying two sets of complementary cues capable of disambiguating the lateral and longitudinal position of the vehicle: lane boundaries and traffic signs. We integrated these cues into a pipeline alongside GPS, IMU, and wheel encoders and showed that the system is able to run in real-time at roughly 7 Hz on a single GPU. We demonstrated the efficacy of our method on a large-scale highway dataset consisting of over 300km of driving, showing that it can achieve the localization accuracy requirements of self-driving cars while using much less storage.

However, the accuracy of this method still relies on perception models trained without localization in mind. In the absence of cues, this approach can fail. By training representations specifically for the task of localization, we can simplify the training procedure while substantially increasing the system's robustness. This is the subject of the following few chapters.

4
LiDAR Matching with Deep Representations

4.1 Introduction

Lightweight maps like those leveraged previously in Chapter 3 draw attention to developing affordable localization efforts. While only requiring small amounts of storage, they encode both the road network's topological structure and its semantics. The primary advantage of localizing with lightweight maps is the fact that this map information is typically already present onboard due to its use in tasks like routing and motion planning.

However, as highlighted in Chapter 3.4, these methods still struggle to achieve centimeter-level accuracy. Furthermore, such localizers rely on the presence of the human-selected cues in the map, and their accurate online detection. This adds another layer of complexity, limiting the localizer's applicability to environments where semantic maps may be unavailable or the cues difficult to perceive.

In contrast, localizers which directly leverage the online LiDAR data and map imagery can bypass these limitations. While techniques from this area have been shown to work well in practice (Levinson, Montemerlo, and Thrun 2007; Levinson and Thrun 2010; Wolcott and Eustice 2015), outperforming methods based on geometric registration (Yoneda et al. 2014), as well as GNSS, visual localization, and lightweight localization, a few notable limitations remain. The approaches we will cover in this and the next chapters aim to address them.

First, LiDAR intensity readings are difficult to calibrate, especially when mapping and localization use different LiDAR units, possibly from different manufacturers. In order for localization to scale to large maps and to avoid having to re-build maps when new vehicles are deployed, intensity-based localization must be agnostic to calibration issues.

Second, dense LiDAR intensity maps, often stored at resolutions as high as 5cm/px can be demanding in terms of transmission and storage, limiting their deployment at scale.

This chapter proposes to learn to extract representations which are optimal for localization directly from the map imagery and the online observations. We propose to learn a BEV affinity function between observations and map data optimized specifically for localization qual-

ity. This bypasses the need to explicitly extract semantic cues while avoiding the challenges typically associated with LiDAR intensity calibration. Later, in Chapter 5, we discuss ways of tackling the second limitation by integrating a data-driven task-specific compression scheme capable of reducing storage requirements for localization maps by several orders of magnitude.

We refer the reader Fig. 4.1 for a sample fragment of the maps used by our system.

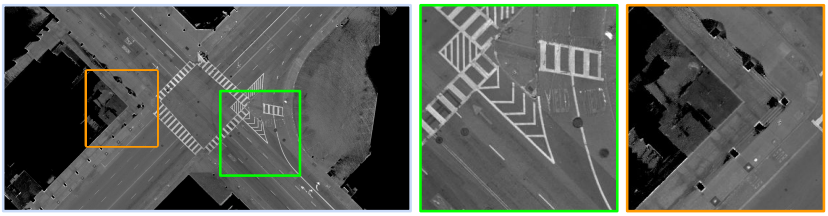


Figure 4.1: **LiDAR intensity map.** An example of a bird's-eye view (BEV) LiDAR intensity map used by our system. It encodes rich information on the environment's appearance and its geometric structure. The orange square highlights an example of geometric structure captured by the BEV images—the corner of a building, while the green one highlights an example of intensity structure—painted lane lines and crosswalks.

4.2 Learning LiDAR Representations for Localization

In this section, we discuss our LiDAR intensity localization system. Following the same framework as in the previous chapter, we formulate localization as a deep recursive Bayesian estimation problem and discuss each probabilistic term. We then present our real-time inference algorithm and describe how our model is trained.

LiDAR Matching Energy. This term measures the probability of the current LiDAR observation at a potential vehicle pose $\mathbf{x}$. It projects the map centered at $\mathbf{x}$ and the LiDAR observation into a shared space and measures their affinity using cross-correlation. This term is expressed in BEV, taking the place of the P_{BEV} term discussed in Eq. (2.10).

We can express deep LiDAR matching as an instantiation of this equation by passing a rasterized bird's-eye view image of the current LiDAR as $\mathcal{Z}_t$, and implementing f_o and f_m as fully convolutional neural networks.

Network Architecture. Our embedding functions $f_o(\,\cdot\,;\mathbf{w}_{\text{o}})$ and $f_m(\,\cdot\,;\mathbf{w}_{\text{m}})$ are customized fully convolutional neural networks. The first network $f_o(\,\cdot\,;\mathbf{w}_{\text{o}})$ takes as input the bird's-eye view (BEV) rasterized image of the k most recent LiDAR sweeps (compensated by ego-motion) and produces a dense representation at the same resolution as the input. The second network $f_m(\,\cdot\,;\mathbf{w}_{\text{m}})$ takes as input a section of the LiDAR intensity map and produces an embedding with the same number of channels as the first one, and the spatial resolution of the map.

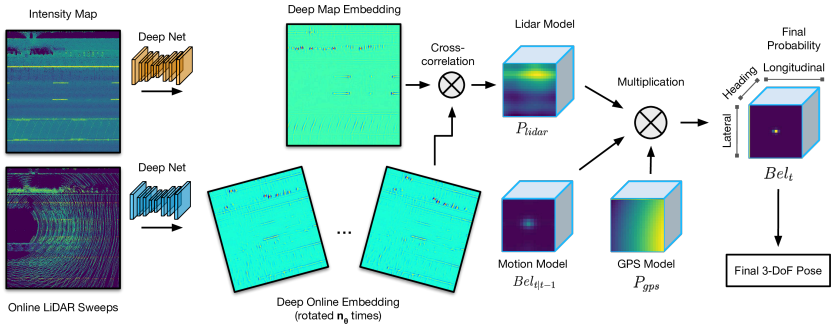
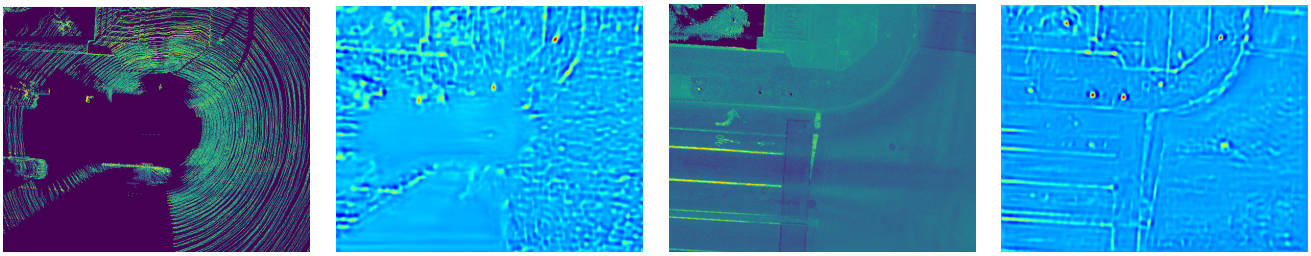


Figure 4.2: **Deep LiDAR localizer architecture.** The full architecture of the proposed localizer, which incorporates our learned LiDAR matching component and outputs a 3-DoF pose at each time step. The top Deep Net is f_m, while the bottom represents f_o.



(a) Online LiDAR image. (b) Online embedding. (c) Intensity map. (d) Map embedding.

Figure 4.3: **One example of the learned input and map embeddings.** The neural networks learn to focus on reliable cues while suppressing unreliable ones and dynamic objects.

We experiment with architectures based on the patch-matching architecture used by (W. Luo, Schwing, and Urtasun 2016) and with LinkNet by (Chaurasia and Culurciello 2017).

We use instance normalization (Ulyanov, Vedaldi, and Lempitsky 2017) after each convolutional layer instead of batch normalization due to its ability to reduce instance-specific mean and covariance shift. Our embedding output has the same resolution as the input image, with a (potentially) multi-dimensional embedding per pixel. The channel dimension for the output embeddings is chosen based on the trade-off between performance and runtime. We refer the reader to Fig. 4.3 for an illustration of a single-channel embedding. Unless otherwise stated, all our experiments use single-channel embeddings for both online LiDAR and the maps, as we found them to be optimal empirically.

By leveraging the now-familiar 3-DoF matching described in Chapter 2.4, we only need to run the embedding networks once, rotate the computed online LiDAR embedding n_θ times, and convolve each rotation with the map embedding to get the probability for all the pose hypotheses in the form of a score map S. Our solution is, therefore, globally optimal over our discretized search space including both rotation and translation. In practice, the rotation of our online LiDAR embedding is implemented using a spatial transformer module (Jaderberg et al. 2015), and generating all rotations takes 5ms in total (we

use $n_\theta = 5$ in all our experiments). A point estimate of the posterior pose can be estimated with the soft-argmax procedure described in Chapter 2.4.4.

4.2.1 Learning

The proposed LiDAR matching system is end-to-end differentiable, enabling us to learn all parameters jointly using backpropagation. We find that a simple cross-entropy loss is sufficient to train the system, without requiring additional, potentially expensive terms, such as a reconstruction loss. We define the cross-entropy loss between the ground-truth position and the inferred score map as

$$\mathcal{L}_{\text{Match}}(\mathbf{y}, \mathbf{y}^{(\text{GT})}) = \sum_{i \in \{0,1\}} \mathbf{y}^{(\text{GT})} \log(\mathbf{y}_i) \tag{4.1}$$

where the $\mathbf{y}$ corresponds to the network's prediction, or P_{LiDAR} (post-softmax) from the above diagram. $\mathbf{y}^{(\text{GT})}$ represents the one-hot encoding of the ground truth offset between the online LiDAR and the intensity map. It therefore consists of all zeroes, except for a single 'one' at the location of the correct (x, y, yaw) offset[1].

[1] We also experimented with an ℓ_2 loss between the soft-argmax output of P_{LiDAR} and the ground truth offset triplet, e.g., as proposed by the authors of GC-Net (Kendall et al. 2017), but we found cross-entropy to consistently work better in practice. We also tried various forms of ground truth label smoothing, but they did not improve results over the one-hot encoding either.

4.3 Experimental Results

Dataset. We collected a new dataset comprising over 4,000km of driving through a variety of urban and highway environments in multiple cities/states in North America, collected with two types of LiDAR sensors. According to the scenarios, we split our dataset into *Highway-LidarA* and *Misc-LidarB*, where *Highway-LidarA* contains over 400 sequences for a total of over 3,000km of driving for training and validation. We select a representative and challenging subset of 282km of driving for testing, ensuring that there is no geographic overlap between the splits. All these sequences are collected by a LiDAR type A. *Misc-LidarB* contains 79 sequences with 200km of driving over a mix of highway and city collected by a different LiDAR type B in a different state. LiDARs A and B differ substantially in their intensity output profiles, as shown in Fig. 4.4.

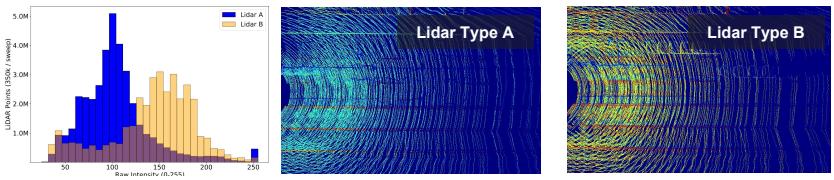
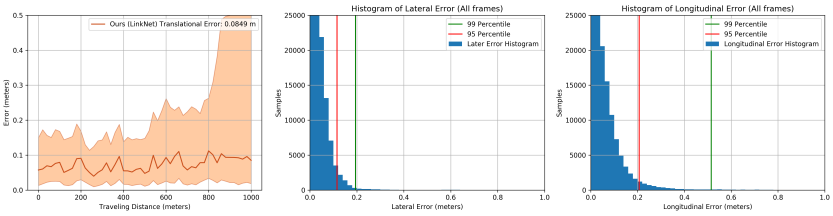


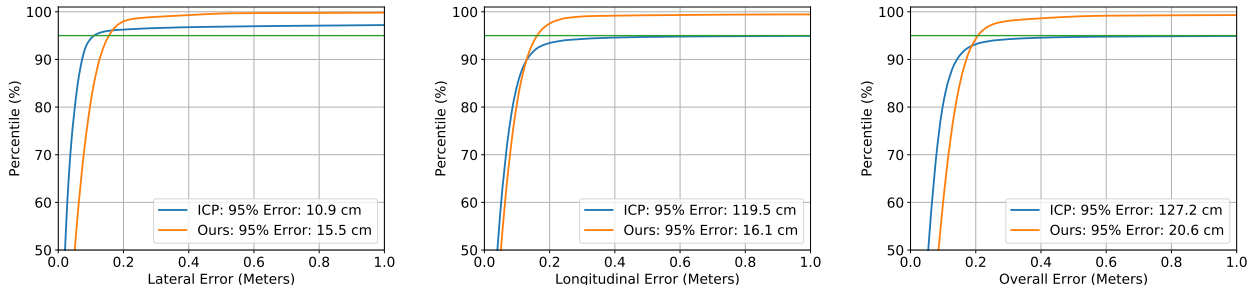
Figure 4.4: **LiDAR Sensor Transfer.** A comparison between the two LiDAR sensors. Left: the different intensity profiles of their sweeps over the same location; right: the color-mapped intensity images.

Experimental Setup. We randomly extracted 230k training samples from the training sequences. We aggregate the five most recent



Error vs. Traveling Dist Lateral Histogram Longitudinal Histogram

Figure 4.5: **Quantitative Analysis.** From left to right: localization error vs traveling distance; lateral error histogram per each timestamp; longitudinal histogram per each step.



Lateral Longitudinal Total Translational

Figure 4.6: **Cumulative error curve for the deep LiDAR localizer on *Highway-LidarA*.** From left to right: lateral, longitudinal, total translational error.

online LiDAR sweeps for each training sample to generate the BEV intensity image using vehicle dynamics, corresponding to 0.5 seconds of LiDAR data. In such a short time, drift is negligible. Our ground-truth poses are acquired through an expensive high-precision offline matching procedure with up to several centimeters of uncertainty. We rasterize the aggregated LiDAR points to create a LiDAR intensity image. Both the online intensity image and the intensity map are discretized at a spatial resolution of 5cm covering a $30m \times 24m$ region. During matching, we use the same spatial resolution, plus a rotational resolution of 0.5°, with a total search range of $1m \times 1m \times 2.5°$ around the dead reckoning pose[2]. We report the median error as well as the failure rate. The median error reflects how accurate the localization is in the majority of cases, while the failure rates reflect the worst-case performance. In particular, we define "failure" if at least one frame with localization error over 1m exists. In addition to these per-sequence metrics, we also plot the per-frame cumulative localization error curve in Fig. 4.6.

Implementation Details. We manually chose the following hyperparameters through validation, namely the motion model variance $\Sigma = \text{diag}([3.0, 3.0, 3.0])$, the GPS observation variance $\sigma^2_{GPS} = 10.0$, and the temperature constant $\alpha = 2.0$.

We implement our full inference algorithm in PyTorch 0.4. The networks are trained using Adam over four NVIDIA 1080Ti GPUs,

[2] Recall that in Chapter 3, we used an uneven search region which was much larger in the longitudinal (along-road) direction. This was because that method struggled to constrain the robot's state longitudinally due to the sparsity of cues that constrain this dimension (signs), especially in highway environments. As the current method can infer cues from the LiDAR map itself, it no longer has this limitation. We can, therefore, use equal search ranges laterally and longitudinally.

with an initial learning rate of 0.001.

We observe that incorporating the rotation component of the matcher in the training pipeline did not meaningfully improve matching performance. However, the added computation in the forward and backward passes slowed down training substantially. As a result, we trained our final models without matching in the yaw dimension. Even though we only trained using 2-DoF matching, the embeddings were able to generalize well to 3-DoF matching.

It is critical to cover all possible 3-DoF offsets within the method's (x, y, yaw) search range. Since ground truth poses are available, each dataset sample encompasses perfectly aligned online and map images. However, in this case, the ground truth offset would always correspond to $(0, 0, 0)$, i.e., the central pixel of the 3D score volume. We observed that this immediately caused overfitting, making the resulting embeddings useless.

To address this, at train time, we always perturb each sample with a random transform from our pre-defined search range. Each random transform can be one-hot encoded, and the ability to "jitter" the online LiDAR like this acts like data augmentation, producing many training samples from each ground-truth-aligned (map, LiDAR) pair.

Ablation studies. We also conduct two ablation studies. Our first ablation verifies whether the motion prior defined in Eq. (2.7) is helpful. We evaluate the algorithm with and without this term, denoted as Motion in Fig. 4.1. Our second ablation, also covered in this table, evaluates whether a probabilistic MLE proposed in Eq. (2.12) helps improve performance, denoted as Prob. The non-probabilistic option is achieved by changing the soft-argmax in Eq. (2.12) to a hard one.

Comparison to Other Methods. We compare our algorithm against several baselines. The raw matching consists of performing the matching-based localization in a manner similar to our method but only using the raw intensity BEV online and map images, instead of the learned embeddings. The ICP baseline conducts point-to-plane ICP between the raw 3D LiDAR points and the 3D pre-scanned localization map at 10Hz. It is initialized in a manner similar to our method, using the previously estimated location plus the vehicle dynamics. This ensures good initialization quality, as required by algorithms from the ICP family.

Localization Performance. As shown in Tab. 4.1, our approach achieves the best performance among all the competing algorithms in terms of failure rate. Both probabilistic inference and the use of a motion prior further improve the robustness of our method. Our ICP baseline is competitive in terms of median error, especially along the lateral direction, but the failure rate is significantly higher. It is also more computationally demanding and requires 3D maps. Both

Method	Motion	Prob	Median Error (cm) Lat	Lon	Total	Failure Rate (%) ≤ 100m	≤ 500m	≤ End
Dynamics	✓		439.21	863.68	1216.01	0.46	98.14	100.00
Raw LiDAR	✓		1245.13	590.43	1514.42	1.84	81.02	92.49
ICP	✓		**1.52**	5.04	**5.44**	3.50	5.03	7.14
Ours (LinkNet)			3.87	4.99	7.76	0.35	0.35	0.72
Ours (LinkNet)	✓		3.81	4.53	7.18	1.06	1.06	1.44
Ours (LinkNet)	✓	✓	3.00	**4.33**	6.47	**0.00**	**0.00**	**0.00**

Table 4.1: **Localization Performance on the Highway-LidarA dataset.** Please note that the numbers in this thesis and the arXiv version of the paper are more up-to-date than those in the CoRL proceedings as they incorporate a small bugfix.

dynamics-only and raw intensity matching result in substantial drift. Moreover, we have observed that deeper architectures and probabilistic inference are generally helpful. Fig. 4.5 shows the localization error as a function of the travel distance aggregated across all sequences from the *Highway-LidarA* test set. The solid line denotes the median, and the shaded region denotes the 95% area, together with the distribution of lateral and longitudinal errors per frame. Fig. 4.6 compares our approach to ICP in terms of cumulative errors with the 95th percentile error reported. From this, we can see that our method significantly outperforms ICP in terms of worst-case behavior.

Domain Shift. In order to show that our approach generalizes well across LiDAR sensors, we conduct a second experiment, where we train our network on *Highway-LidarA*, a dataset collected using LiDAR A, which consists purely of highway data, and test on the test set of *Misc-LidarB*, which is a dataset encompassing both highway and city driving, collected in a different state with a different LiDAR (type B). In order to better highlight the difference, in Fig. 4.4 we show the intensity value distributions of the two LiDAR types and their raw intensity images, collected at the same physical location[3]. Tab. 4.2 showcases the results of this experiment. The table highlights that our neural network can generalize both across LiDAR models and across environment types.

[3] The LiDAR A sample is not part of the dataset, as *Highway-LidarA* is completely geographically disjoint from *Misc-LidarB*.

Runtime Analysis. We conduct a runtime analysis over both embedding networks and matching. Our LinkNet-based embedding networks take less than 10ms each for a forward pass over the online and map images, respectively. We also compare the cuDNN implementation of FFT-conv and standard spatial convolution. FFT reduces the run time of the matching by an order of magnitude, bringing it down from 27 ms to 1.4 ms for a single-channel embedding. This enables us to run the localization algorithm at 15 Hz, achieving our real-time operation goal.

Emergent Behavior. By optimizing localization performance, we observed emergent effects in the neural networks used to compute

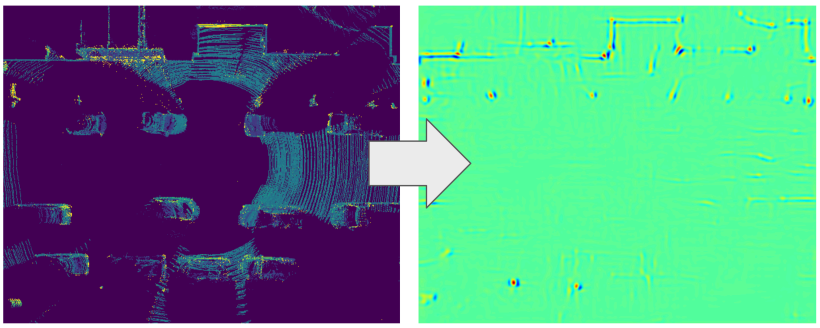


Figure 4.7: **Emergent Behavior in Deep LiDAR Matching.** We notice that when translating the raw LiDAR (left) into the deep embedding (right), the neural network learns to remove objects that are not reliable enough for localization, such as cars, despite never being explicitly trained to do so

Method	Motion	Prob	Median Error (cm)			Failure Rate (%)		
			Lat	Lon	Total	$\leq$ 100m	$\leq$ 500m	$\leq$ End
Dynamics Only	Yes	No	195.73	322.31	468.53	6.13	68.66	84.26
ICP	Yes	No	**2.57**	15.29	16.42	0.46	28.43	37.53
Ours (Transfer)	Yes	No	6.95	**6.38**	**11.73**	**0.00**	**0.71**	**1.95**

Table 4.2: **Cross-dataset generalization.** Localization Performance on Misc-LidarB trained on Highway-LidarA.

the deep embeddings. We can see an example of this effect in Fig. 4.7, which highlights how the fully convolutional online LiDAR embedding learns to remove distractors not relevant to the end task, such as other cars. Cars act as distractors even when parked, as they are not consistently reliable over long periods. In a sense, the embedding networks learn to become car "anti-detectors" despite never seeing a perception training label. This highlights the power of simple tasks such as LiDAR matching for applications like feature pre-training. Recent research has drawn similar parallels between perception and neural rendering (H. Yang et al. 2024).

4.4 Conclusion

In this chapter, we proposed an effective, real-time, and calibration-agnostic LiDAR localization method for self-driving cars. Our method projects the online LiDAR sweeps and the intensity map into a joint embedding space. Localization is conducted through efficient convolutional matching between the embeddings. This approach allows our full system to operate in real-time at 15Hz while achieving centimeter-level accuracy without intensity calibration. The method also generalizes well to different LiDAR types without the need to re-train. The experiments illustrate the performance of the proposed approach over two comprehensive test sets covering over 500 km of driving in diverse conditions.

Nevertheless, in spite of its robustness, the proposed approach still relies on dense intensity map imagery to localize. The storage

requirements for these maps can snowball as the operational domain of a robot grows to city scale and beyond. In the next chapter, we tackle this challenge by extending the proposed approach to incorporate optimizing map compressibility in its learning objective.

5
Task-Specific Map Compression

5.1 Overviews and Motivation

As discussed in the previous chapter, self-driving vehicles usually employ LiDAR-based localization systems to precisely localize within pre-built maps and leverage the information encoded within.

LiDAR localizers rely on the existence of a dense HD map, which contains point clouds (Wolcott and Eustice 2014; Yoneda et al. 2014) or the LiDAR intensity imagery described in Chapter 4 and in papers like (Levinson, Montemerlo, and Thrun 2007; Levinson and Thrun 2010; Wolcott and Eustice 2015). One of the main difficulties in scaling current localization systems to large environments is the storage required for dense HD maps. For instance, storing a LiDAR intensity map as a 16-bit PNG file would require roughly 900 GB for a city such as Los Angeles and over 168 TB for the entire United States[1].

Storing this information onboard the vehicle is infeasible for scalability past a single city[2]. Streaming the HD map data on the go makes the system dependent on a reliable LTE or 5G connection, which may not always be available, while also incurring additional subscription costs. Storage-intensive maps are also time-consuming to deploy and update, which slows down software development and operations.

To address these challenges, in this chapter, we propose to learn to compress the map representation such that it is optimal for the localization task. As a consequence, we can achieve higher compression rates without loss of localization accuracy and robustness compared to standard coding schemes which optimize for reconstruction, thus ignoring the end task. In particular, we leverage a fully convolutional neural network to learn to binarize the map features, and then compress the binarized representation using run-length encoding on top of Huffman coding. Both the binarization net and its corresponding decoder are learned end-to-end using a task-specific loss. We demonstrate the effectiveness of this idea in the context of the LiDAR intensity localization system presented in Chapter 4, and show that it

[1] Based on information from the US Bureau of Transportation Statistics, assuming that it takes approximately 4 MB to store a 150 m road segment as a 16-bit PNG single-channel image (https://www.bts.gov).

[2] While 900 GB is not impossible for modern SDVs, the high-endurance flash storage typically used by such robots is high-value real estate: these drives also need to accommodate files containing the weights of all ML models used onboard, as well as several other map layers. Additionally, the drives also need to have enough space left over to log all sensor data from a day of operations for metrics and monitoring purposes.

is possible to learn a task-specific compression scheme which reduces storage requirements by two orders of magnitude over general-purpose codecs such as WebP without sacrificing performance.

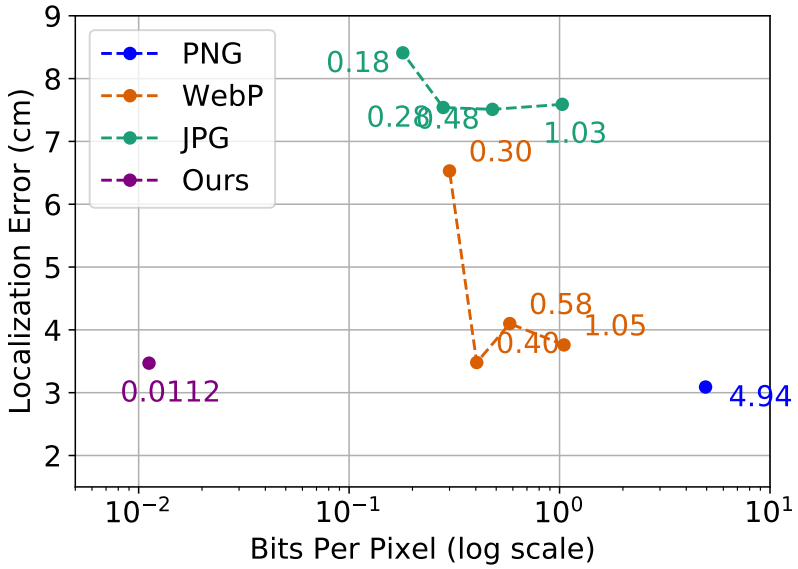


Figure 5.1: **End failure rate for localization under different map compression settings.** Lower and to the left is better. Multiple readings for WebP and JPG represent different quality factors specified during encoding. The numbers represent the precise map storage bitrates.

5.2 End-to-End Compressed Localization

5.2.1 Overview

Our approach learns a compressed deep embedding of the map that can be stored directly onboard, dramatically reducing the requirements of state-of-the-art LiDAR intensity-based localization systems. The compression module can be learned end-to-end, jointly with the deep LiDAR matching approach presented in Chapter 4.

As discussed in the previous chapter, the LiDAR matching model encodes the agreement between the current online LiDAR observation and the map indexed at the hypothesized pose $\mathbf{x}$:

$$P_{\text{LiDAR}} \propto s\left(\pi\left(f_o(\mathcal{I}; \mathbf{w}_\text{o}), \mathbf{x}\right), f_m(\mathcal{M}; \mathbf{w}_\text{m})\right), \tag{5.1}$$

where $f_o(\cdot)$ and $f_m(\cdot)$ are the embedding networks for online LiDAR sweeps, and for the map, respectively, while π is a rigid transform that converts the online embedding image to the map coordinates using the hypothesized pose $\mathbf{x}$; $\mathcal{M}$ is the dense LiDAR intensity map representation, and s is the correlation operator between the online embedding and map embedding. The computation of this term can be written as a feed-forward network, as shown in Chapter 4.

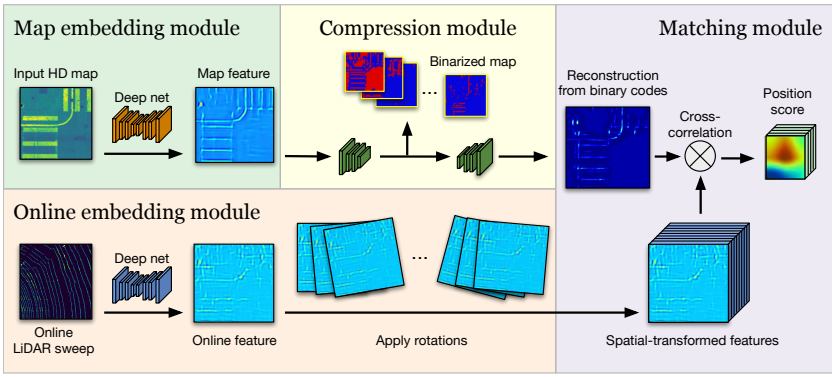


Figure 5.2: **Joint compression and LiDAR matching architecture.** The proposed approach embeds a compression module in the map network and trains it jointly with everything else. This allows our method to discard information irrelevant to LiDAR localization, bringing about substantial gains in compression efficiency.

While effective for localization, the dense intensity map used in Chapter 4 requires a large amount of onboard storage. This prevents the method from scaling to larger operational domains and higher map resolutions. To tackle this problem, we introduce a novel learning-to-compress module that significantly reduces map storage, allowing us to potentially store maps for an entire continent.

5.2.2 Deep Localization with Map Compression

Unlike previous compression networks aimed at optimizing the reconstruction error or perceived visual quality, this chapter argues that optimizing for the task at hand is important for further reducing the storage requirements. Toward this goal, we extend the architecture from Chapter 4 and include a compression module responsible for encoding the map with binary codes through deep convolutional neural networks. Importantly, our encoding can be learned end-to-end with our localizer.

We refer the reader to Fig. 5.2 for an illustration of the overall architecture of our joint compression and matching network. The network includes three components. First, our embedding module takes the map $\mathcal{M}$ and the online LiDAR sweep $\mathcal{I}$ as inputs and computes a deep embedding representation of both. So far, the step is identical to computing the embeddings in the previous chapter. A compression module is then applied over the map embedding layer, which converts the high-dimensional float-valued deep embedding map to a series of low-resolution binary images. We use this representation to store the map in a compact manner at rest and employ a decoding module to decompress the binary codes back into the real-valued embedding representation for inference. Finally, following the now-familiar framework introduced in Chapter 2.4.2, we conduct matching between the reconstructed map embedding and the online embedding. This gives us a score for each possible transformation. We use softmax to build

the probability P_{LiDAR} over our localization search space from the raw matching score. We now describe the modules in more detail.

Embedding Module. The embedding should capture robust yet discriminative contextual features while preserving pixel-accurate details for precise matching. Motivated by this fact, we designed this module to be a fully convolutional encoder-decoder network following Chapter 4. It has a U-Net architecture (Ronneberger, Fischer, and Brox 2015), and the encoder consists of four blocks, each of which has two stride-1 3×3 convolutional layers and one stride-2 3×3 convolutional layer that down-samples the feature map by a factor of 2. The number of channels per block are 64, 128, 256, and 512, respectively. The decoder network has four decoder blocks, each of which takes the last decoder block's output and the corresponding encoder layer feature as input in an additive manner. Each block contains one 3×3 deconvolution layer followed by one stride-1 3×3 conv. The final embedding map has the same spatial resolution as the input and a single channel. In this way, the decoder combines both high-level contextual information as well as low-level details.

Compression Module. The task-specific map compression module is the core contribution of this chapter. The purpose of this module is to convert the large-resolution, high-precision embeddings into low-precision, lower-resolution ones, without losing critical information for matching. We employ a fully convolutional encoder-decoder network to achieve this goal. The output of the encoding module is passed through a grouped soft-max module with a binarization component. The binarization step ensures that the module outputs can be processed effectively by lossless compression methods such as Huffman encoding.

The grouped softmax and subsequent binarization operation are defined as

$$p_j = \frac{\exp(f_j)}{\sum_{k \in \mathcal{S}_j} \exp(f_k)}, \quad b_j = \begin{cases} 1 & \text{if } p_j \geq 0.5 \\ 0 & \text{otherwise} \end{cases}, \tag{5.2}$$

where $\mathcal{S}_j$ is the index group that j belongs to, with each group representing a non-overlapping subset of the full index set $\{1, \ldots, K\}$; $\mathbf{f} = [f_0, \ldots, f_i, \ldots]$ is the input feature volume[3]. The benefit of using grouped-softmax as encoder activation along with the binarizer is twofold. First, within each group, we have at most one non-zero entry. Thus, with the same number of channels, it has better sparsity than the sigmoid function, increasing the compressibility of the binary encoding. Second, compared to standard soft-max, it increases the potential capacity since the grouping of indices allows a more structured encoding. While the component is non-differentiable, backpropagation is still feasible thanks to the use of a straight-through estimator, which

[3] Informally, j indexes over the spatial location. Each group in the softmax yields a "slice" of the binarized map shown in Fig. 5.3.

we will discuss in detail in Chapter 5.2.3.

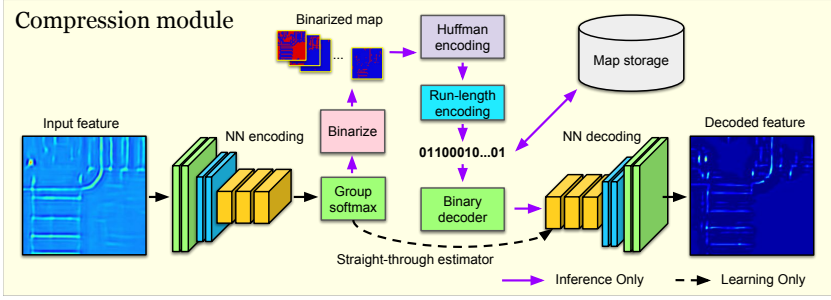


Figure 5.3: **Our compression module.** We obtain gradients for training with a straight-through estimator.

We only need to store the highly compressed binary map embeddings for onboard localization. We process the binarized feature maps using a two-step lossless binary encoding scheme. Our first step is a Huffman encoding. The motivation is that the different features appear in the environment with different frequencies. The Huffman dictionary is built by the one-hot encoding of the softmax latent probability $\mathbf{p}$ per pixel[4].

The second step uses a run-length encoding (RLE) conducted over the flattened Huffman code map to further reduce the size by making use of the fact that codes appear consecutively. For instance, 5555558 could be further reduced to 5681. This gives us the final binary code that we store. Note that both Huffman encoding and run-length encoding are lossless. We choose Huffman+RLE due to its efficiency and effectiveness. Empirically, we also show that this approach reaches 72.5% of the ideal entropy lower bound. While other types of entropy coding, such as arithmetic coding, exist, they are slower and bring marginal improvements to compression rates (Wikipedia Contributors 2019).

The decoder module takes the binary code as input. First, it transforms the Huffman+RLE codes back to the binary map, and then applies a series of deconvolutional blocks to recover the full high-resolution, high-precision embedding of the map that we use for matching. Fig. 5.3 illustrates the structure of the compression module.

Matching Module. Our matching module follows the previous two chapters, where a series of spatial transformer networks rotate the online embedding multiple times at $|\Theta|$ different candidate angles. Within each rotation angle, translational search based on inner-product similarity is equivalent to convolving the map embedding with the online embedding as kernels. Thus, enumerating all the possible pose candidates is equivalent to a convolution with $|\Theta|$ kernels. Unlike standard convolutions used in convolutional neural networks, this

[4] Thus, for a 128-softmax vector, the dictionary size is 128. Frequency is computed in a batch manner. For instance, if the 'class' 5 (00000101) appears 50% of the time we will use a shorter-length code 0 to encode it.

convolution has a very large kernel. We exploit FFT-conv as before to accelerate our matching modules by an order of magnitude over traditional, matrix-multiplication-based convolutions on a GPU.

5.2.3 Training

We train LiDAR localization and map compression jointly with a two-part loss. The first part mirrors the loss used to train the original compression-less matching system, while the second promotes efficient compression:

$$\mathcal{L} = \mathcal{L}_{\text{Match}}(\mathbf{y}, \mathbf{y}^{(\text{GT})}) + \lambda_1 \mathcal{L}_{\text{CodeLen}}(\mathbf{p}) + \lambda_2 \mathcal{L}_{\text{HardBin}}(\mathbf{p}). \tag{5.3}$$

Here, $\mathbf{y}$ is used to denote the final softmax-normalized output probability, $\mathbf{y}^{(\text{GT})}$ is the one-hot encoded ground truth offset, and $\mathbf{p}$ is the embedding after the group-softmax layer described earlier.

$\mathcal{L}_{\text{Match}}$ is the original LiDAR localization matching loss described previously in Eq. (4.1). It is implemented as a simple cross-entropy loss which encourages the matching score to be the highest at the GT position while lowering the scores of positions elsewhere:

$$\ell_{\text{Loc}}(\mathbf{y}, \mathbf{y}_{\text{gt}}) = \sum_i y_{\text{gt},i} \log(y_i). \tag{5.4}$$

As before, jittering the ground-truth alignment is important to avoid overfitting. $\mathcal{L}_{\text{CodeLen}}$ and $\mathcal{L}_{\text{HardBin}}$ are described next.

Code Length Loss. Intuitively, we want the codes produced by our encoder to be as short as possible, in order to maximize storage efficiency. Given that the output size of our network is fixed in practice, we instead minimize the entropy of their codes, as proposed in past deep compression approaches (Toderici et al. 2016).

Producing low-entropy codes makes them easy to compress using established entropy coding algorithms, such as Huffman Coding or Arithmetic Coding. This is a direct consequence of Shannon's Source Coding Theorem (Shannon 1948), which states that entropy acts as a lower bound to the optimal code length. We therefore also use entropy as our surrogate loss, resulting in the following code length loss, defined over a mini-batch of size B:

$$\mathcal{L}_{\text{CodeLen}}(\mathbf{p}) = \bar{\mathbf{p}} \log \bar{\mathbf{p}}, \tag{5.5}$$

where $\bar{\mathbf{p}} = \frac{1}{W \times H \times B} \sum_i \mathbf{p}_i$ is the mean probability of the probabilistic code outputs.

Hard Binarization Loss. In order to apply entropy coding on the codes, they must first be binarized by clamping them to either zero or one following Eq. (5.2). This operation is not differentiable, and we use a straight-through estimator (Bengio, Léonard, and Courville

2013) for training: during the forward pass we do binarize the codes output by the network, but during the backward phase, we treat the binarization as an identity function.

This induces a gap between the binary values used by the compression decoder and those actually output by the encoder.

In order to reduce this gap and reduce the information loss induced by binarization, we regularize the individual code outputs to be as close as possible to zero and one. We implement this regularization with a per-pixel entropy term

$$\mathcal{L}_{\text{HardBin}}(\mathbf{p}) = \sum_i p_i \log p_i, \tag{5.6}$$

where $\mathbf{p}$ represents all soft code probability outputs, and i simply sums over all the code pixels produced by the encoder. We find that this approximation provides good gradients for the function to be learned.

Efficient Inference. After training the full system, in the offline map encoding stage we use the trained compression network to compress the map into a binary code to minimize the onboard storage requirements. During onboard inference, the compressed code is recovered, and the decoder transforms it into the dense HD map embedding, used for localization.

As in the previous chapters, we integrate the proposed matching architecture into a histogram filter with a 5cm resolution in the x and y dimensions. The main difference from the approach in Chapter 2.4.2 is that when computing the BEV term for LiDAR matching $P_{\text{LiDAR}}(\mathcal{I}_t \mid \mathbf{x}; \mathbf{w})$, we first retrieve a local map binary code $\mathbf{b}$. The feature network computes the LiDAR embedding and then $\mathbf{b}$ is passed through the decoder of the compressor to recover the map embedding $f_m(\mathcal{M}; \mathbf{w}_m)$. After that, the matching score P_{LiDAR} can be computed efficiently for all hypothesized poses using the now-familiar FFT-conv operation.

As discussed in Chapter 2.4.2, the GPS and dynamics terms can be computed efficiently in this setting and fused with the other terms. Following the previous chapters, once each term has been computed, the posterior can be obtained with a multiplication followed by renormalization. Then, if desired, the point estimate of the pose at the current time is computed as a soft-argmax aggregation.

5.3 Experimental Results

5.3.1 Datasets

The proposed joint map compression and LiDAR localization system is evaluated on two datasets: an urban dataset collected in Pittsburgh,

PA, and the highway dataset from Chapter 4, which is collected in Pennsylvania and California.

The highway dataset contains over 400 sequences of highway driving with a total of 3,000 km traveled. It contains a dense bird's-eye view LiDAR intensity map stored in lossless PNG format at a resolution of 5cm/px. The self-driving vehicles used to collect the data are outfitted with a 64-beam LiDAR operating at 10Hz, an IMU, wheel encoders, and a consumer-grade GPS. We follow the same setting as in the previous chapter and select 282 km of driving as testing, ensuring that there is no geographic overlap between the splits.

The urban dataset consists of 15,554 km of driving. It covers diverse road structures and challenging scenarios, including various types of intersections, reversing, and parallel parking, as well as some regions with poor lane markings and map changes.

The ground truth poses for both datasets are computed offline, using a high-precision batch optimization leveraging LiDAR, IMU, and wheel encoder data to localize the vehicle against the map. The maps are built using multiple passes through the covered areas, using a solution based on Graph-SLAM (Thrun, Burgard, and Fox 2005).

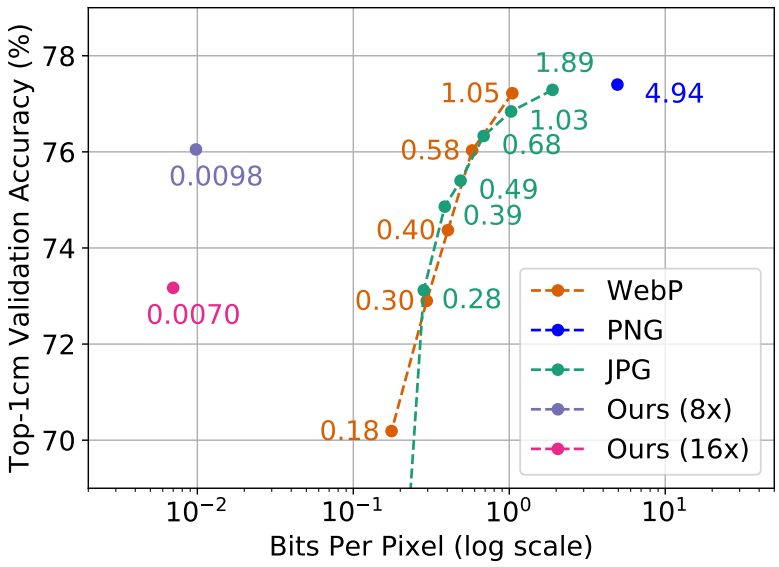


Figure 5.4: **Top-1 Matching Performance vs. Bits per Pixel.** The diagram plots how well we can register an observation to the prior map, as a function of how well the map is compressed. Higher and more to the left is better.

5.3.2 Experimental Setup

To our knowledge, at the time of writing, no previous work has integrated LiDAR intensity localization with deep compression. We therefore evaluate our work against baselines without compression on

localization metrics alone, such as those described in Chapter 4, and measure the performance degradation when using the map compression module.

Since the intensity map is stored as an image, we compare our compression to several traditional image compression algorithms such as JPEG and WebP. For each compression algorithm, we compress the training and testing map images and train a standard learn-to-localize matching network as in Chapter 4. We also train a reconstruction-based compression network ('ours (recon)'), which shares the same architecture as our compression module, with the only exception that it is trained for reconstruction error of the feature map only, not for matching performance. This is meant to analyze whether the task-specific compression helps our matching task.

For our proposed method, we adopt two different compression settings by changing the downsampling levels we used for the binary codes. We evaluate an 8× downsampling model and a 16× downsampling model, where the 16× model has an extra set of downsampling and upsampling modules before binarization, which leads to more compression at a small cost in localization accuracy. All the competing algorithms have the same embedding feature network as our proposed model. Additionally, we performed experiments with reduced map resolution as an alternative baseline for reducing map storage.

The neural network encoder is a fully convolutional residual network where each scale includes two 3×3 standard residual blocks (K. He et al. 2016) and a stride-2 3×3 conv between scales. The channel counts per scale are $8, 16, 32$ and 64 respectively. The decoder is a fully convolutional network with several transposed convolutional layers. We use PReLU (K. He et al. 2015) as the activation function.

We train all competing algorithms over 343k and 230k training samples for the urban and highway datasets respectively. We aggregate five online LiDAR sweeps and rasterize them into a bird's-eye view image at 5cm/pixel, to match the map resolution. We build an intensity image centered at the SDV, covering a range of $(-12\text{m}, 12\text{m})$ laterally, and $(-15\text{m}, 15\text{m})$ longitudinally. All networks are trained on four Nvidia 1080 Ti GPUs using PyTorch. We use the Adam optimizer (Kingma and Ba 2015) with an initial learning rate of 10^{-3}. We observed that training the matching network end-to-end from scratch works, but is slow to converge. We therefore adopt a stage-wise training procedure. First, we train the matching network without compression, as described in Chapter 4.2, then add the compression module and continue with end-to-end learning.

The proposed system uses LiDAR readings, an inertial measurement unit (IMU), and wheel encoders to localize against a pre-built LiDAR intensity map. The formulation assumes geometrically cali-

brated sensors.

Method	Median error (cm)			Failure rate (%)			Bits per pixel
	Lat	Lon	Total	≤ 100m	≤ 500m	End	
Lossless (PNG)	**3.62**	**4.53**	**7.06**	**0.00**	**0.35**	**0.71**	4.94
WebP-50	3.87	4.87	7.52	**0.00**	0.71	**0.71**	1.46
WebP-20	4.03	5.27	8.02	**0.00**	1.06	8.87	0.91
WebP-10	4.45	7.09	9.79	0.35	9.57	24.37	0.70
WebP-5	4.10	6.40	8.99	0.35	9.57	14.69	0.55
Ours	**3.62**	4.77	7.19	0.35	**0.35**	**0.71**	**0.007**

Table 5.1: **Online localization performance on the highway dataset.** The proposed learning-based compression method outperforms all traditional, off-the-shelf approaches, while requiring less storage by several orders of magnitude.

5.3.3 Matching Performance

We begin by evaluating the validation performance of our LiDAR matcher with and without compression. Next, we perform several ablation studies to motivate the various design choices in our architecture, after which we integrate the matcher into a localizer and evaluate its online localization performance and failure rate on our test set.

In order to evaluate the performance of the models in terms of finding the best match in a compressed map, we report the performance of the competing algorithms under the matching setting.

We conduct matching over a 1×1 m search range, after perturbing the initial position of the vehicle around the GT position. We uniformly sample the translational perturbation within this 1 m^2 region, and the angular perturbation between 0 and 5°. We report top-1 px and top-9 px as our metrics, representing whether the prediction is in the same pixel as the GT or within the 3×3 px region centered around the GT, respectively. We report matching accuracy as a function of bitrate per pixel on the urban dataset in Fig. 5.4. Note that the proposed algorithm in the 8× setting achieves 76% top-1 px accuracy with a map compression rate of 0.0098 bits per pixel. Both results are better than all competing algorithms. Furthermore, the top-9 px accuracy we obtain is on par with the no-compression upper bound. Notably, the BPP is around 20-400 times smaller than all competing algorithms. Under the 16× setting the top-1 px accuracy is 3% lower but achieves a higher compression rate at 0.007 bits per pixel.

Ablation Study. We conduct an ablation study over the matching performance. We first validate whether jointly training the compression module with our matching task loss helps improve the matching performance and increase the compression rate. For this, we train a compression module using only reconstruction loss (without the

Method	BPP	Top-9 px	Top-1 px
Lossless (PNG)	4.93	97.47%	**77.40%**
Ours (recon, 8×)	0.0520	97.27%	75.83%
Ours (recon, 16×)	0.0140	96.95%	74.86%
Ours (match, 8×)	0.0098	**97.73%**	76.05%
Ours (match, 16×)	**0.0070**	97.25%	73.17%

Table 5.2: **Ablation studies on matching performance.** Optimizing jointly for both map reconstruction and matching greatly reduces the storage requirements compared to lossless codecs such as PNG, but task-specific supervision leads to a superior compression rate.

matching task loss). Tab. 5.2 illustrates the results of this ablation. Jointly training the compression module with the task-specific loss greatly helps the performance. The 16× downsampled model pushes the compression rate even further, with a 3% drop on top-1px results.

In the next section, we also analyze whether a lower compression rate can be achieved by downsampling the map spatially.

5.3.4 Online Localization

Nevertheless, while matching performance is a good proxy metric, in order to truly validate a LiDAR matcher we must analyze it within the context of a full localization system. We follow Chapter 4 and compute the median and worst-case localization error on the test split as our metrics. Specifically, we report median, p95, and p99 errors in meters along the lateral and longitudinal directions. We also report an out-of-range (failure) rate, which represents the percentage of 1 km segments where the method reaches a localization error of 1 m or higher.

Method	**Median error (cm)**			**Failure rate (%)**			**Bits per pixel**
	Lat	Lon	Total	≤100m	≤500m	End	
Lossless PNG	**1.55**	**2.05**	**3.09**	**0.00**	1.09	2.44	4.94
JPG-50	3.29	5.60	7.59	**0.00**	1.09	5.26	1.03
JPG-20	3.77	4.99	7.51	**0.00**	**0.00**	1.75	0.48
JPG-10	3.42	5.46	7.54	**0.00**	1.09	5.26	0.28
JPG-5	4.32	5.48	8.41	**0.00**	1.09	1.25	0.18
WebP-50	1.62	2.75	3.76	**0.00**	3.26	3.30	1.05
WebP-20	1.86	2.85	4.10	4.08	8.70	14.63	0.58
WebP-10	1.60	2.26	3.48	**0.00**	1.09	2.50	0.40
WebP-5	1.65	5.75	6.53	2.04	5.43	13.95	0.30
Ours	1.61	2.26	3.47	**0.00**	1.09	**1.22**	**0.0083**

Table 5.3: **Online localization performance on the urban dataset.** Consistent with our analysis at the matcher level, the proposed approach exceeds off-the-shelf compression methods in terms of localization metrics, while approaching the "oracle" performance of lossless representations.

Tab. 5.3 shows the online localization performance on the urban dataset. We use 'method-X' to denote the quality setting of a given method, such as JPEG-5. The quality parameter, which takes values from 0 to 100, is used to trade off storage for reconstruction accuracy.

A quality of zero means the most aggressive lossy compression possible, often resulting in heavy distortion, while a quality of 100 is equivalent to, or close to, lossless compression. While most of the baselines provide reasonable results, our method is better than competing algorithms such as JPEG-5 and WebP-5, which show high failure rates at extreme compression levels. In terms of worst-case performance, measured by failure rate, our method is on par with high-quality compression such as WebP-50 and JPEG-50, and a lossless method, while our bitrate per pixel is 100 times smaller. This is shown in Fig. 5.1, where we plot the percentage of failures after 1 km against the storage.

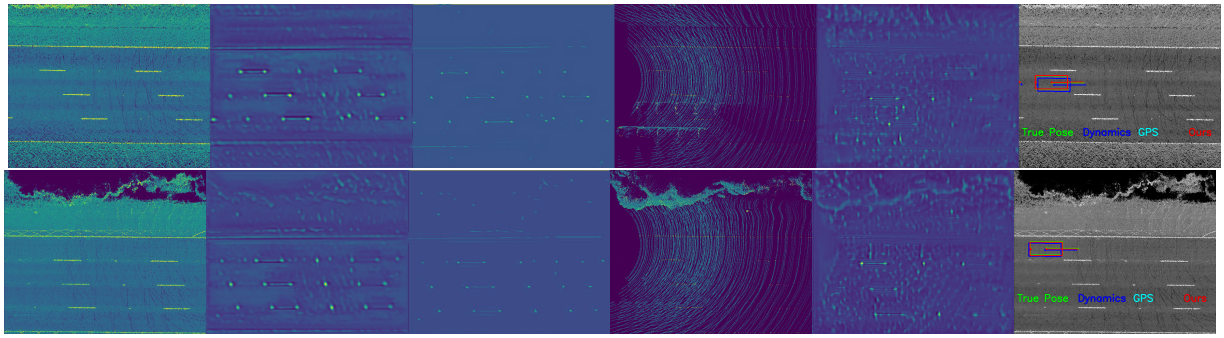


Figure 5.5: **Qualitative results from our highway dataset.** From left to right: (1) the original map, (2) its computed deep embedding, (3) the compressed embedding, (4) online LiDAR observation, (5) its embedding, and (6) the localization result.

Tab. 5.1 depicts the online localization performance on the highway dataset. We can see that traditional off-the-shelf compression algorithms like WebP have a large performance drop compared to our compression-based matching. While the method using the reconstruction loss obtains storage roughly in the same magnitude as our approach, it suffers a large performance drop. This indicates the importance of the matching loss term for effectively selecting portions of the map to keep. Meanwhile, our method based on the matching task loss exhibits no performance drop at half the storage of the pure reconstruction network, making it more than 400 times smaller than the lossless compression bitrate.

Tab. 5.4 showcases the ablation study on the urban dataset. We compare reconstruction-loss-driven compression models against our matching-loss-driven compression model under various architectures. The table shows that in terms of online localization error, the compression model trained with a task-specific-driven loss is better than the reconstruction-driven model, with smaller bitrates and a lower failure rate.

Additionally, as we will see in Chapter 6, the computational overhead of localization can be further reduced by sharing features with other tasks, such as perception.

Comparison to smaller map resolutions. We also performed

Method	Median error (cm)			Failure rate (%)			Bits per pixel
	Lat	Lon	Total	≤100m	≤500m	End	
Lossless PNG	**1.55**	**2.05**	**3.09**	**0.00**	1.09	2.44	4.93580
Ours (recon, 8×)	1.59	2.16	3.24	**0.00**	1.09	**1.22**	0.02689
Ours (recon, 16×)	1.76	2.48	3.62	**0.00**	**0.00**	2.56	0.01155
Ours (match, 8×)	1.61	2.26	3.47	**0.00**	1.09	**1.22**	0.00830
Ours (match, 16×)	1.62	2.77	3.84	1.00	2.17	4.26	**0.00733**

Table 5.4: **Ablation studies on the urban dataset.** We compare a map reconstruction loss with our task-specific matching loss, each under two different configurations of our binary code generator.

Method	Res	Median Err (cm)			Failure Rate (%)			Bits per m^2
	cm/px	Lat	Lon	Total	≤ 100m	≤500m	End	
PNG	5	**1.55**	**2.05**	**3.09**	**0.00**	1.09	**2.44**	1948.55
PNG	10	4.37	6.68	9.50	3.19	3.26	4.00	402.84
PNG	15	15.73	23.66	31.73	10.31	20.65	22.03	173.97
JPG	10	4.51	5.78	8.95	0.00	1.09	10.64	63.42
JPG	15	11.67	18.20	25.14	9.28	13.04	16.28	29.00
Ours	5	1.76	2.48	3.62	**0.00**	**0.00**	2.56	**2.87**

Table 5.5: **Localization performance on our urban dataset using maps of reduced spatial resolution.** We used 5cm/px in the submission. Map storage is measured in bits/m^2 in order to account for different resolutions (bits-per-pixel (bpp) are no longer meaningful if the area of a pixel can change). *Ours* refers to our 16× downsampling method. JPG quality is 50.

experiments with reduced map resolutions and off-the-shelf lossy image compression on our urban dataset to investigate the impact on storage requirements and localization accuracy. We note that unlike the tables in the paper, here we measure the storage requirements in bits / m^2, in order to account for the different map resolutions. As shown in Tab. 5.5, in terms of storage requirements, using JPG at 15cm/px does come within roughly an order of magnitude of the storage required by our approach. However, the localization performance is substantially reduced (16.28% failure rate, as opposed to 2.56% for our binary coding).

5.3.5 Qualitative Results

Fig. 5.5 shows examples of the deep map embeddings computed by our system (before and after compression) together with the (uncompressed) online observation embedding and the localization result.

5.3.6 Storage Analysis

We now turn back to the approximate storage requirements described in this chapter's introduction, and showcase projected numbers when compressing all maps using our proposed method in Tab. 5.6. Our approach can compress a 5cm/px HD map of the entire Los Angeles county to just 1.5 GB, allowing it to fit in RAM on most current smartphones. We can also fit the entire USA road network at the same resolution in just 280 GB.

Compression	LA County	Full US
Lossless (PNG)	900 GB	168 TB
WebP 1	32 GB	5.99 TB
Ours (match, 8×)	**1.5 GB**	**0.28 TB**

Table 5.6: **Estimated map storage requirements using various compression methods.** Estimates of road network length based on numbers provided by the US Bureau of Transportation Statistics.

5.3.7 Discussion

HD maps can provide a wide range of rich prior knowledge to autonomous vehicles, acting in effect as yet another sensor. However, the SDV must be able to robustly estimate its pose to leverage this information.

While deep LiDAR matching can be trained to achieve superior robustness to non-deep counterparts, its scalability is still limited by its reliance on dense map imagery. Off-the-shelf image compression can alleviate some of this cost, but its applicability is limited due to the misalignment between the localization performance that we are interested in, and the distortion minimization optimized by traditional image compression.

In this chapter, we proposed to align these objectives by learning to compress the map representation such that it is optimal for the localization task. Our experiments on a state-of-the-art LiDAR localizer have shown that it is possible to learn a task-specific compression scheme which reduces storage requirements by two orders of magnitude compared to general-purpose codecs such as WebP without sacrificing localization performance. Furthermore, by integrating the learning of our domain-specific compression network into the training pipeline of the deep LiDAR matcher, we minimize the computational and operational overhead.

6
Towards Full-System Understanding: Joint Localization and Perception

6.1 Overview

It is important to take a step back and consider the purpose of high-precision localization. At a coarse level, localization is used for high-level navigation. However, at a finer level, it is used to precisely integrate detailed information from the map into the decision-making process of a robotic system.

One common application takes the form of using HD maps[1] in SDV applications. These need to be reasonably aligned with the vehicle's local reference frame in order to allow meaningful fusion between sensor and map data. A key design question thus arises: How good does this alignment need to be?

[1] We use the same nomenclature as in Chapter 3 and use the term 'HD map' to specifically refer to sparse semantic information such as lane boundaries, centerlines, traffic sign locations, etc.

In this chapter, we investigate this question in the context of autonomous driving. Specifically, we first explore the impact of localization errors on the entire SDV software stack, culminating with analyzing motion planner outcomes. Second, we propose a lightweight addition to a perception and prediction system capable of correcting pose errors stemming from localization.

In practice, the proposed method could be used either as a fallback if the primary localizer (e.g., a geometry-based registration localizer (Yoneda et al. 2014)) fails, or, thanks to its computational efficiency, the primary localizer in resource-constrained systems.

We focus our attention on bounded, in-domain, localization errors, for example, those caused by an Iterative-Closest-Point-based localizer failing to converge to a solution. We leave complete failures, like obstructed sensors or system crashes, and out-of-distribution errors, like failing to localize in a completely different operational domain, as future work. We refer the reader to recent papers such as (Deschênes et al. 2021; Ebadi et al. 2023) for discussions on the design of systems resilient to a broader class of failures.

6.2 Background

As reviewed in Chapter 2.3, many tasks in robotics can be broken down into a series of subproblems that are easier to study in isolation, facilitating the interpretability of system failures (B. Zhou, Krähenbühl, and Koltun 2019). In particular, it is common to subdivide the self-driving problem into five critical subtasks:

1. Localization: placing the car on a high-definition (HD) map with centimeter-level accuracy.
2. Perception: estimating the number and location of dynamic objects in the scene.
3. Prediction: forecasting the trajectories and actions that the observed dynamic objects might do in the next few seconds.
4. Motion planning: coming up with a desired trajectory for the ego-vehicle, and
5. Control: using the actuators (i.e., steering, brakes, throttle, etc.) to execute the planned motion.

Moreover, it is common to solve the above problems *sequentially*, such that the output of one subsystem is passed as input to the next, and the procedure is repeated iteratively over time. This *classical* approach lets researchers focus on well-defined problems that can be studied independently, and these areas tend to have well-understood metrics that measure progress on their respective sub-fields. For simplicity, researchers typically study autonomy subproblems assuming that their inputs are correct. For example, state-of-the-art perception-prediction (P2) and motion planning (MP) systems often take HD maps as input, thereby assuming access to accurate online localization. We focus our attention on this assumption and begin by studying the effect of localization errors on modern autonomy pipelines. Here, we observe that localization errors can have severe consequences for P2 and MP systems, resulting in missed detections and prediction errors, as well as bad planning that leads to larger discrepancies with human trajectories and increased collision rates. Please refer to Fig. 6.1 for an example of an autonomy error caused by inaccurate localization.

In contrast to the classical formulation, recent systems have been designed to perform multiple autonomy tasks jointly. This *joint* formulation often comes with a shared neural network backbone that decreases computational and system complexity while still producing interpretable outputs, making it easier to diagnose system failures. However, these approaches have so far been limited to jointly performing perception and prediction (P2) (W. Luo, Yang, and Urtasun 2018; Casas, Luo, and Urtasun 2018; Casas, Gulino, Suo, Luo, et al. 2020; Liang et al. 2020), P2 and motion planning (P3) (Zeng et al.

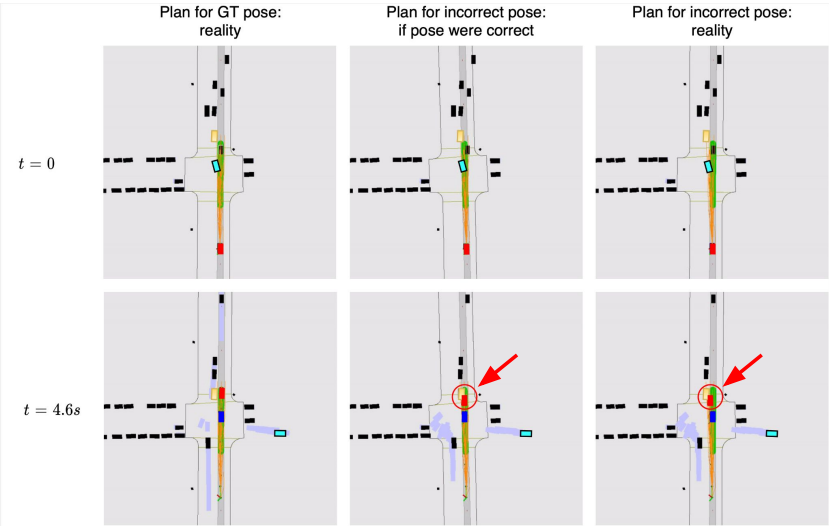


Figure 6.1: **A scenario where a small localization error results in a collision.** The top row visualizes the first time step, and the bottom row visualizes a later time step where a collision occurs. **Black rectangles** represent reality; the pale blue rectangles are forecasted object trajectories. The SDV is the red rectangle. The samples predicted by the motion planner are shown as orange lines. The three columns visualize different variants of the same scenario. (Left) The planned trajectory of the SDV when there is no localization error. (Middle) What the SDV "thinks" is happening, based on its estimated pose that has an error of (x, y, yaw) = (10 cm, 0 cm, 1.5 deg). (Right) What the SDV is actually doing when subject to the pose error; this is the same trajectory as shown in the middle image, but rigidly transformed so that the initial pose agrees with the GT pose. The collision (red circle) occurs because the yellow vehicle is not perceived at t = 0 due to occlusion (by the cyan vehicle); the localization error then causes the SDV to go into the lane of opposite traffic, which results in a collision.

2019, 2020; Sadat et al. 2020; Y. Hu et al. 2023), semantic segmentation and localization (Radwan, Valada, and Burgard 2018) or road segmentation and object detection (Teichmann et al. 2018).

In this chapter, and informed by our analysis of the effects of localization error, we apply the joint design philosophy to the tasks of localization, perception, and prediction; we refer to this joint setting as **LP2**. We design an LP2 system that shares computation between the tasks, which makes it possible to perform localization with as little as 2 ms of computational overhead while still producing interpretable localization and P2 outputs. We evaluate our proposed system on a large-scale dataset in terms of motion planning metrics, and show that the proposed approach matches the performance of a traditional system with separate localization and perception components, while being able to correct localization errors online, in addition to being overall faster and simpler.

6.3 The Effects of Localization Error

Since state-of-the-art perception-prediction (P2) and motion planning (MP) stacks make extensive use of accurate localization on high-definition maps (often assuming perfect localization (Casas, Luo, and Urtasun 2018; Casas, Gulino, Suo, Luo, et al. 2020; M. Bansal, Krizhevsky, and Ogale 2019; Zeng et al. 2019)), we study the effects of localization error on a state-of-the-art P2 and MP pipeline. We begin by describing how these modules work and how they use localization.

Perception-Prediction (P2). P2 models are tasked with perceiving actors and predicting their future trajectories to ensure that

motion planning has access to safety-critical information about the scene for the entire planning horizon.

We study the state-of-the-art Implicit Latent Variable Model (Casas, Gulino, Suo, Luo, et al. 2020) (ILVM), the latest of a family of methods that use deep neural networks with voxelized LiDAR inputs to jointly perform detection and prediction (W. Luo, Yang, and Urtasun 2018; Casas, Luo, and Urtasun 2018; Zeng et al. 2019). ILVM encodes the whole scene in a latent random variable and uses a deterministic decoder to efficiently sample multiple scene-consistent trajectories for all the actors in the scene. Besides LiDAR, the ILVM backbone takes as input a multi-channel image which encodes the semantics of the surrounding map[2], which the model is expected to use to improve detection and forecasting. While we always process the LiDAR scans in the vehicle frame, aggregating them using relative pose information, the alignment between them and the map relies on an absolute, map-relative pose. Thus, localization error results in a *misalignment* between the semantic map and the LiDAR scan.

[2] Each channel encodes a particular kind of map element. One channel encodes walkways, one encodes lanes, etc., for a total of 13 layers. Please refer to (Casas, Gulino, Suo, Luo, et al. 2020) for more details.

Motion Planning (MP). Given a map and a set of dynamic agents and their future behaviors, the task of the motion planner is to provide a route that is safe, comfortable, and physically realizable to the control module. We study the Path Lateral Time (PLT) motion planner (Sadat et al. 2019), a method that samples physically realizable trajectories, evaluates them, and selects the one with the minimal cost.

The PLT planner receives $S = 50$ Monte Carlo samples from the joint distribution over the trajectories of *all* actors $\{Y^1, \dots, Y^S\}$ from the P2 module. It then samples a small set of trajectories $\tau \in \mathcal{T}(\mathcal{M}, \mathcal{R}, \mathbf{x}_0)$ given the map $\mathcal{M}$, high-level routing $\mathcal{R}$, and the current state of the SDV $\mathbf{x}_0$. The planned trajectory

$$\tau^* = \underset{\tau \in \mathcal{T}(\mathcal{M}, \mathcal{R}, \mathbf{x}_0)}{\operatorname{argmin}} \sum_{s=1}^{S} c(\tau, Y^s) \tag{6.1}$$

is then computed to be the one with the minimum expected cost over the predicted futures, as defined by a cost function c that takes into account safety and comfort. In this case, erroneous localization gives the planner a wrong idea about the layout of the static parts of the scene.

6.3.1 Experimental setup

LP3 Dataset. Evaluating the localization, perception, prediction, and motion planning tasks requires a dataset that contains accurately localized self-driving segments, together with the corresponding HD appearance maps (to evaluate localization), as well as annotations of

dynamic objects in the scene, their tracks, and their future trajectories (to evaluate P3). At the time of writing, no current public dataset satisfied all these criteria[3]. Therefore, we used our own LP3 dataset. The LP3 dataset is named after its ability to allow evaluating Localization (L) as well as Perception, Prediction, and motion Planning (P3). LP3 is a subset of the ATG4D dataset (Casas, Gulino, Suo, Luo, et al. 2020; Zeng et al. 2019; Casas, Luo, and Urtasun 2018), which also has appearance maps available. In particular, our maps consist of 2D images generated from aggregated LiDAR intensity, which summarizes the appearance of the ground (cf. the top left of Fig. 6.3) and are between 6 and 12 months old by the time the SDV traverses the scene. The dataset encompasses 1,858 sequences of 25 seconds each, all captured in Pittsburgh, PA.

[3] A few days after submitting our original paper, the nuScenes dataset (Caesar et al. 2020) added support for an appearance layer, thereby enabling similar experiments to the ones we present in this work.

Besides bounding boxes for vehicles, pedestrians, and bicycles in the scene, the dataset provides *semantic map* annotations, such as lanes, traffic signs, and sidewalks. Importantly, the LP3 dataset also provides a map *appearance layer* comprised of the LiDAR intensity of the static elements of the scene as captured by multiple passes of LiDAR scans through the area (please refer to the top left of Fig. 6.3 for an example). Our LP3 dataset makes it possible to evaluate methods that jointly perform appearance-based LiDAR localization and P2, and to calculate motion planning metrics.

Simulating Localization Error. We simulate localization error and study its effects on downstream autonomy tasks. Given a maximum amount of noise $m \in \mathbb{R}$ (which we call *maximum jitter*), we perturb the ground truth pose on evaluation frames by sampling translational or rotational noise from a uniform distribution $\varepsilon \sim \mathcal{U}(-m, m)$. To understand the effects of different types of noise, we evaluate translational noise and rotational noise independently.

Metrics. For perception, we focus on the mean average precision metric with at least 70% overlap between the predicted and the ground truth boxes (mAP@0.7, following (Casas, Luo, and Urtasun 2018)). For prediction, we report the mean scene final displacement error (mean SFDE[4]) between the ground truth and the predicted trajectory after 5 seconds (i.e., planning horizon) (Casas, Gulino, Suo, Luo, et al. 2020).

[4] We use meanSFDE instead of minSFDE since for large numbers of samples ($S = 50$ in our case), minSFDE is overly optimistic. The presence of unrealistic false positive trajectory samples can interfere with the SDV, causing it to break or swerve, creating a dangerous situation. However, the minSFDE will not capture this dangerous behavior as long as there is at least one good sample. Please refer to (Casas, Gulino, Suo, and Urtasun 2020) for a more detailed discussion.

We run the planner at the beginning of the segments and let the trajectory unfold for 5 seconds. We then measure the percent of segments for which there is a collision, and the ℓ_2 distance between the predicted trajectory and the trajectory followed by the human driver after 5 seconds.

Note that in our setting, all the actors are "passive," in the sense that they follow their pre-recorded trajectory independently of the actions taken by the SDV. This is often called an *open-loop* evaluation.

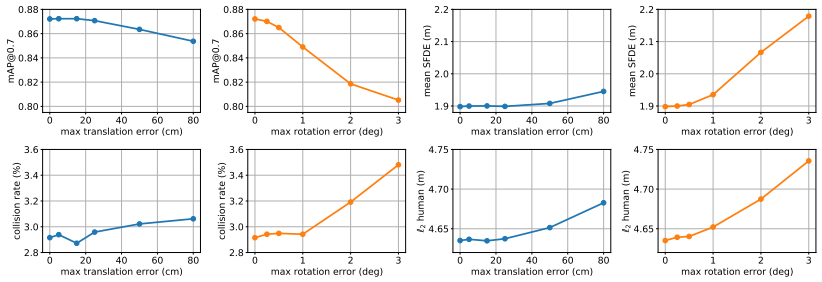


Figure 6.2: **The effects of localization error on perception-prediction and motion planning.** (Top) The effects of perturbing the ego-pose on P2. SFDE is the mean displacement error across all samples at the 5s mark as defined in (Casas, Gulino, Suo, Luo, et al. 2020), and mAP@0.7 is the mean average precision evaluated at an IOU of 0.7. (Bottom) The effects of perturbing the ego-pose on planning. The collision rate is the percentage of examples for which the planned path collides with another vehicle or pedestrian within the 5s simulation, and ℓ_2 human is the distance between the planned path and the ground truth human-driven path at the 5s mark.

While evaluating our task in a *closed-loop* setting would be more desirable, building a simulator of reactive agents and counterfactual sensor inputs comes with its own set of challenges (e.g., realistic LiDAR simulation, realistic controllable actors, etc.) and is out of the scope of our work.

6.3.2 Results

We show the effects of perturbing the pose of the ego-vehicle on P2 We observe that the performance of ILVM is barely affected by translational jitter up to 25 cm, and rotational jitter up to 0.5°. Larger amounts of translational noise have little effect (~2% mAP, .05 mean SFDE) up to 80 cm, while the effect is stronger for rotational error (~7% mAP, 0.30 mean SFDE) up to 3°.

In the bottom row of Fig. 6.2, we show the effects of perturbing the ego-vehicle pose on motion planning. Similar to P2, MP performance does not degrade much until translational noise exceeds 25 cm or rotational noise exceeds 0.5°. We also observe that large translation errors have small effects relative to rotational noise for both collision rate and distance to human route. While this is somewhat expected (as rotational error can cause straight paths to run into sidewalks or oncoming traffic), it is interesting to quantify these effects formally.

6.4 Joint Localization, Perception, and Prediction

We now formulate a model that performs joint localization and P2 (LP2). First, we lay out the key challenges that we would like our system to overcome and then explain our design choices in detail.

6.4.1 System Desiderata

Low Latency. In order to provide a safe ride, a self-driving car must react quickly to changes in its environment. In practice, this means that we must minimize the time from perception to action. In a naive,

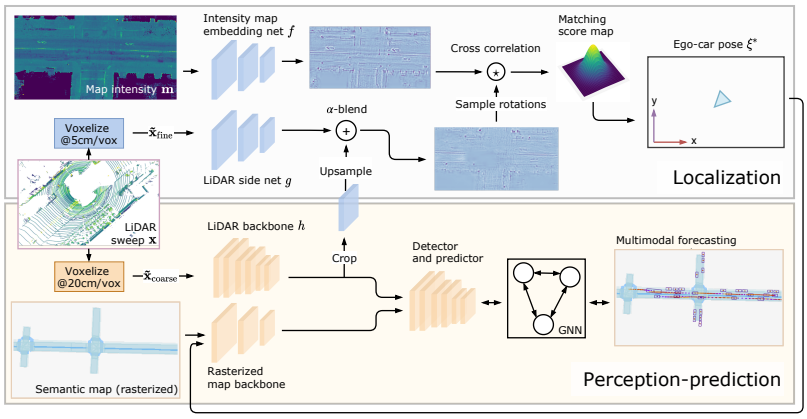


Figure 6.3: **The architecture of the combined localization and perception-prediction (LP2) model.** We integrate a LiDAR matching component into a perception architecture, which gives it the ability to recover from upstream localization problems. By sharing feature maps between the two tasks, we can achieve this with very little computational overhead.

cascaded autonomy system, the running time of each component adds up linearly, which may result in unacceptable latency. It is, therefore, necessary for new additions to the system to have as little impact on the overall run time as possible, for example, by reusing computation wherever possible.

Learning-Based Localization. As discussed in Chapters 4, 5, and 7, localization systems with learned components are typically better at discerning semantic aspects of the scene that are traditionally difficult to discriminate with purely geometric features (e.g., growing vegetation, tree stumps, and dynamic objects). Additionally, they have the potential of being more invariant to appearance changes due to season, weather, and illumination. Therefore, we would like to incorporate a learning-based localization component in our system. Moreover, since P2 systems are typically heavily driven by learning, it should be possible to incorporate learning-based localization by sharing computation between the two modules, resulting in reduced overhead to the overall LP2 system.

Simple Training and Deployment. We would like our joint LP2 system to be easier to train and deploy than its classical counterpart. Given the large amounts of ML infrastructure invested around P2 systems (e.g., on dataset curation, labeling, active learning, and monitoring), it makes sense to design a localization subsystem that can be trained as a smaller addition to a larger P2 model (Sculley et al. 2015). This should also make it easier to iterate on the more lightweight localization module without the need to retrain the more computationally expensive P2 component.

6.4.2 Designing an LP2 System

We now explain our model design choices, highlighting the ways they overcome the challenges listed above and achieve our design goals. We show an overview of the proposed architecture in Fig. 6.3.

Input Representation. Our system receives LiDAR as the sensor input, which is then converted to a bird's-eye view (BEV) voxelization with the channels of the 2D input corresponding to the height dimension (B. Yang, Luo, and Urtasun 2018). Despite P2 and localization models both relying on some form of voxelized LiDAR input, perception-prediction models often use a coarser LiDAR resolution (e.g., 20 cm (W. Luo, Yang, and Urtasun 2018; Zeng et al. 2019)) to accommodate larger regions, while matching-based localizers typically require a finer-grained resolution to localize with higher precision (Chapter 4.3).

Using only a fine-resolution voxelization for an LP2 model would be the simplest but it imposes high runtime efficiency costs. Therefore, to accommodate these resolution differences, our method simultaneously rasterizes the incoming LiDAR point cloud $\mathbf{x}$ into two tensors of different resolutions, $\tilde{\mathbf{x}}_{\text{coarse}}$ for perception and $\tilde{\mathbf{x}}_{\text{fine}}$ for localization.

Perception and Prediction. For our P2 subsystem, we rely on ILVM (Casas, Gulino, Suo, Luo, et al. 2020), a perception method whose robustness to localization errors we quantified in Chapter 6.3; this corresponds to the lower part of Fig. 6.3. The proposed P2 approach contains four main submodules. (i) A lightweight network processes a rasterized semantic map centered at the current vehicle pose, which transforms the map from its own reference frame to the vehicle reference frame. We pass our estimated pose to this module. (ii) Another neural backbone h extracts features from a coarsely voxelized LiDAR sweep $\tilde{\mathbf{x}}_{\text{coarse}}$ which is expressed in the vehicle reference frame. These two feature maps are concatenated and passed to (iii) a detector-predictor which encodes the scene into a latent variable Z, and (iv) a graph neural network where each node represents a detected actor, and which deterministically decodes samples from Z into samples of the joint posterior distribution over all actor trajectories.

Localization. We tackle localization using the deep LiDAR embedding ground intensity localization approach discussed in Chapter 4. The idea behind ground intensity localization (Levinson, Montemerlo, and Thrun 2007) is to align the (sparse) observed LiDAR sweep $\mathbf{x}$ with a pre-built (dense) map of the LiDAR intensity patterns of the static scene, $\mathbf{m}$. This localizer learns deep functions that produce spatial embeddings of both the map $f(\mathbf{m})$ and the voxelized LiDAR sweep $g(\tilde{\mathbf{x}}_{\text{fine}})$ before alignment. Following the same parametrization as in Chapter 4, we express the vehicle pose with three degrees of freedom

(DoF), x, y, and yaw, represented as $\mathbf{x} \in \mathbb{R}^3$, and perform inference following the framework from Chapter 2.4.

Multi-Resolution Feature Sharing. An important advantage of localizing with LiDAR matching is that, in contrast to, e.g., point cloud-based localizers (Du, Wang, and Cremers 2020), it uses the same BEV input representation as P2, enabling a substantial amount of computation to be shared between both systems. However, as discussed earlier, the inputs to the P2 and localization backbones use different resolutions, which can make information fusion difficult.

We address this issue by upsampling a crop of the LiDAR feature map computed by the coarse perception backbone to match the resolution of the finer features in the localization backbone. We then add the feature maps together using a weighted sum to produce the final localization embedding, as depicted in Fig. 6.3. This allows us to compute localization LiDAR embeddings with very little runtime and memory overhead compared to the base perception-prediction network.

6.4.3 Learning

We optimize the full model using side-tuning (J. O. Zhang et al. 2020). We first train the heavier perception-prediction module, and then add the LiDAR branch of the localizer as a side-tuned module[5].

In the second stage, we freeze the weights of the perception-prediction network (yellow modules in Fig. 6.3), and only learn the map and online branches of the localizer (purple modules in Fig. 6.3). There are three benefits to this approach: first, there is no risk of catastrophic forgetting in the perception-prediction task, which can be problematic as it typically requires 3–5× more computation to train compared to the localizer alone; second, we do not need to balance the loss terms for localization vs. perception-prediction, eliminating the need for an additional hyperparameter; third, training the localizer network can be done much faster than the full system since the P2 header no longer needs to be evaluated and fewer gradients need to be stored.

Perception and Prediction. We train the P2 component using supervised learning by minimizing a loss which combines object detection with motion forecasting while accounting for the multimodal nature of the trajectory predictions. The P2 loss is, therefore, structured as

$$\mathcal{L}_{\text{P2}} = \mathcal{L}_{\text{Det}} + \alpha \mathcal{L}_{\text{Pred}}, \tag{6.2}$$

where $\mathcal{L}_{\text{Det}}$ optimizes a binary cross entropy term for the object detections and one based on smooth-ℓ_1 for the box regression parameters (B. Yang, Luo, and Urtasun 2018), $\mathcal{L}_{\text{Pred}}$ optimizes the ELBO of the log-likelihood of the inferred n trajectories over t time steps, condi-

[5] We note that the localization and the perception map embedding networks do not share any weights or feature maps. This allows us to learn the intensity map embeddings separately and even pre-compute them offline. The perception (semantic) map embeddings cannot be cached because the semantic map also encodes dynamic elements such as traffic light states.

tioned on the input (Casas, Gulino, Suo, Luo, et al. 2020), and α represents a scalar weighting term selected empirically.

Localization. Learning f and g end-to-end produces representations that are invariant to LiDAR intensity calibration, and ignore aspects of the scene irrelevant to localization[6]. Learning is performed as presented in Chapter 4 by treating localization as a classification task and minimizing the cross-entropy between the discrete distribution of $\mathbf{p}(\mathbf{x})$ and the ground truth pose offset $\mathbf{p}^{\mathrm{GT}}$ expressed using one-hot encoding:

$$\mathcal{L}_{\mathrm{Loc}} = -\sum_{\mathbf{x}} \mathbf{p}(\mathbf{x})^{\mathrm{GT}} \log \mathbf{p}(\mathbf{x}). \tag{6.3}$$

[6] Specifically, f and g correspond to the map and the online embedding networks, denoted in Chapter 4 as $f_m(\cdot; \mathbf{w}_m)$ and $f_o(\cdot; \mathbf{w}_o)$, respectively.

The map and online embedding networks f and g use an architecture based on the P2 map raster backbone and do not share weights. In Chapter 6.5, we also show that it is possible to substantially improve run time by reducing the capacity of g while keeping f fixed, with little impact on overall performance. We refer to this architecture as a *Pixor* backbone (B. Yang, Luo, and Urtasun 2018).

6.5 Experiments

We design our experiments to test the accuracy of our multi-task model on the joint localization and P2 (LP2) task. We also show how these improvements translate to safe and comfortable rides based on motion planning metrics. We refer to the task of doing localization, P2, and motion planning as LP3.

Dataset and Metrics. We use the LP3 dataset (cf. Chapter 6.3) for all our experiments. To evaluate localization accuracy, following prior work (Chapter 5), we report the percentage of frames on which the localizer matches the ground truth exactly, and where it matches the ground truth or a neighboring offset as recall @ 1 (r@1) and recall @ 2 (r@2), respectively. In our setting, the former metric corresponds to exactly matching the ground truth, up to our state space resolution (5 cm and 0.5°), while the latter corresponds to being inside a 15cm × 15cm × 1.5° region centered at the ground truth. Following Chapter 6.3, we focus on mAP@0.7 for detection and mean SFDE for prediction. For motion planning, besides collision rate and ℓ_2 distance to human trajectory, we also measure lateral acceleration, jerk, and progress toward the planning goal.

Experimental Setup. There are multiple ways to design an experiment that tests a localizer. One alternative is to start the state estimation at the identity and later align the produced trajectory with the ground truth (as is often done in SLAM (Thrun 2007)). Alternatively, online localization often initializes the robot pose at the

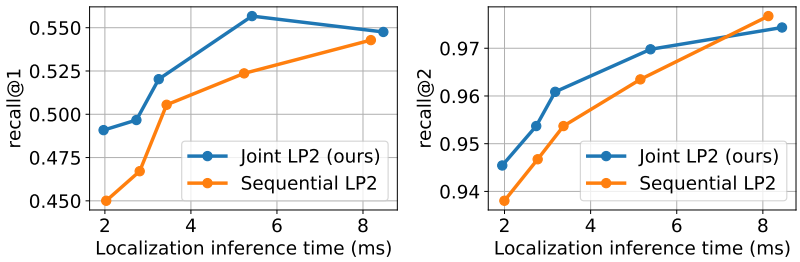


Figure 6.4: **Localizer embedding runtime vs. recall.** The localization performance and runtime of the single-task (i.e., sequential) and multi-task (i.e., joint) localization methods. Faster inference is achieved by narrower and shallower networks for the online LiDAR embedding. Note that the Y axes are focused on narrow intervals to improve visibility; on an absolute scale, the impact of reducing the network capacity is small.

ground truth location, and measures how far a localizer can travel before obtaining an incorrect pose (see Chapters 3, 4; also (Y. Zhou et al. 2020)). By definition, these setups assume that the initial pose is correct and do not test the ability to recover from localization failure—which is crucial for self-driving. Instead, we assume a self-driving scenario where the localization of the pose is initially incorrect. As such, we perturb the true pose of the vehicle and thus measure the ability of the localizer to recover from this failure, as well as the ability of P2 and MP to deal with the localization failure. The perturbations are performed following the same uniform noise policy described in Chapter 6.3.1 with 0.5 meters for translation and 1.5° for rotation.

Implementation Details. We train our model for five epochs with the Adam (Kingma and Ba 2015) optimizer using 16 GPUs. The coarse LiDAR tensor $\tilde{\mathbf{x}}_{\text{coarse}}$ is rasterized at 20 cm/voxel, while the fine tensor $\tilde{\mathbf{x}}_{\text{fine}}$ uses 5 cm/voxel. The spatial region corresponding to the coarse LiDAR voxelization is 144m × 80m × 3.2m, while the spatial region corresponding to the fine LiDAR voxelization is 48.05m × 24.05m × 3.2m[7]. Reducing the spatial extent of the high-resolution rasterization reduces the run time of the system without sacrificing performance. The localization search range covers ±0.5m relative to the initial vehicle pose estimate in the x and y dimension, discretized at 5 cm intervals, and [-1.5°, -1.0°, -0.5°, 0.0°, 0.5°, 1.0°, 1.5°] in the yaw dimension. We do not use height information from the maps, which are encoded as BEV images. When training the localizer, following Chapter 4.3, we add uniform noise to the ground truth pose up to the size of the search range.

[7] The dimensions ensure pixel-level alignment of the coarse and fine features in the presence of the upsampling connection applied to the coarse features before fusing them with the fine ones.

Localization inference time comparison. While nearly identical in terms of matching accuracy when comparing models with recall @ 2 performance similar to those presented in Chapter 4, the proposed approach is much faster due to a more efficient architecture and sharing computation with the perception backbone, as depicted in Fig. 6.4.

Results. In Table 6.1, we evaluate localization and P2 performance through motion planning metrics. Notably, despite significant varia-

Model	P2 pose (GT, N)	Planning Pose (GT, N)	r@1 ↑ (%)	r@2 ↑ (%)	Collision ↓ (% up to 5s)	ℓ_2 human ↓ (m @ 5s)	Lat. acc. ↓ (m/s^2)	Jerk ↓ (m/s^3)	Progress ↑ (m @ 5s)
ILVM	GT	GT	-	-	**2.915**	**4.64**	**2.13**	**1.82**	**24.95**
ILVM	GT	N	-	-	3.168	*4.68*	2.21	*1.83*	**24.95**
ILVM	N	N	-	-	3.511	4.70	*2.20*	*1.83*	*24.92*
Ours Joint (Tiny)	N	N	*46.6*	*93.5*	2.962	**4.64**	**2.13**	**1.82**	**24.96**
Ours Joint (Big)	N	N	**52.5**	**96.9**	*2.922*	**4.64**	**2.13**	**1.82**	**24.95**

Table 6.1: **Motion planning evaluation using different pose estimates and actor predictions.** For the P2 and Planning poses: GT denotes ground truth (the pose was not altered); N denotes that localization noise was added (translation and rotation sampled uniformly at random from $[-0.5m, +0.5m]$ and $[-1.5\,\text{deg}, +1.5\,\text{deg}]$, respectively). 'Big' refers to the largest width Pixor Embedding Net from Fig 6.4, and 'Tiny' refers to the smallest. Bold denotes the best results (within an epsilon threshold), and italics the second-best results.

tion in localization metrics, both of our localization models perform similarly well when evaluated in terms of the motion planning metrics. These results further confirm the observations from our jitter experiments (Fig. 6.2): both P2 and a short-term rollout of PLT perform similarly well when subject to a modest amount of localization error. This means that besides localization accuracy (which is important from an interpretability perspective), we have plenty of room to optimize for latency and simplicity when designing the localization component of an LP2 architecture. Our tiny Pixor-based localizer only takes 2 ms of overhead on top of the P2 subsystem, while providing a robust learned localization signal to the autonomy system.

Ablation Study. We perform an ablation study to investigate the trade-off between matching performance and inference time in the localization part of our system. We show our results in Fig. 6.4.

We compare the effectiveness of our localization network trained to reuse P2 features (i.e., the joint LP2 network) to a network trained to perform localization from scratch (i.e., as used in a sequential setting). In both cases, we achieve faster inference by using shallower (fewer layers) and narrower networks (fewer channels) for the online LiDAR embedding. The reported inference time does not include the intensity map embedding branch, which can be pre-computed offline. The largest model corresponds to an architecture similar to the P2 rasterized HD map backbone (which is itself a smaller version of the P2 LiDAR backbone), while the faster and smaller models have fewer layers or fewer channels in each layer. The four largest models have 11 convolutional layers and a factor of $C = 1/2^0, 1/2^1, 1/2^3, 1/2^4$ the number of channels as the largest model. The smallest (fifth largest) has $C = 1/2^4$ and five layers rather than 11, which corresponds to one layer around each of the three pooling/upsampling stages followed by a final layer. We observe that while reducing model size leads to a minor drop in matching accuracy, this does not end up affecting motion planning, as shown in Tab. 6.1, while at the same time reducing the online embedding computation time four-fold.

Finally, Table 6.2 compares our proposed online LiDAR embed-

Model	Time (ms)	r@1	r@2
LiDAR Localizer (Chap. 4)	25.92	0.52	0.95
LiDAR Localizer (Pixor-based)	2.79	0.47	0.95
Joint LP2 (Ours)	1.95	0.49	0.95

Table 6.2: **Localization inference time comparison.** While being nearly identical in terms of matching accuracy when comparing models with recall @ 2 performance similar to the system in Chap. 4, the proposed approach is much faster due to a more efficient architecture and sharing computation with the perception backbone.

ding networks to the state of the art. The U-Net-based approach from Chapter 4 was shown to outperform classic approaches like ICP-based localization, especially in challenging environments such as highways. Our results show that we can already match the original performance with a much faster network architecture, and that leveraging the perception feature maps allows even smaller models to perform at the same level. We measure all inference times on an NVIDIA Quadro RTX 5000 GPU[8].

Discussion. An alternative approach to tackle the LP2 problem is to train a network from scratch that balances both the localization and perception-prediction terms. We tried this but found that it performed worse than fine-tuning the localization module *a posteriori*. Understanding why this phenomenon occurs, and training joint LP2 networks from scratch is a question we leave for future work.

[8] The Quadro RTX 5000 GPU (Turing) was released in 2018, and its performance roughly matches an RTX 2080 SUPER. It has 16GB of GDDR6 and 3072 CUDA Cores. It should not be confused with the much faster RTX A5000 Ampere Generation, released in 2021, or with the likewise faster NVIDIA RTX 5000 Ada Generation, which was released in 2023.

6.6 Conclusion

While prior research in autonomous driving has explored either full end-to-end learning or the joint study of tasks such as object detection and motion forecasting, the task of localization has not received as much attention in the context of perception and planning systems, despite the strong reliance of self-driving vehicles on HD maps for these tasks.

In this chapter, we studied how localization errors affect state-of-the-art perception, prediction, and motion-planning systems. Our analysis showed that while perception is robust to relatively small localization errors, motion planning performance suffers more, especially in case of yaw errors, motivating the need to detect and correct such issues. We subsequently proposed a multi-task learning solution capable of jointly localizing against an HD map while also performing object detection and motion forecasting, and showed that localization errors can be successfully detected and corrected in less than 2 ms of GPU time.

6.6.1 Future Work in Joint Perception and Localization

Our foray into this line of research has uncovered multiple areas for improvement that may be addressed in future work. For example, it

would be interesting to evaluate our proposed architecture in a closed-loop setting with both reactive actors and counterfactual sensor inputs (Gulino et al. 2024; Z. Yang et al. 2023; Tonderski et al. 2024). We would also like to better characterize the performance of a localizer trained jointly with other autonomy components, including metrics like its localization recovery rate, its localization accuracy when enhanced with a time-aware pose filter, and its robustness to changes in the map (e.g., due to construction, road re-pavement, or seasonality). Moreover, future work could benefit from a more thorough comparison with classical localization approaches such as those based on LiDAR registration (Yoneda et al. 2014). Similarly, future work may further evaluate the motion planner by classifying the incurred collisions according to their severity (Antonante et al. 2023), which would enable a better understanding of the types of problems that arise due to localization errors. Finally, an alternative to jointly learning LP2 networks is to train P2 modules with imperfect localization as input, akin to data augmentation, while also applying regularization to avoid pathological cases of P2 modules which simply ignore the HD map altogether.

6.6.2 Limitations of the Proposed Online Localization Approaches

Despite the robustness of the localization approaches presented in the past four chapters, they are not without limitations, and several avenues of improvement remain.

- **State Space Limitations.** The proposed methods operate on a discretized state space limited to three degrees of freedom: x, y, and yaw. While this can be sufficient for many ground robotics applications, to achieve full scalability, we need robustness to factors such as vibrations, bumps, potholes, etc. This requires modeling the full 6-DoF rigid pose of the vehicles—$(x, y, z, \text{yaw}, \text{pitch}, \text{roll})$.

- **Initialization.** The proposed systems tackle online localization. They assume a good initial pose estimate located within the (limited) search range of the matcher. This setting also limits the system's ability to recover from failures, though this rarely occurs in practice.

  While we do tackle global localization next, in Chapter 7, future research could investigate whether our assumption of accurate initial location can be relaxed. Advances in this area can also aid cases where pose is lost, and recovery must be performed without a reliable initial estimate.

- **Angular Dimension.** Even though we can perform spatial matching efficiently in the spectral domain, enabling a fine-grained search

range, matching over the yaw dimension still requires an individual correlation operation for each candidate. This limits the size and granularity of the search range in this dimension. Continuously matching in the yaw dimension, as well as in the pitch and roll ones, can be done using the Fourier-Mellin transform, which has been successfully applied to various image and RADAR matching tasks in the past (Dasgupta and Chatterji 1996; Checchin et al. 2010). Spectral matching has also been shown to be effective for 6-DoF matching (Bernreiter et al. 2021). Extending these techniques to deep LiDAR matching can lead to improved accuracy and robustness by unlocking wider convergence basins for the optimization problem.

7
Large-Scale Analysis: The Pit30M Dataset

7.1 Overview

Past chapters have highlighted the critical role of localization in autonomous driving. Localization allows SDVs to navigate to their destinations and to leverage HD maps, which boosts and contextualizes downstream autonomy tasks such as perception, motion forecasting, and motion planning.

As discussed in Chapter 2, we can divide localization tasks into two broad categories: online and global. Previous chapters have focused on the former category.

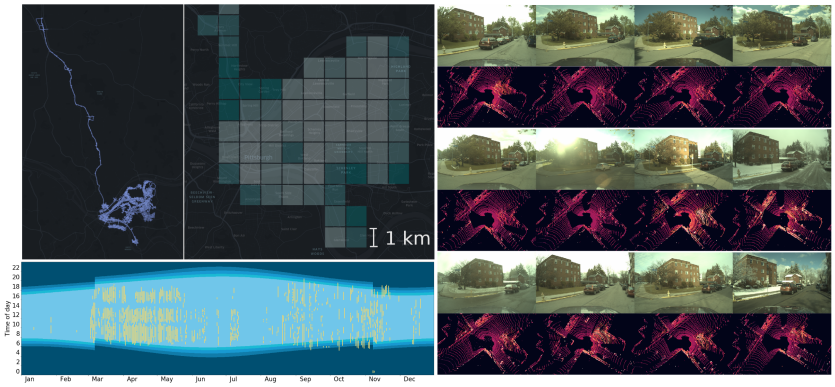


Figure 7.1: **The proposed new localization dataset, Pit30M.** Top left: The geographic extent of the dataset. Each square is 1 km^2, for a total area of about 50 km^2 plus over 20 km of highway in the Pittsburgh Metropolitan Area. Bottom left: The temporal span of our dataset. The background colors code for night, day, and astronomical, nautical, and civil twilight (resp. dawn). Right: Examples of images and LiDAR point clouds taken in the same place at different times. These trips all happened between February 2017 and March 2018.

Online localization assumes that the pose at the previous time step is known, and is tasked with propagating that information over time and adjusting it based on current sensory measurements. However, small errors may accumulate during online localization, making the pose estimate drift over time or even fail altogether. Global localization aims to overcome this issue by estimating the global pose without any assumptions from previous steps. Global localization, therefore, serves a dual function: it is used to initialize an online localization system, and it forms an important fail-safe module that allows robots to

recover from temporary online localization failures.

Autonomous driving faces unique challenges when it comes to global localization, such as fast motion, dynamic objects, and rapidly changing environmental conditions. To systematically evaluate various global localization approaches in this context, we need a benchmark that reflects the setting and its particular challenges. Ideally, the dataset employed to carry out this study should be diverse, large-scale, and have accurate ground truth over a variety of environments and traffic scenarios.

Furthermore, since autonomous driving platforms typically carry a variety of sensors that provide complementary information (such as LiDAR, camera, GPS, and IMU), the benchmark should contain multi-sensory data to enable researchers to exploit multi-modal inputs for the global localization task. Unfortunately, no existing dataset fulfills all these criteria.

In this chapter, we introduce Pit30M, a dataset that spans over a year of driving in Pittsburgh, PA, USA, comprising over 1,000 trips, 25,000 km, and 1,500 hours driven under different times of day, seasons, and weather conditions. The Pit30M dataset is accessible online under a permissive non-commercial license[1]. We provide accurate ground truth poses (under 10 cm of error) for all our data. With over 30 million images and LiDAR sweeps, our dataset is one to two orders of magnitude larger than the biggest publicly available dataset for this task. We also provide metadata such as time of day, weather, and approximate occlusion by leveraging image and LiDAR segmentation, which allows us to formally quantify the diversity in our dataset and understand localization errors. We give an overview of our dataset's geographical and temporal extent in Fig. 7.1.

[1] The dataset is available for download at https://pit30m.github.io/, and it can be accessed with the help of a Python Software Development Kit (SDK), which is open source at `https://github.com/pit30m/pit30m` and installable as: `pip install pit30m`

Due to their scalability and accuracy in the context of a dataset as dense as Pit30M, we focus on retrieval-based methods for localization in this chapter. We investigate both image- and LiDAR-based approaches to retrieval localization. For visual localization, and with our dataset's scale, diversity, and density, we find that a modern convolutional backbone with a simple pooling scheme performs on par with state-of-the-art architectures specifically designed for this task, such as NetVLAD (Arandjelovic et al. 2016). For LiDAR-based localization, we investigate both the latest network architectures and suitable point cloud representations. We show that bird's-eye view voxelization coupled with a strong convolutional backbone is competitive with the best previously proposed point cloud representations and architectures for this task—which also rely on NetVLAD pooling. Finally, we provide an analysis of the failure modes and complementarity of LiDAR- and camera-based localization.

7.1.1 Other benefits

Thanks to its scale, density, and diversity, the Pit30M dataset can be applied to a multitude of modern computer vision tasks beyond its original goal of studying localization for SDVs.

- **Novel View Synthesis.** The Pit30M dataset includes high-resolution, multi-sensor data which covers many parts of Pittsburgh multiple times under varying conditions: different lanes, times of day, weather, and occlusion levels. This setup lends itself well to analyzing modern approaches to large-scale novel-view synthesis (Tancik et al. 2022; Rematas et al. 2022; Turki et al. 2023; Kerbl et al. 2023; Yan et al. 2024) by enabling the selection of challenging validation sets with materially different viewpoints of the same physical regions. Large-scale neural rendering methods are in dire need of benchmarks for quantifying extrapolation quality; for example, prior work such as SUDS (Turki et al. 2023) are evaluated on KITTI using the same camera as the method is trained with, and only on novel timestamps, which limits the amount of extrapolation the representation is evaluated on, compared to, e.g., evaluating a new camera on a different lane[2]. Similarly, (Yan et al. 2024) showed promising results by leveraging multi-object-aware Gaussian Splatting (Kerbl et al. 2023) but only performed quantitative evaluation within the same sequence—"We select every 10th image in the sequence as the test frames and use the remaining for training". These relatively simple protocols highlight the need for new, more challenging benchmarks for NVS.
- **Vision foundation models.** Large VLMs[3] have demonstrated remarkable performance in zero-shot tasks such as classification, segmentation, or image captioning (Oquab et al. 2023) and are now a common component of commercial chatbots such as Gemini or ChatGPT. Such foundation models have likewise been explored for autonomous driving, but they are still in their infancy (A. Hu et al. 2023). Nevertheless, BEV features can generalize well, as we saw with the "anti-car" detector from Chapter 4 or the efficacy of the multi-task approach described in Chapter 6. A dataset of the scale of Pit30M has the potential to be used in the training of such models at a scale that is incomparable with what is available in other datasets, which has the potential for massive uplift in downstream tasks such as perception and prediction.
- **Unsupervised object discovery.** It has been shown that lack of labels is not a limitation when it comes to pre-training, as many methods exist that can bootstrap detectors from raw data (You, Luo, Phoo, et al. 2022; L. Zhang et al. 2023; K. Luo et al. 2024). The large number of diverse traffic scenes present in Pit30M

[2] Relighting is a task which is covered even less by benchmarks. A potential direction for future relighting benchmarks could involve being where given a sequence recorded in lighting condition A (e.g., mid-day) and *one* image in lighting condition B (e.g., at dusk). The evaluation would then involve rendering the whole sequence from a different lane and camera, using lighting condition B, which would be very challenging. However, the availability of ground truth could ensure quantifiable progress is made towards solving this task.

[3] Visual-Language Models

makes it an attractive dataset for studying how unsupervised object discovery and self-training scales with large amounts of domain-specific data.

- **Approximate Nearest-Neighbor (ANN) benchmarking.** Approximate nearest-neighbor methods benefit from testing on a wide range of domains. In computer vision, this has traditionally been done with datasets such as SIFT10M (Jegou, Douze, and Schmid 2010; Martinez-Covarrubias 2018). A large, multi-sensor dataset such as Pit30M can be used to bring ANN benchmarking to the next level through its sheer scale and multi-modality. ANN is a critical component of large-scale retrieval-augmented generation (RAG), a line of work which can be used to dramatically increase the context window size of large language models (LLMs) and vision-language models (VLMs) (Lewis et al. 2020). Advances in ANN can, therefore, bring about performance and efficiency benefits to a wide range of tasks, from robot localization to language modeling.

7.2 Current Localization Datasets

Datasets are a key component of research in large-scale localization. On the one hand, datasets that span large city areas, such as SFO-Landmarks (D. M. Chen et al. 2011) and Tokyo/Pittsburgh Street view (Torii, Sivic, et al. 2015; Torii, Arandjelovic, et al. 2015) often provide only GPS readings as reference poses[4]. Unfortunately, GPS can be inaccurate by several meters, making it difficult to quantify the error of localization methods that aim for sub-meter accuracy. This is evident in the evaluation protocol of most previous work in large-scale retrieval-based localization, which considers a database match to be correct if it is within 20 or 25 meters of the query (Torii, Sivic, et al. 2015; Arandjelovic et al. 2016; Uy and Lee 2018; W. Zhang and Xiao 2019; Z. Liu et al. 2019). This level of accuracy is insufficient for most robotic applications, including SDVs.

[4] SFO-Landmarks also considers visual overlap to compute ground truth.

Other datasets such as Cambridge (Kendall, Grimes, and Cipolla 2015) and Aachen (Sattler et al. 2012, 2018) derive ground truth from SfM[5] models (Schonberger and Frahm 2016), for which the error is difficult to quantify and, due to the computational cost of SfM, remain challenging to extend to city-scale. Urban driving datasets such as KITTI (Geiger et al. 2013), Oxford RobotCar (Maddern et al. 2017) DeepLoc (Radwan, Valada, and Burgard 2018) and NCLT (Carlevaris-Bianco, Ushani, and Eustice 2016) use robotic platforms for data collection. However, they either only cover multiple areas of the environment just once, or focus on revisiting the same route up to 100 times, limiting geographic extent. Finally, these datasets often

[5] Structure-from-Motion

derive ground truth localization from GPS and inertial filters, which do not achieve centimeter-level accuracy[6].

Recent work has used manual annotations to provide more accurate ground truth localization, either via human verification of 2D-3D matches with existing SfM models (Sattler et al. 2017) or by manually aligning LiDAR and SfM point clouds in publicly available datasets (Sattler et al. 2018). However, these annotations have remained relatively small-scale efforts, providing around 100,000 localized images in the largest case. In contrast, we aim for a dataset that provides millions of accurately-localized images and LiDAR point clouds.

[6] While RTK ground truth was eventually released for the Oxford Robotcar dataset (Maddern et al. 2020), it was not available at the time of our original analysis and publication.

7.3 Pit30M: Global Localization at City Scale

We assume that the region where our SDV is located has previously been covered with an appearance database. This dataset should ideally have three characteristics:

- **Diversity** in appearance is necessary to train models that learn to recognize the same site under changes due to weather, seasons, illumination, construction, occlusion, and dynamic objects in the scene.
- **Scale** refers to the area spanned by the dataset. We want our dataset to cover an entire city, as this is the typical operational domain of a self-driving car.
- **Accurate ground truth** provides a clear evaluation benchmark for methods that achieve sub-meter accuracy. We can also use this ground truth as a supervisory signal to improve retrieval-based localization.

	Distance (km)	Images (thousands)	Accurate GT (thousands)	Geo span (km^2)	Time span	S	LiDAR type
Pittsburgh 250k (Torii et al.)	–	250	–	$\sim$ 16	–	–	–
Tokyo 24/7 (Torii et al.)	–	*600	**1	2.56	–	–	–
SFO Landmarks (Chen et al.)	–	1 700	1 700	$\sim$ 18	–	–	(Unspecified) ✓
DeepLoc (Radwan et al.)	4	2	2	0.015	1 day	10	(Unspecified) ✓
NCLT (Carlevaris-Bianco et al.)	147	630	630	$\sim$ 1	15 m	27	(Velodyne 32) ✓
Aachen (Sattler et al., 2012)	–	5	5*	$\sim$ 1.5	–	–	
CMU (Bansal et al.)	98.7	82	82*	–	3 m	12	
Robotcar (Maddern et al.)	1 000	‡8 500	38*	$\sim$ 10	18 m	133	(2x SICK LMS) ✓
Pit30M (Ours)	25 000	30 000	30 000	$\sim$ 50	14 m	1343	(Velodyne 64) ✓

Table 7.1: **Comparison of datasets for large-scale visual localization.** ‡The dataset has >20M images, but we consider only the frontal camera to make it comparable to our dataset. *Including synthesized views. **The number of query images localized manually. S denotes the number of sessions. 'm' is short for months.

We aggregated a dataset of 1,343 trips and 30 million images and LiDAR point clouds from data collected by a fleet of self-driving cars. Our data was collected from January 2017 to February 2018 in the Pittsburgh metropolitan area, PA, USA. The vehicles carry a Velodyne HDL-64E LiDAR sensor, a wheel odometer, and an IMU, which we use to localize offline using vehicle dynamics and LiDAR registration against a pre-existing dense 3D scan of the scene geometry. These measurements are all fed to a commercial batch optimization system validated to yield under 10 cm localization error. We use a high-definition, global shutter color camera located on the roof of the vehicle, facing forward at all times, which provides images at a resolution of 1920×1200 pixels. The horizontal and vertical fields of view are $78.6°$ and $52.5°$, respectively[7]. The intrinsic and extrinsic calibration parameters of the cameras and LiDAR (e.g., the LiDAR-to-camera rigid transformation) are computed and validated a priori using a standard setup consisting of fiducial targets and non-linear optimization. We also carry a consumer-grade GPS sensor. The continuous stream of points produced by the LiDAR is broken up into 100ms partitions and motion-compensated. The corresponding camera image is selected such that it is as close as possible to the moment that the LiDAR's rolling shutter passed through the middle of its FoV. The synchronization is within a few milliseconds.

[7] The dataset itself includes six additional cameras for some of the existing logs. These include four additional wide-FoV cameras (forward-left, forward-right, rear-left, rear-right), as well as a forward-facing narrow FoV stereo pair. We did not include these cameras in our analysis at the time of publication due to computational constraints. Please refer to the Pit30M website and development kit for additional information.

Pit30M is, to the best of our knowledge, the largest benchmark for large-scale localization to date both in terms of images, readings, and accurate ground truth information. Table 7.1 provides summary statistics of existing datasets (described in the previous section) as well as ours, and Fig. 7.1 shows the extent of our data. The proposed dataset includes over 25,000 km and 1,500 hours of driving, resulting in a benchmark that is one to two orders of magnitude larger than those used in previous work. Moreover, our dataset spans all seasons, diverse weather conditions (including rain, sleet, and snow), multiple times of day, including images taken at night and with low natural lighting, as well as construction and changes in buildings and pavement.

Large-scale metadata. Previous datasets have provided manual, trip-level metadata, typically for identifying challenging conditions for localization. For example, the Oxford dataset (Maddern et al. 2017) provides 11 different tags, including "sun," "clouds," "dusk," and "snow," and the CMU seasons dataset (Sattler et al. 2018) includes tags for "park," "urban," "foliage," and "low sun," among others.

Unfortunately, trip-level tags can be ambiguous; e.g., the same trip may be sunny and cloudy at different times. Instead, we have collected more granular metadata using historical weather and astronomical data, which can be obtained at scale. In particular, we have collected

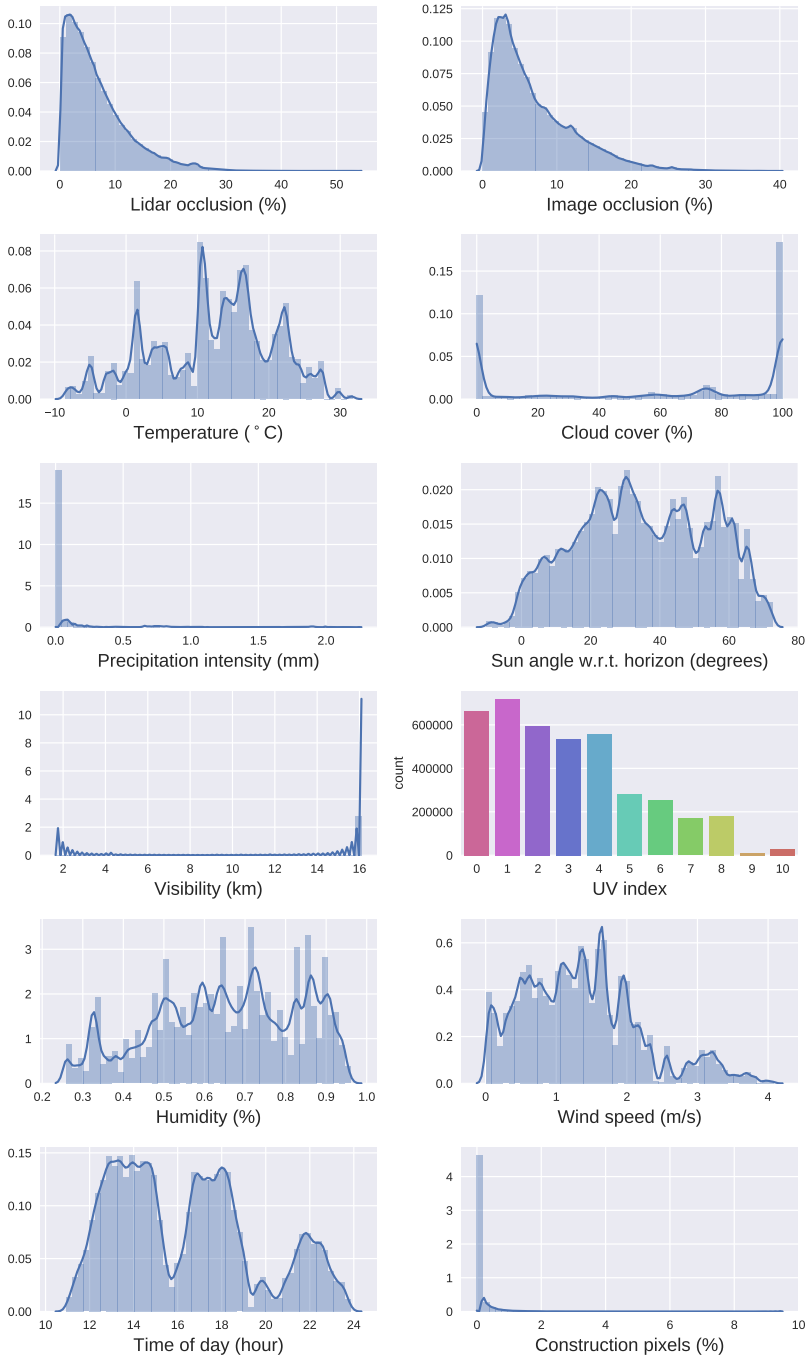


Figure 7.2: **Probability density functions (PDFs) for metadata in Pit30M.** For a complete description of these tags, please refer to Table 7.2

weather via the darksky.net public API, and estimated the sun's angle in the sky using the `skyfield` library[8]. We have also used state-of-the-art LiDAR (C. Zhang, Luo, and Urtasun 2018) and image (Bai and Urtasun 2017) semantic segmentation to estimate the degree of background occlusion in our dataset. We showcase our labels by analyzing the results of our preliminary benchmark in Chapter 7.5. Fig. 7.2 shows the probability density functions of some of our tags. We provide a detailed description of the collected data in Table 7.2.

[8] While the darksky service has shut down completely as of January 1st, 2023, following its acquisition by Apple, all historic data for Pit30M is still available.

Field	**Units**	**Description**
Time of day	Datetime	Unix timestamp.
Angle of the sun w.r.t. horizon	Degrees	Angle between the horizon and the center of the sun, as observed from the location of the particular sensor reading. This value is zero at both dawn and twilight and is commonly used to estimate light for navigation purposes in, e.g., nautical applications. The value becomes negative when the sun is below the horizon relative to the observer.
Precipitation	Millimeters	An amount larger than zero means snow or rain is present.
Visibility	Kilometers	Distance up to which objects can be clearly discerned. Heavy fog may result in low visibility.
UV index	Integer	Integer value indicating the intensity of solar UV rays.
Temperature	Degrees Celsius	Ambient temperature.
Humidity	Percent	The amount of water vapor present in the air, as a percent of the maximum that the air could hold at the same temperature.
Cloud cover	Percent	Percent of the sky covered by clouds.
Wind speed	meters per second	Speed of the wind.
Image occlusion	Percent	Percent of pixels taken by dynamic objects in the scene.
LiDAR occlusion	Percent	Percent of LiDAR points taken by dynamic objects in the scene.
Construction	Percent	Percent of image pixels that correspond to active construction elements.

Table 7.2: **Semantic labels in Pit30M.** We provide these labels to help researchers categorize and understand the performance of localization algorithms.

7.4 Case Study: Benchmarking Large-Scale Global Localization

We turn our attention to benchmarking retrieval-based localization approaches. Formally, let $\mathcal{Z}$ be either an image or LiDAR sweep point cloud, let $\mathcal{G}$ be the GPS pose, and $\mathbf{y}$ the position of the SDV we are trying to infer.

In the retrieval localization setting, we start from a dataset of pre-localized sensor observations, and represent each full sensor reading as a D-dimensional vector $\mathbf{z} = f(\mathcal{Z}) \in \mathbb{R}^D$, to obtain a database of vector-pose pairs $\mathcal{D} = \{(\mathbf{z}_1, \mathbf{y}_1), (\mathbf{z}_2, \mathbf{y}_2), \ldots, (\mathbf{z}_n, \mathbf{y}_n)\}$. Given an online

(query) sensor reading $\mathcal{Z}_q$, we first compute its global feature representation $\mathbf{z}_q = f(\mathcal{Z}_q)$ and infer the current pose via (high-dimensional) nearest neighbor search:

$$\hat{\mathbf{z}} = \underset{\mathbf{z}_i}{\operatorname{argmin}} \|\mathbf{z}_q - \mathbf{z}_i\|_2^2 , \tag{7.1}$$

outputting the pose associated with the nearest dataset descriptor $\hat{\mathbf{z}}$. Since each observation is associated with a single vector $\mathbf{z}_i$, the obtained pose is expressed in a global coordinate frame.

We now introduce a couple of simple yet effective retrieval convolutional networks for computing the embedding vectors $\mathbf{z}$. By leveraging the supervision provided by our dataset, we show that strong convolutional backbones with simple pooling schemes can match the state of the art in image and LiDAR retrieval at the time of publication. This allows us to showcase the importance of our data and the gains brought about by its scale. This can give us insights into the state of the art of retrieval-based localization for robotics.

Learning for retrieval. We rely on deep convolutional networks trained with a standard triplet loss, common in the retrieval literature:

$$\mathcal{L}_{\text{retrieve}} = \max\{d(\mathbf{a}, \mathbf{p}) - d(\mathbf{a}, \mathbf{n}) + m, 0\} , \tag{7.2}$$

where $d(\cdot, \cdot)$ is the Euclidean distance function and the triplet $(\mathbf{a}, \mathbf{p}, \mathbf{n})$ consists of three latent, ℓ_2-normalized embeddings produced by our embedding function $f(\mathcal{Z})$. In this context, $\mathbf{a}$ is an "anchor" descriptor, $\mathbf{p}$ is a "positive" descriptor, and $\mathbf{n}$ is a "negative" descriptor. We build our triplets such that the geo-location of the positive image is closer to the anchor than the negative image by a pre-defined margin of at least $m = 0.5$ in embedding feature space. In our experiment, we consider sensor readings within one meter to be positives and within two to four meters to be negatives; notice that this fine-grained distinction is enabled by the accurate ground truth provided by our dataset.

We also ensure that the heading angle of the three images is within a range of 30°, so the images have overlapping fields of view. Finally, we ensure that the positive and negative samples do not come from the same trip as the anchor, which encourages the learned representations to be invariant to factors such as time of day, weather, and dynamic objects in the scene. We collect a triplet for each image in the dataset and learn $f(\cdot)$ via backpropagation. We implement f in practice as a ResNet-50 (K. He et al. 2016).

GPS + retrieval. We also consider using GPS to restrict the search area. This is a more realistic scenario in the context of self-driving, yet it is less studied in the literature. Here, we collect a set of embeddings located within τ meters from the GPS reading, where τ is a tunable hyperparameter that we set based on the empirical error

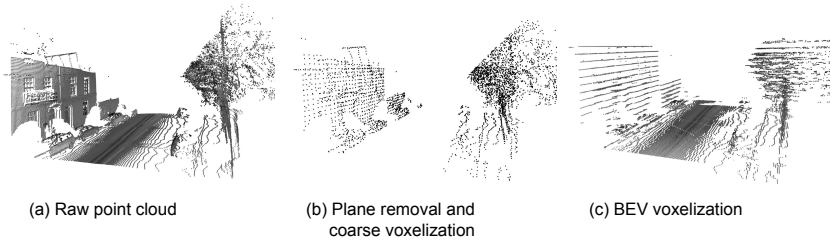


Figure 7.3: **LiDAR representations benchmarked in this work.** (a) Raw point cloud (not used by any method). (b) Point cloud after ground plane removal and downsampling to 4,096 points (Uy and Lee 2018; W. Zhang and Xiao 2019; Z. Liu et al. 2019). (c) BEV voxelization with intensities. We use the latter as input to CNNs.

of our GPS measurements[9]. This filtering can be done efficiently with a KD-tree, as both the database locations and the GPS readings are low-dimensional. We then perform retrieval as in the global case, only restricted to this region.

Bird's-Eye View (BEV) representation. Although using CNNs with images in the above formulation is straightforward, it is not immediately clear how that can be achieved when the input is a point cloud. For this, we introduce a conceptually simple representation that achieves state-of-the-art results on publicly available benchmarks. (Please see Chapter 7.5.5 for detailed experiments on the Oxford Robotcar (Maddern et al. 2017) dataset.) We preprocess the raw LiDAR point cloud as a BEV multi-channel representation by discretizing 3D space into an evenly-spaced voxelization of size $l \times w \times h$. We use the forward-left-up convention, so x represents the direction the vehicle is facing, y points to the left, and z denotes height. Crucially, we treat the resulting voxelization as a 2D image by discretizing the z-axis into c channels, allowing it to be plugged directly into standard 2D CNNs. This representation has proven useful for efficient real-time LiDAR object detectors (Xiaozhi Chen et al. 2017; W. Luo, Yang, and Urtasun 2018), and with our approach, we show that it also produces competitive results on LiDAR retrieval. In contrast, previous retrieval work has operated directly on the point clouds, which are heavily downsampled for computational reasons (Uy and Lee 2018; Z. Liu et al. 2019; W. Zhang and Xiao 2019). We visualize different LiDAR representations in Fig. 7.3 and present additional implementation details for generating them below.

Oxford Robotcar dataset. We preprocess the LiDAR with a resolution of 8 pixels/meter in width and height, in an area of 40m for the x-axis and 25m in the y-axis centered approximately around the position of the car in the middle frame. Along the z-axis, we discretize 10m into 16 channels, resulting in a BEV image with $320 \times 200 \times 16$ voxels. We obtain the intensity for each voxel by averaging the intensities of all the LiDAR points within the voxel. Finally, we pass these BEV images through a standard ResNet-50 architecture (K. He

[9] We set $\tau = 20$m in our main results, which is a conservative yet effective error bound for an inexpensive consumer-grade GNSS unit operating in environments where satellite reception is present but potentially very noisy, such as urban canyons.

et al. 2016), except that we use max- instead of average-pooling in the final layer.

Pit30M dataset. We use different parameters in this dataset, as the LiDAR sensor produces a 360° point cloud around the car with a larger reach than the 2D sensors used in the Oxford dataset. The car itself casts a large shadow in the point cloud, which induces further domain shift. We discretize an area of 200m in the x-axis and 125 m in the y-axis, with a resolution of 2.4 meters per pixel. The area is centered around the vehicle. We discretize 3.2 meters in the z-axis into 16 channels. We use a ResNet-50 backbone (K. He et al. 2016) with average pooling and follow the training procedure described earlier in this chapter.

7.5 Case Study: Experiments

7.5.1 Evaluation Protocol

We split Pit30M in terms of different trips. This partition protocol suits the self-driving scenario well, where a fleet typically maps the drivable areas beforehand, but new trips face changes in visual appearance and new dynamic objects on the road. We randomly select 941 trips for training, 134 for validation, and 268 for testing. We further select 10,000 random query sensor readings from the test partition to report our final localization metrics. We report the percentage of correctly localized queries for increasing distance ranges. This results in a monotonically increasing curve for increasing distance, with a hypothetical perfect localizer having a performance of 100 for every distance value.

7.5.2 Benchmarked methods

Camera-based methods. From classical methods, we consider SIFT-based VLAD (Jégou et al. 2011) and DenseVLAD (Torii, Arandjelovic, et al. 2015), a variant of VLAD specifically designed for very dense datasets with high visual variability. For these two methods, we learn the visual vocabulary on a subset of 5M images from the training set and use 128 SIFT clusters.

In VLAD, following the same pipeline design as the authors, we detect keypoints using a Hessian-based affine-invariant detector (Mikolajczyk et al. 2005) and describe them using SIFT descriptors. We build the visual vocabulary using k-means clustering and aggregate fixed-dimension VLAD descriptors as described in the original paper.

DenseVLAD follows a similar approach, except that instead of relying on a particular keypoint detector, we sample keypoints in every image following a dense grid. Following the original paper, the

DenseVLAD descriptors are sampled every two pixels, using patch sizes of 16, 24, 32, and 40 pixels. For both VLAD and DenseVLAD, we use 128 visual words learned on a subset of 5M of the training set features to compute the descriptors, which are then projected to a smaller dimension using PCA.

We also consider NetVLAD (Arandjelovic et al. 2016), a deep learning approach that uses VGG-16 convolutional feature maps as local features, and adds a learnable VLAD-based pooling layer. We use the TensorFlow implementation[10] of (Cieslewski, Choudhary, and Scaramuzza 2018). Since previous work (Sarlin et al. 2019) has relied on the best model from (Arandjelovic et al. 2016) trained on Pittsburgh[11] for global localization, we also benchmark this pre-trained network out-of-the-box on our dataset to provide context about the performance of a strong baseline in the field. We call this method NetVLAD-OOB.

[10] https://github.com/uzh-rpg/netvlad_tf_open

[11] NetVLAD implementation: https://www.di.ens.fr/willow/research/netvlad/. The Pittsburgh (Pitts250k) is from (Torii, Sivic, et al. 2015).

LiDAR-based baselines. We consider PointNet-Max (Qi, Su, et al. 2017), PointNetVLAD (Uy and Lee 2018), and PCAN (W. Zhang and Xiao 2019) and train them on the Pit30M dataset. We use the publicly-available implementations of PointNetVLAD[12] (Uy and Lee 2018). We also implemented our own version of PCAN (W. Zhang and Xiao 2019). Note that we do not evaluate the state-of-the-art LPD-Net (Z. Liu et al. 2019) on Pit30M due to the lack of a public implementation. However, as shown in Chapter 7.5.5, benchmarking on the Oxford datasets shows that our BEV + ResNet50 method is competitive with this strong baseline.

[12] https://github.com/mikacuy/pointnetvlad

7.5.3 Results

Quantitative results. We show the results of our retrieval-based benchmark in Fig. 7.4 and detailed results in Fig. 7.3. Regarding image retrieval, while VLAD is outperformed by both DenseVLAD and NetVLAD–OOB, the differences between the hand-crafted DenseVLAD and deep NetVLAD are rather small, and no clear winner emerges. Our image-based ResNet50 baseline performs on par with NetVLAD. In LiDAR retrieval, BEV emerges as the best method overall, outperforming all its image counterparts and other LiDAR methods.

Qualitative results. We show some qualitative results for the retrieval-based methods on Pit30M in Fig. 7.5. We focus on challenging queries featuring glare, low sun angles, snow, and rain. For example, the second row shows a query with glare, where both ResNet and NetVLAD manage to localize correctly, but DenseVLAD fails. The bottom row shows an interesting example where snow makes LiDAR highly reflective; however, all LiDAR methods are able to localize within 5m.

% within (metres)	0.25m	0.50m	1.0m	5.0m	average	median
GPS	0.4	1.7	6.2	73.7	4.20	3.40
Image-based methods						
VLAD	8.59	20.01	33.44	51.40	2401.33	3.95
DenseVLAD	14.50	34.12	56.15	77.82	843.98	0.81
NetVLAD-OOB	13.85	32.47	55.27	82.38	476.33	0.84
NetVLAD	42.57	74.38	86.07	87.60	577.69	0.29
Resnet50 (Img)	45.40	78.51	90.39	91.87	418.64	**0.27**
GPS + VLAD	10.44	24.78	43.58	78.80	3.53	1.26
GPS + DenseVLAD	15.33	36.47	61.33	90.78	2.05	0.72
GPS + NetVLAD-OOB	14.47	33.88	58.24	90.87	2.05	0.77
GPS + NetVLAD	43.53	77.78	92.73	96.80	0.92	0.28
GPS + Resnet50 (Img)	45.74	79.79	93.49	97.06	0.86	**0.27**
LiDAR-based methods						
PointNet Max	36.53	66.46	78.82	80.77	944.28	0.34
PointNetVLAD	51.60	83.29	91.89	93.16	322.85	0.24
PCAN	48.95	81.88	91.51	92.47	339.96	0.26
BEV + Resnet50	60.17	86.08	91.39	92.56	353.27	**0.20**
GPS + PointNet Max	37.73	70.03	86.70	91.95	1.78	0.32
GPS + PointnetVLAD	50.77	82.43	91.27	94.35	1.42	0.25
GPS + PCAN	48.43	80.77	90.63	93.39	1.55	0.26
GPS + BEV + Resnet50	59.38	84.88	91.25	93.74	1.51	**0.21**

Table 7.3: **Detailed localization results for retrieval-based approaches.** We report the percent of correct predictions within different distance thresholds, and mean and median over the entire query set. Top: Image-based methods. Bottom: LiDAR-based methods.

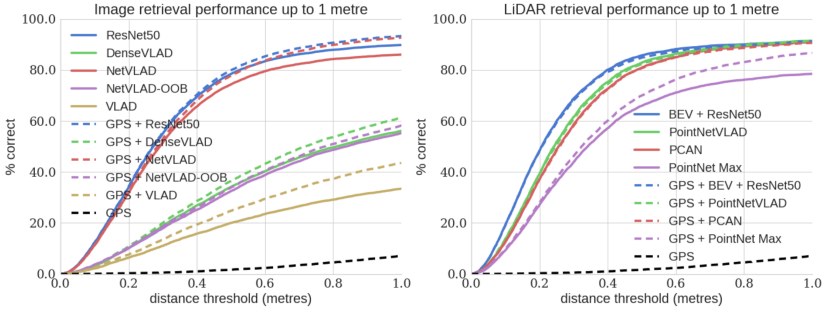


Figure 7.4: **Performance of retrieval-based methods.** Left: Image retrieval results. Right: LiDAR retrieval results.

In Fig. 7.8–7.13, we show additional qualitative results with challenging conditions for localization, such as images with snow, rain, low sun angle, and high occlusion. We also show some failure modes for both images and LiDAR.

Figure 7.5: **Qualitative results under exhaustive search.** Left: Query. Middle: Image retrieval method. Right: LiDAR retrieval methods. The insets display the error between the retrieved result and A digital copy is recommended, and zooming in is encouraged.

7.5.4 Analysis

We use the annotations provided by Pit30M to analyze our benchmarking results. We focus on understanding our best-performing methods, GPS+Resnet-50 (images) and GPS+BEV+Resnet-50 (LiDAR).

First, in Fig. 7.6 (left), we observe that large GPS errors tend to cause large overall localization errors for both images and LiDAR. This is unsurprising since we limit the search radius to 20m around the GPS prediction. In Fig. 7.7, we show the pairwise Pearson correlation coefficient between our labels and failure cases of image and LiDAR retrieval (a query is considered a "failure" if its error exceeds

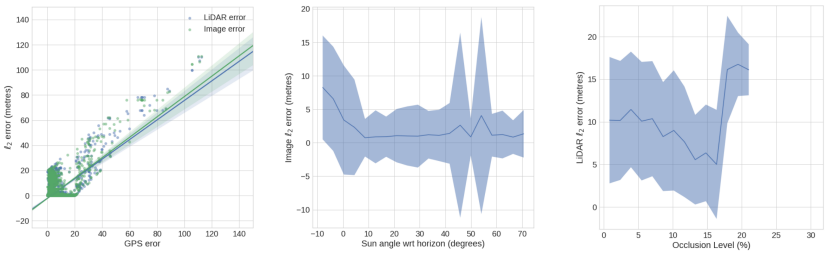


Figure 7.6: **Examples of analysis enabled by the Pit30M metadata.** Left: GPS error correlates with both image and LiDAR localization errors. Middle: Image localization error vs. sun angle in the horizon (altitude angle). We observe a smooth error increase as the sun approaches the horizon. Right: We plot LiDAR queries with more than 1 meter of error (failure cases) against LiDAR occlusion. We observe a sharp spike in error when between 15 and 20% of points correspond to dynamic objects.

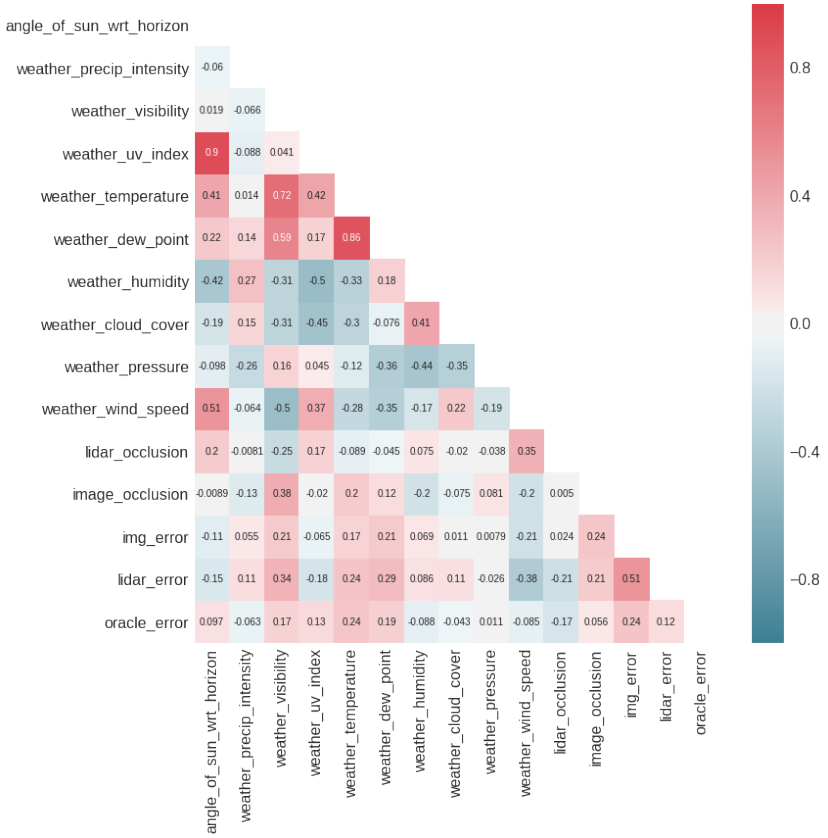


Figure 7.7: **Pairwise correlations between metadata in Pit30M and error of different methods.** "Oracle error" stands for a hypothetical method that can pick the best of either image or LiDAR prediction for each query.

2m), after removing cases where GPS error is above 20m. While we observe that, for example, image error is correlated with image occlusion, and image and LiDAR errors are highly correlated, this analysis is somewhat limited, as it is only able to capture linear correlations. Thus, we further examine image error vs. sun angle in Fig. 7.6 (middle); here, we observe increased errors during dawn and twilight and no effect during daytime. We also examine LiDAR error vs. occlusion, and observe a spike in errors when 15-20% of points are assigned to dynamic objects, indicating that enhanced robustness to occlusion has the potential to further boost the accuracy of the localizers we studied.

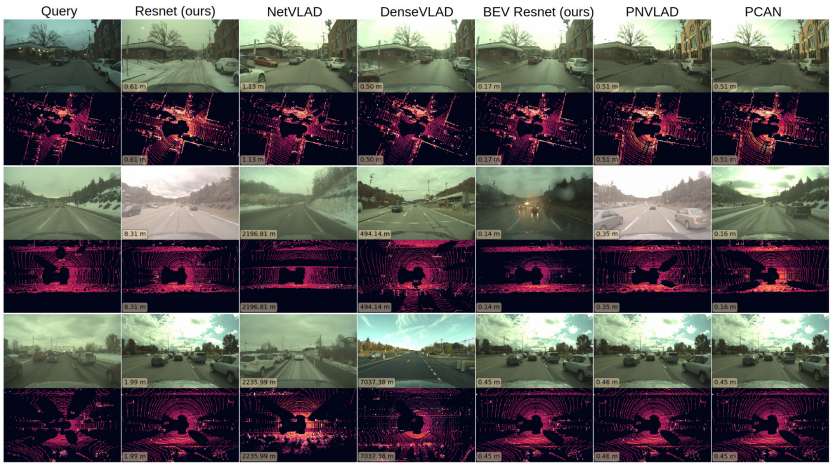


Figure 7.8: **Results with snow.** The second and the third examples show NetVLAD and DenseVLAD struggling with cross-seasonal matches.

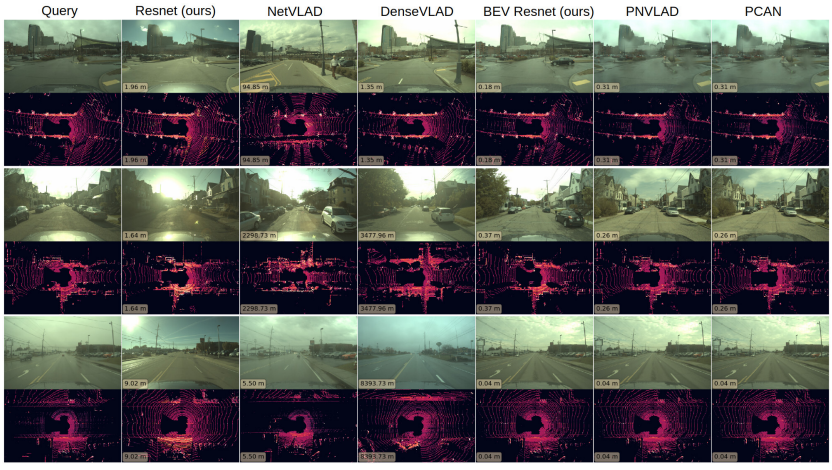


Figure 7.9: **Results with low sun angle.** The second example shows a successful ResNet match, despite the low sun clearly visible in the frame.

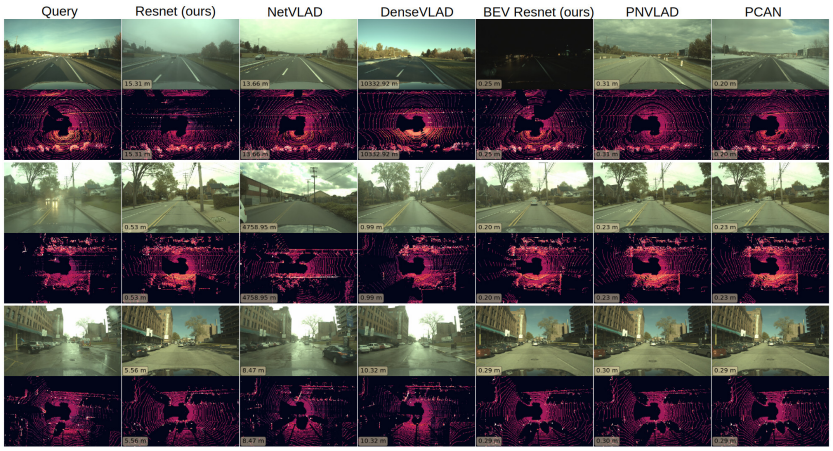


Figure 7.10: **Results with rain.** In the second and third examples, we see that the heavy rain in the query affects the matching quality in the image networks.

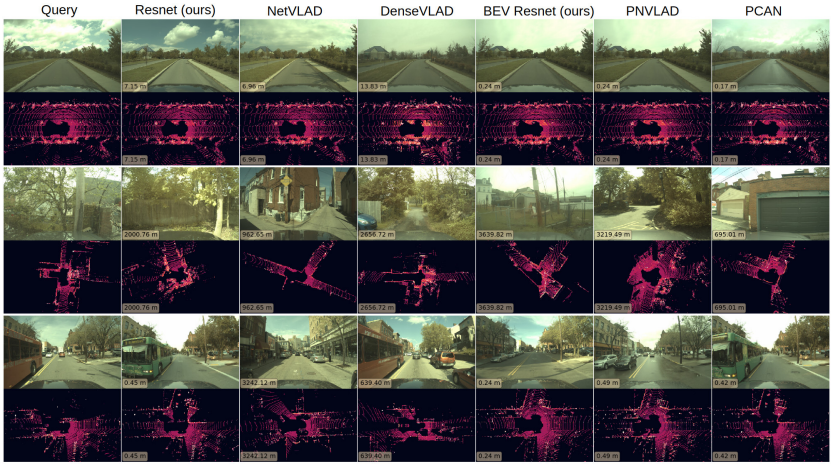


Figure 7.11: **Results with occlusion.** The first example demonstrates the difficulty in precise image retrieval with few landmarks in the image. The second example shows retrieval failures across the image and LiDAR networks, likely caused by the atypical location and heavy vegetation.

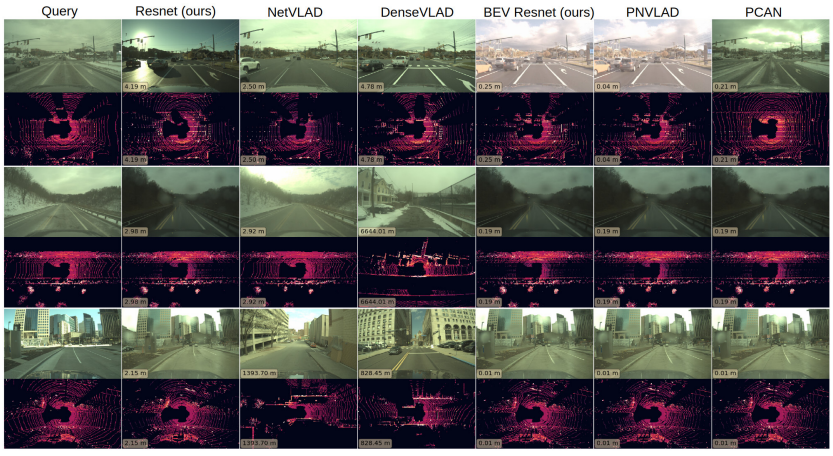


Figure 7.12: **Results with multiple challenging modalities.** The first example shows a low-light query with snow covering the ground, while the last example shows both rain and sunshine, which NetVLAD and DenseVLAD have trouble handling.

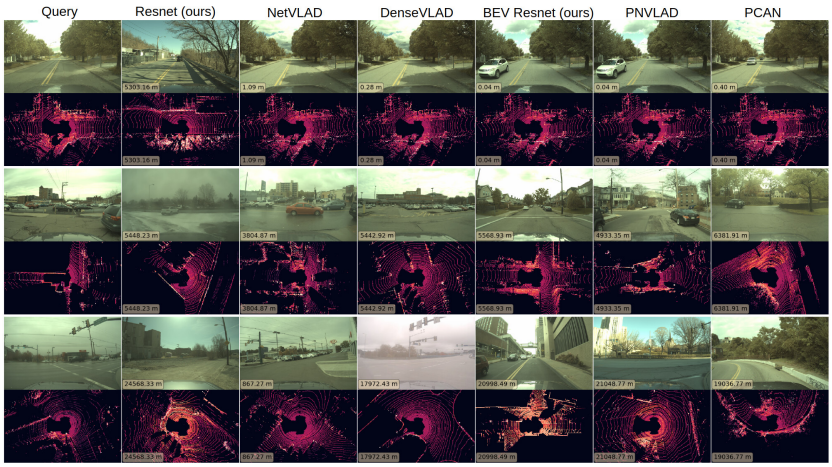


Figure 7.13: **Failure cases.** The second and third examples cause failures in both LiDAR and image retrieval, presumably due to the lack of distinctive landmarks in the sensor readings.

7.5.5 Oxford Robotcar

We benchmark different LiDAR representations on the Oxford RobotCar dataset (Maddern et al. 2017) which contains over 100 trips and 1,000 km driven in the city of Oxford, UK over the span of a year. The vehicle carries two 2D LiDAR sensors that scan the scene as the car moves through it. This creates a 3D point cloud, as shown in Fig. 7.3. Unfortunately, the dataset does not provide accurate ground truth for its sensor readings, as it only provides GPS locations[13]. Nonetheless, to the best of our knowledge, all previous work on LiDAR-retrieval-based localization has been benchmarked on this dataset (Uy and Lee 2018; Z. Liu et al. 2019; W. Zhang and Xiao 2019).

We use the publicly available code by (Uy and Lee 2018) to generate the training and test partitions on the dataset[14]. For each session, we aggregate the 2D LiDAR scans using vehicle dynamics to build a full reference map, and associate them with global coordinates from the GPS readings. Next, we generate train and test sub-maps by using two geographically disjoint sets of reference maps, with each sub-map containing all the LiDAR points collected over 20m of driving. For the training set, positive sub-maps are defined as being at most 10m from the query sub-map, while negative sub-maps are defined as being at least 50m away from the query.

In-house datasets. Previous work has also typically benchmarked on the three "in-house" datasets provided by (Uy and Lee 2018). Unfortunately, only heavily downsampled point clouds are available online. We contacted the authors asking for the full point clouds, but were told that they are no longer available, so it was not possible to benchmark our approach on them.

[13] As of 2023, the authors also released high-quality RTK ground truth for the dataset (Maddern et al. 2020), but it was not available at the time of the Pit30M publication.

[14] github.com/mikacuy/pointnetvlad

	r@1%	r@1	Inference
PointNet-Max (Charles R. Qi et al., 2016)	73.87	54.16	–
PointNetVLAD (Uy and Lee, 2018)	81.01	62.76	13.09 ms
P-CAN (W. Zhang and Xiao, 2019)	86.40	70.72	–
LPD-Net light (Z. Liu et al., 2019)	89.55	77.92	18.88 ms
LPD-Net (Z. Liu et al. 2019)	<u>94.92</u>	**86.28**	23.58 ms
BEV + Resnet50 (ours)	**95.10**	<u>86.13</u>	8.41 ms

Table 7.4: **Comparison of LiDAR-based retrieval methods on the Oxford RobotCar dataset**. Our method achieves competitive results at lower inference times. All times benchmarked on an NVIDIA GTX 1080Ti GPU.

Evaluation protocol. We follow the standard evaluation procedure for Oxford RobotCar, where a sub-map match must be within 25m of the query to be considered correct. As such, the dataset contains 21,711 sub-maps for training and 3,030 sub-maps for evaluation. We generate the train and test sub-maps by splitting the corresponding reference maps at intervals of 10m and 20m, respectively. For the baseline networks (Qi, Su, et al. 2017; Uy and Lee 2018; W. Zhang and Xiao 2019), the raw point clouds are preprocessed by removing the ground planes from each sub-map, and downsampled to 4,096 points using a coarse voxel filter. We follow previous work and report average recall@1, and average recall@1%, meaning whether a correct match is retrieved as the top result, or within the top 1% of the retrieved maps, respectively, averaged over all queries.

Training details. Our network is trained with a lazy quadruplet loss (Uy and Lee 2018), and each batch consists of one anchor, two positives, and 18 negative examples. Finally, we refresh the representation cache used for hard negatives every 1,000 iterations. We train our network using the Adam optimizer (Kingma and Ba 2015), with an initial learning rate of 0.001, which we decrease by a factor of ten every time the validation accuracy plateaus. Our network takes roughly eight hours to train on a single 1080Ti GPU.

Results. We report results in Tab. 7.4. Our method is competitive with the state-of-the-art LPD-Net (Z. Liu et al. 2019), while being much faster at inference time. Notably, our method achieves this by using a strong backbone but without relying on NetVLAD pooling (unlike all our baselines). We believe that this result shows the effectiveness of BEV + CNN representations for LiDAR-retrieval-based localization.

7.6 Practical Matters: Releasing a Petabyte-Scale Dataset

Releasing a public dataset for the benefit of the research community involves many labor-intensive steps beyond the initial data selection and experimental validation. The dataset must be organized in an easy-to-use structure, leveraging open standards; it should come with an SDK[15] which provides convenient access to the dataset, and care

[15] Software Development Kit

must be taken to remove personally identifiable information from it.

Furthermore, despite not being discussed very much in academia, promoting a dataset and proactively building a community around it is perhaps just as important to its success as the actual content or scale, especially in the modern era, where there are already many computer vision datasets to choose from, whether they focus on robots (Maddern et al. 2017; Geiger et al. 2013; Mao et al. 2021; Sun et al. 2020; Wilson et al. 2023; Pitropov et al. 2021; Burnett et al. 2023) or not (Deng et al. 2009; Sattler et al. 2018; A. Bansal, Badino, and Huber 2014).

This section will cover these challenges in depth, and discuss the solutions we used for Pit30M. We hope that this information will be helpful to other researchers or engineers interested in releasing a large-scale public dataset.

It is worth noting that the Pit30M SDK, unlike this document, is constantly evolving. For the latest information please refer to the official website at pit30m.github.io.

7.6.1 Format and Structure

Finding the right balance between simplicity and complexity is critical when designing a dataset format.

Simple designs typically involve individual files, such as images stored under a directory tree. An index file specified in a standard format such as CSV may be used to store information such as ground truth, the dataset split to which a sample belongs, as well as metadata like the weather and camera ID. Further information, such as robot calibration, may be stored as plain text files. Given its ubiquity, NumPy may also be used to store non-image data, such as LiDAR, thanks to its efficiency, openness, cross-language support, and flexibility.

More complex formats may also be used. For example, `Zarr` (Miles et al. 2020) or `tfrecord` (Abadi et al. 2016) files could enable enhanced neural network training throughput by grouping the samples into large chunks which may be quicker to read from a network file systems than individual files. However, given the need to shuffle data for ML training, the advantages of large chunks are relatively small. Robotics-specific formats such as `rosbag` or the recent `mcap` format [16] allow structured access to multi-modal time-series robot data but tend to be very inefficient for random access, making them a poor choice for datasets focused on training ML models.

[16] mcap.dev

While proprietary formats such as MATLAB's `.mat` are yet another option, they should be avoided because of their poor cross-platform support and the need to purchase an expensive software license in

order to access advanced functionality.

For Pit30M we opted to use a flat file structure, organizing the files by driving log. As the Pit30M global localization benchmark involved heavily downsampling the raw data, we built simple yet effective index files to represent the "main" global localization dataset files. Each index contains the GT pose, log ID, and path to an image or LiDAR file.

The dataset is thus organized by the ID of the driving log, and each log contains the raw sequential data from each sensor in a separate directory. The dataset split indexes simply point to the specific samples we selected to make up the Pit30M global localization benchmark, though users can always access the raw data directly using the SDK. This can be useful for analyzing tasks like online localization, SLAM, or novel view synthesis in the future.

The general structure of the dataset, which can be accessed directly in its corresponding AWS S3 bucket even without a dedicated SDK (more details to follow), is as follows:

```
pit30m/
├── ...
├── <log_uuid>/
│   ├── cameras/
│   │   ├── hdcam_02_starboard_front_roof_wide/
│   │   │   ├── index/
│   │   │   │   ├── index_v02.npz
│   │   │   ├── 000/
│   │   │   │   ├── 00000.day.webp
│   │   │   │   ├── ...
│   │   │   ├── ...
│   │   │   ├── nnn/
│   │   │   │   ├── nnn00.npz.lz4
│   │   │   │   ├── nnn01.npz.lz4
│   │   │   │   ├── ...
│   │   ├── hdcam_04_starboard_rear_roof_wide/
│   │   │   ├── ...
│   │   ├── hdcam_08_port_rear_roof_wide/
│   │   │   ├── ...
│   │   ├── hdcam_10_port_front_roof_wide/
│   │   │   ├── ...
│   │   ├── hdcam_12_middle_front_roof_narrow_left/
│   │   │   ├── ...
│   │   ├── hdcam_12_middle_front_roof_narrow_right/
│   │   │   ├── ...
│   │   ├── hdcam_12_middle_front_roof_wide/
```

```
│   │   │   ├── ...
│   ├── lidars/
│   │   ├── hdl64e_12_middle_front_roof/
│   │   │   ├── index/
│   │   │       ├── ...
│   │   │   ├── 000/
│   │   │   │   ├── 00000.npz.lz4
│   │   │   │   ├── ...
│   │   │   ├── ...
│   │   │   ├── nnn/
│   │   │   │   ├── nnn00.npz.lz4
│   │   │   │   ├── nnn01.npz.lz4
│   │   │   │   ├── ...
│   ├── all_poses.npz
│   ├── raw_calibration.yml
│   ├── raw_imu.npz.lz4
├── <log_uuid>/
│   ├── ...
├── ...
```

The LiDAR is stored as a self-describing NumPy file containing N entries for the N points in a sweep. Both LiDAR and camera sub-directories include indexes which associate each image or sweep with a timestamp, some metadata like the shutter time, and the ground truth pose of the SDV at that time[17]. Please refer to the SDK for detailed information on the structure of such an index. Chapter 7.3 provides additional details on the image and LiDAR data incorporated into the dataset.

[17] Please note that the samples corresponding to the held-out test set will contain blank values instead of poses.

We also group every 100 samples into their own sub-directory to reduce the risk of slowdown associated with listing gigantic prefixes.

7.6.2 Hosting

Depending on the scale of a dataset, simply finding a way to make the raw data available can be challenging. While large companies like Google can easily afford to host these kinds of datasets, providing access to such data can be much more difficult for small companies or academic labs.

While simply hosting tarballs on a university server may work, if the download speeds are very low, the likelihood of dataset adoption drops as other researchers gravitate towards easier-to-use datasets. Additionally, depending on the scale of the dataset, it may be desirable to allow users to programatically download specific parts of a dataset, for example, a "mini," "medium," and "full" version, to suit their storage and compute budget. Most research projects do not

require the entire dataset.

Another option is to use a third-party hosting server, such as Amazon's S3. However, hosting 300 TB in AWS can cost tens of thousands of dollars per month, especially when considering additional network transfer costs. Fortunately, there are options for providing free access to host an academic dataset, using programs such as AWS Open Data. This is the program we selected for Pit30M given the sheer scale of the dataset.

We applied for hosting under this program, and Pit30M was accepted with a maximum storage capacity of 300 TB. With the judicious use of lossy image compression[18], we were able to hit this goal without removing any of the raw log data. Furthermore, by ensuring our samples are compressed, we can also maximize the throughput of reading the data at training time, eliminating the need to download large PNGs, which can quickly become a bottleneck when training at scale.

7.6.3 Anonymization

It is important to remove personally identifiable information from the dataset. Because the dataset involves driving on public roads, we need to ensure all sensitive information present in camera images, namely faces and license plates, is anonymized appropriately by blurring. None of the other modalities contained any sensitive information. The LiDAR point clouds are much too coarse to allow faces or any sort of writing to be recognized.

The anonymization process consists of two steps: detecting the sensitive information, and removing it. For the former, we leveraged per-frame object detection, while for the latter we relied on a strong Gaussian blur[19].

While preparing the dataset, we identified the open-source Understand AI anonymizer[20] as the most promising tool for detecting and blurring sensitive information due to its simplicity, high-quality results, and easy-to-use API.

Nevertheless, as we continued working on generating the public version of the dataset, we realized that, while effective, this model was much too slow, taking over 100 ms to process at 1080p on an NVIDIA RTX 3090 GPU. Furthermore, the package was unmaintained and relied on a relatively old version of TensorFlow. To complicate things further, the object detector and object blurring components were tightly coupled, and the actual model architecture was obfuscated. This made the library challenging to modify or optimize.

To this end, following (Fähse 2021) we distilled the off-the-shelf Understand AI model into a highly optimized YOLOv5 (Redmon et al.

[18] We found WebP with a quality setting of 85 to provide the best balance between decompression speed, PSNR, and space. A detailed analysis of the different trade-offs between major image compression formats (and their various implementations—`libjpeg` is not the same as `mozjpeg`) is beyond the scope of this chapter and could probably take up several chapters on its own. In practice, WebP is a good compromise between decompression speed, quality, and storage. In the time since we generated the dataset, a few new options, including JPEG-XL and AVIF have started gaining traction. We did not consider them at the time, but they would be worth looking into for future projects, especially since AVIF is likely to have good hardware support on modern computers. AVIF is the single-image version of the AV1 codec. Please refer to `https://aomedia.org/` for more information.

[19] While naively blurring sensitive elements interferes with image realism and may limit applicability in detection tasks, we found the simplicity of this approach appealing and the downsides not too serious given the dataset's primary goal of benchmarking global localization. Gaussian blur is used for anonymization in many commercial applications, including Google Street View and datasets such as A2D2 (Geyer et al. 2020).

[20] github.com/understand-ai/anonymizer

2016; Jocher 2020) architecture which we compiled into a TensorRT engine[21]. After several iterations we reached a model which ran 30× faster than the baseline, taking us from 7.4 to 250 FPS on an AWS instance equipped with an A10G (RTX 3090 equivalent) GPU.

[21] While torchscript compilation provided some modest benefits, converting our model to an fp16 TensorRT engine constituted the most significant performance uplift.

Implementation details. We used the `yolov5m` architecture variant trained with the `hyp.scratch-high.yaml` config for 70 epochs. We performed inference at 1280×800 and rescaled the boxes to the original image size of 1920×1200. In practice, we found this to be perfectly adequate, while also being computationally efficient. Any faces and plates missed by the lower resolution model typically span only a handful of pixels in size in the original images, making them illegible and thus harmless from a privacy point of view.

We likewise hand-implemented a GPU-accelerated Gaussian Blur operator in PyTorch, which allowed us to reduce the time spent applying a Gaussian blur to a full-resolution image from 107 ms (CPU, OpenCV) to 12.5 ms (GPU, naive implementation), and finally to 1.9 ms (GPU, separable filter + TorchScript). The time also includes feathering the edges of the blur mask in order to avoid jarring borders.

7.6.4 Dataset SDK

We also provide a Python SDK for interacting with the dataset. It includes a command-line tool for exploring the data, log readers that help deal with the log indexes, and a few simple PyTorch dataloaders that are compatible with streaming from S3.

Note that while the SDK provides many convenient features for dealing with Pit30M, it is not required since the files follow a simple and relatively flat structure which can be accessed with any S3 library or the standalone S3 command-line suite provided by AWS[22]. Neither option requires any sort of AWS account.

[22] `https://docs.aws.amazon.com/cli/`

Anyone can install the SDK using Python's `pip` package manager as `pip install pit30m`. This allows the Pit30M utilities to be used from user scripts, like training or visualization code, or from a command-line interface (CLI) as `python -m pit30m.cli`. Please refer to the project website[23] or the SDK repository [24] for specific code documentation and examples of use cases.

[23] pit30m.github.io

[24] github.com/pit30m/pit30m

7.7 Conclusion

In this chapter, we introduced Pit30M, a novel large-scale dataset for image and LiDAR localization, and studied retrieval-based methods in the context of self-driving cars. Our dataset provides extensive metadata and sub-meter ground truth and allows researchers to study accurate global localization at city scale. Furthermore, we saw that

this dataset can be applied to a wide range of other research tasks, such as vision foundation models, object discovery, and novel view synthesis. We provided an initial benchmark with multiple methods for visual and LiDAR localization, and in the process, showed that strong modern convolutional backbones perform remarkably well in this scenario. Our analysis also hints at future research directions using multi-sensor fusion and highlights challenging scenarios for localization. Our dataset and metadata are available on the Pit30M project website.

8
Conclusions and Future Directions

> When I learned the meaning of "I" and "me" and found that I was something, I began to think.
>
> *Hellen Keller*

Autonomous vehicles have the potential to change our lives for the better by reducing logistics costs, improving access to transportation, and making travel safer. However, in order to accomplish this, every component of an autonomous robot needs to operate robustly in a wide range of environments. In this thesis, we covered a line of work studying the robustness of robotic systems at the module and system levels through the lens of localization. To conclude the dissertation, we summarize the key contributions together with their strengths and limitations and discuss future work.

8.1 Summary

This thesis focused on improving robot localization systems by analyzing their limitations at both the module and system levels. Through this endeavor, we developed multiple insights that apply to a wide range of problems in robotics and machine learning.

In Chapter 3, we began by proposing a simple yet effective way to localize an SDV within a known map by matching semantic information perceived online to similar cues stored in a map. This approach was scalable and lightweight as it leveraged information already necessary for autonomous driving, such as the locations of lanes and traffic signs. Nevertheless, the method still suffered from drift, particularly longitudinally, due to the sparsity of observations.

We tackled this limitation in Chapter 4 by proposing a way to leverage dense map information by learning the localization matching function directly. By eschewing the need to extract human-curated

features such as lane lines and instead using all the information present in the sensor data while relying on a neural network to infer the most salient localization cues on its own, we ended up with a more robust solution. However, in spite of its improved accuracy, this approach sacrificed scalability by relying on dense map imagery, which can take up several terabytes in storage, driving up costs and slowing down operations.

We addressed the storage challenges in Chapter 5 by observing that the matching functions learned from data tended to focus on a specific set of sparse cues, which they identified as being reliable and effective for localization. This led to an enhancement brought about by introducing a novel binary compression module and adding sparsity and compressibility priors into the training framework of the LiDAR matcher. This allowed the network's emergent sparse perception to be translated into real storage space gains, enabling us to reduce map size by an order of magnitude compared to off-the-shelf lossy codecs like JPEG or WebP without sacrificing localization performance. The results from this chapter highlight the importance of task awareness when designing compression algorithms for data from a specialized domain, such as LiDAR map imagery.

In Chapter 6, we took a step back and analyzed localization performance in the context of an entire robotic system. Given their importance in associating sensor observations with prior map data, we analyzed the accuracy requirements of localization systems. We showed that map-based perception systems tend to be robust to translation errors in their pose but are much more sensitive to angular errors. As the range of robotic sensors evolves and improves, this issue will continue to become more salient as rotational misalignment increases as a function of distance. Following our analysis, we proposed a multi-task solution that can run LiDAR localization as a part of a perception neural network, giving it the ability to correct a wide range of localization errors with minimal computational overhead.

Finally, in Chapter 7, we shifted our focus from online to global localization, and studied the importance of large and diverse datasets through its lens. We introduced a new public dataset called *Pit30M* and used it to analyze global localization for self-driving cars. We observed that as the scale of a dataset increased, the specific architectural differences of global localization models that were state of the art at the time became less important. Recent studies have confirmed the potential of relatively simple architectures and large quantities of data in a wide range of tasks, including natural language processing (Brown et al. 2020) and visual representation learning (Radford et al. 2021).

While all chapters focused on applications in autonomous driving, many of the insights apply to other types of robots and indeed

to other areas of machine learning. For example, the idea of blending classic graphical models with deep representations has a wide range of applications, from planning (Y. Hu et al. 2023) and game playing (Schrittwieser et al. 2020) to protein folding (Jumper et al. 2021) and catalyst discovery (Tran et al. 2023). The idea of task-specific compression is likewise powerful given the vast amounts of data produced by most computer systems, whether they are robot drivers or sales platforms; most of this data will never be seen by a human, but rather, it will get processed by a specific set of algorithms. Why not optimize and compress the representations accordingly? Reducing information to the absolute minimum required to solve a task, bypassing a human's notions of "fidelity," also has numerous applications in privacy-focused and federated ML, where only key aspects of complex data points need to be shared with the learning process.

The system-wide analysis we performed in Chapter 6 highlighted the importance of going beyond traditional task-level benchmarks and studying the impact of a model on the overall system performance. In recent years, there has been a constant growth in the number of applications integrating machine learning models at some level, and this trend has only accelerated with the explosion in popularity LLMs such as ChatGPT and Llama2 have experienced. From autonomous robots (Mao et al. 2023) to customer support, large models are becoming critical components of more and more applications. Many times, a model which performs well on a set of standard benchmarks is integrated into a broader system where it ends up performing poorly, perhaps due to domain shift or due to certain assumptions present in the standard benchmarks no longer holding (J. Li et al. 2024). Chapter 6 presents a methodology for establishing a system-wide benchmark and analyzing system-level metrics as a function of a single module (SDV localization in our case), which can be applied to complex real-world production systems in order to validate their resilience.

While LiDAR was the primary sensor in most of the presented applications, the insights in this thesis can likewise be applied to camera and RADAR systems, e.g., by using BEV unprojection for monocular camera images (Philion and Fidler 2020; W. Yang et al. 2021) or by explicitly leveraging depth information inferred from stereo (Lipson, Teed, and Deng 2021). Likewise, BEV matching techniques also apply to RADAR data (Barnes, Weston, and Posner 2019; Weston et al. 2022).

8.2 Future Work

In spite of recent advances, the task of autonomous driving is still far from solved. Many open challenges remain at the intersection of areas

as diverse as computer vision, systems engineering, safety, and ethics.

We can distill the limitations discussed throughout this thesis into several avenues that are promising for future research. These are examples of areas that are worth exploring in order to advance along the path towards safe, effective, inclusive, and scalable robotic agents.

Multi-sensor, low-cost perception. While the majority of applications presented in this thesis have focused on LiDAR as the primary sensors, experiments such as those in Chapter 3 or Chapter 7 have demonstrated the potential of multi-sensor or camera-based methods. The use of multiple sensors brings about redundancies that can improve safety. Cleverly combining multiple lower-cost sensors (such as consumer-grade cameras and RADARs) can also lead to systems which perform on par with those leveraging much more expensive ones. One substantial barrier to adopting intelligent robots in our daily lives is the fact that many effective applications still require costly sensors. For example, most autonomous vehicles rely on LiDAR sensing, and while high-resolution LiDAR sensors have seen substantial cost reductions over the past decade, they are still not affordable enough to deploy everywhere[1].

Many exciting new developments, such as event cameras (Gallego et al. 2020; Low and Lee 2023) and compact solid-state LiDARs (Velodyne Lidar, Inc. 2021), have the potential to increase the reach of automated systems by reducing costs. Several of these techniques can also improve system performance in general. For example, event cameras, also known as dynamic vision sensors, can help improve reaction times due to their extremely low latencies, leading to improved safety by allowing robots to react to hazards more quickly than other sensors, such as LiDAR, would allow (Gallego et al. 2020; Gehrig and Scaramuzza 2024). However, many of these sensors are still far from seeing wide adoption[2], and most research in autonomous driving is still overwhelmingly focused on spinning LiDARs and cameras (Sun et al. 2020; Y. Hu et al. 2023; Caesar et al. 2020; Mao et al. 2021). Nevertheless, as robotics researchers, we need to continue striving to broaden our horizons and explore the potential of less-beaten sensing paths: the rewards are bound to be worth it!

[1] While several autonomous driving companies, such as Waymo (Verghese 2019) and Aurora (Aurora 2021), have gone the way of developing and manufacturing LiDAR sensors in-house, this approach is only applicable to large, well-established companies due to the significant required amounts of upfront capital investment.

[2] As of March 2024, there have been no further announcements regarding the Velabit sensor, and while there are no official announcements, the project appears to have been canceled (example discussion).

8.2.1 Simulation

Two major limitations still constrain the development of robotic systems, even when using the largest and best-curated datasets. First, any dataset collected in the real world will be unable to capture the full distribution of possible appearances and behaviors a driver could encounter in the real world. Second, as robotic systems need to interact with the world, the use of a static dataset means such a system

would never receive *feedback* from the environment on its actions, beyond what static labels provide. In other words, the robot would lack *embodiment*: development on static datasets can only proceed in open loop. This limitation can be alleviated by real-world testing, but that brings its own host of challenges, such as increased costs, slow iteration speed, and safety concerns[3]. Furthermore, recent research has shown that metrics computed in open loop are not predictive of closed-loop (i.e., real-world) performance (Codevilla et al. 2018; Dauner et al. 2023). Therefore, we need a way to synthesize and potentially search for challenging events, interactions, and object appearances.

[3] Even in the presence of a human safety driver, testing early software on a real SDV in scenarios involving other vehicles poses some collision risk. Furthermore, physical track testing cannot involve scenarios where a test failure is known to cause a collision. This excludes many critical corner cases, such as aggressive cut-ins or lead-actor braking.

At the same time, the robotic system being trained and evaluated (e.g., the SDV software stack) must be able to interact with its environment, affecting its and other agents' states in a continual closed-loop feedback cycle.

Simulation can accomplish both of these goals. First, a simulator can be used for synthetic data generation tuned to cover a broader range of challenging object appearances, backgrounds, and scenarios than even the best-curated real dataset. Second, we can use the simulator for closed-loop evaluation and training, which bypasses the embodiment problem.

As an example, consider the analysis from Chapter 6.3.1, which relied on open-loop evaluation. The open-loop nature of the evaluation limited the time horizon of our analysis, as unrolling the planner and the environment for too long can cause very unrealistic results due to the simulated traffic's inability to react to the SDV. By using a high-fidelity closed-loop simulator, we could increase the realism of the analysis, e.g., by evaluating much longer time horizons in order to uncover more subtle issues brought about by localization errors.

While simulation brings its own set of challenges, like the need to reproduce real-world environments in high fidelity, render realistic sensor observations, simulate human-like reactive actors, and generally quantify and minimize the sim-to-real domain gap, recent work (Mildenhall et al. 2021; Zyrianov, Zhu, and Wang 2022; Kerbl et al. 2023; Gulino et al. 2024) has shown tremendous progress in all these areas. Further research has explored additional ways of improving the realism (Manivasagam et al. 2023), flexibility (Murthy et al. 2020), and especially the scalability (Petrenko et al. 2021; Shacklett et al. 2023; Gulino et al. 2024) of simulators. By enriching our simulators, we can dramatically improve the speed of iteration in robotics without sacrificing safety.

Mixed and augmented reality simulation. At the same time, there is still a system domain gap between running in the real world and even the highest-end simulator. For example, most simulation

benchmarks and analysis papers highlight photo-realism and open-loop perception fidelity (Manivasagam et al. 2023) but fail to discuss the system design challenges, the importance of simulating system dynamics, as well as the vast complexities associated with running every component in real-time with low latency. Designing a feedforward network that runs in 100 ms is easy. Designing a system that can handle sensor inputs, preprocess them, route them to multiple GPUs, validate the outputs, then actuate a series of effectors at 10Hz or faster, reliably, over tens of thousands of hours of operation is much more challenging. While previous work has demonstrated the potential for pure sim-to-real generalization on specific tasks such as robot locomotion (Hwangbo et al. 2019) by leveraging aggressive domain randomization, safely generalizing system performance purely from simulated data remains an open research question.

A promising avenue for approaching this set of challenges is by bridging the gap between simulation and reality through vehicle-in-the-loop or mixed-reality simulation (Zofka et al. 2018, 2023; Shen et al. 2024). While vehicle-in-the-loop simulation typically refers to replaying driving log data onboard an actual vehicle in order to reproduce its computational setup perfectly, mixed reality simulation takes this one step further, running simulation onboard a functioning robot as it operates on a closed course, blending the simulated observations with the real sensor data. This can bring about the diversity, safety, reproducibility, and, to a partial extent, scalability benefits of pure simulation, combining them with the advantages of real-world testing—real sensors, real onboard computers, real noise, and real actuation.

Large-scale simulation. Much like scalability has benefitted models in NLP and generative modeling (e.g., training on 1000s of GPUs or TPUs), so too can it benefit robotic models that rely on closed-loop simulation. However, the need for closed-loop evaluation and training brings a unique set of challenges to traditional data-parallel scaling paradigms. It is no longer enough to train a neural network at scale—instead, it becomes necessary also to run large-scale simulations in which a robot system can operate and receive feedback. This poses exciting software engineering challenges, as tasks like physics simulation, traffic simulation, and sensor data rendering also need to be performed in batch, on large-scale computer clusters. While early work achieved promising results simulating lightweight environments such as Atari games[4], autonomous driving simulation has much higher requirements in terms of sensor rendering (photo realistic camera rendering, LiDAR rendering, etc.), environment scale (e.g., entire cities), and actor behaviors. Recently released simulators (Petrenko et al. 2021; Shacklett et al. 2023; Gulino et al. 2024) focus on batched

[4] The famous Atari 2600 had a 1.2 MHz processor, meaning that even low-end modern computers can simulate Atari games tens of thousands of times faster than real-time, thanks to increased clock speeds, processor counts, and instruction-level parallelism (Bellemare et al. 2013; Petrenko et al. 2021).

processing for tasks like graphics rendering (Petrenko et al. 2021) or actor behavior simulations (Gulino et al. 2024), making full use of GPU acceleration to achieve large throughput. For example, Megaverse (Petrenko et al. 2021) can simulate complex 3D environments at over 100k FPS on a single system equipped with a consumer-grade Nvidia RTX 3090 GPU, albeit with relatively simple graphics. Advancing this line of research and designing robotic systems that can learn in batched environments and then transfer the knowledge to the real world effectively is critical for the development of practical autonomous robots.

8.3 Outlook

The field of autonomous driving is full of challenging problems, and we still have many years ahead of us before we can call them solved. Nevertheless, the payoff for solving them will be nothing short of a technological revolution. In addition to their applications in mobility, advances in AI capable of unlocking safe autonomous driving at scale can transfer to many other fields, such as automated agriculture, search and rescue, space exploration, autonomous manufacturing, and surgical robotics. This thesis only scratched the surface of a fraction of the problems that we need to solve to reach our goals. I am nevertheless profoundly grateful for having had the chance to build my skills as a scientist by working in this area, and I am excited to see what the future of robotics holds!

Bibliography

Abadi, Martín, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. "TensorFlow: A System for Large-Scale Machine Learning." In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 265–83.

Agarwal, Sameer, Noah Snavely, Ian Simon, Steven M Seitz, and Richard Szeliski. 2009. "Building Rome in a day." In *ICCV*.

Aghili, Farhad, and Chun Yi Su. 2016. "Robust relative navigation by integration of ICP and adaptive Kalman filter using laser scanner and IMU." *TMECH*. https://doi.org/10.1109/TMECH.2016.2547905.

Agro, Ben, Quinlan Sykora, Sergio Casas, Thomas Gilles, and Raquel Urtasun. 2024. "UnO: Unsupervised Occupancy Fields for Perception and Forecasting." In *CVPR*, 14487–96.

Agro, Ben, Quinlan Sykora, Sergio Casas, and Raquel Urtasun. 2023. "Implicit Occupancy Flow Fields for Perception and Prediction in Self-Driving." In *CVPR*, 1379–88.

Alet, Ferran, Tomás Lozano-Pérez, and Leslie P Kaelbling. 2018. "Modular Meta-Learning." In *CoRL*.

Amini, Alexander, Guy Rosman, Sertac Karaman, and Daniela Rus. 2019. "Variational End-to-End Navigation and Localization." In *ICRA*.

Antonante, Pasquale, Sushant Veer, Karen Leung, Xinshuo Weng, Luca Carlone, and Marco Pavone. 2023. "Task-Aware Risk Estimation of Perception Failures for Autonomous Vehicles."

Arandjelovic, Relja, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. 2016. "NetVLAD: CNN Architecture for Weakly Supervised Place Recognition." In *CVPR*, 5297–307.

Artuñedo, Antonio, Jorge Villagra, Jorge Godoy, and Maria Dolores Del Castillo. 2020. "Motion Planning Approach Considering Localization Uncertainty." *IEEE Transactions on Vehicular Technology* 69 (6): 5983–94.

Aurora. 2021. "The Power of FMCW Lidar + Scale: Why Acquiring OURS Lidar Unlocks the Commercialization of the Aurora Driver."

2021. https://blog.aurora.tech/progress/the-power-of-fmcw-lidar-and-scale-acquiring.

Bai, Min, G. Mattyus, N. Homayounfar, S. Wang, SK. Lakshmikanth, and Raquel Urtasun. 2018. "Deep Multi-Sensor Lane Detection." In *IROS*.

Bai, Min, and Raquel Urtasun. 2017. "Deep Watershed Transform for Instance Segmentation." In *CVPR*.

Bailey, Tim, and Hugh Durrant-Whyte. 2006. "Simultaneous Localization and Mapping (SLAM)." *Update* 13 (September): 108–17. https://doi.org/10.1109/MRA.2006.1678144.

Balle, Johannes, Valero Laparra, and Eero P. Simoncelli. 2016. "End-to-End Optimization of Nonlinear Transform Codes for Perceptual Quality." *2016 Picture Coding Symposium (PCS)*. https://doi.org/10.1109/pcs.2016.7906310.

Balntas, Vassileios, Karel Lenc, Andrea Vedaldi, and Krystian Mikolajczyk. 2017. "HPatches: A Benchmark and Evaluation of Handcrafted and Learned Local Descriptors." In *CVPR*, 5173–82.

Bansal, Aayush, Hernán Badino, and Daniel Huber. 2014. "Understanding How Camera Configuration and Environmental Conditions Affect Appearance-Based Localization." In *IV*, 800–807. IEEE.

Bansal, Mayank, and Kostas Daniilidis. 2014. "Geometric Urban Geo-Localization." In *CVPR*.

Bansal, Mayank, Alex Krizhevsky, and Abhijit Ogale. 2019. "ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst." In *RSS*.

Barfoot, Timothy D. 2024. *State Estimation for Robotics*. Cambridge University Press.

Barnes, Dan, Rob Weston, and Ingmar Posner. 2019. "Masking by Moving: Learning Distraction-Free Radar Odometry from Pose Information." In *CoRL*.

Bârsan, Ioan Andrei, Peidong Liu, Marc Pollefeys, and Andreas Geiger. 2018. "Robust Dense Mapping for Large-Scale Dynamic Environments." In *ICRA*, 7510–17. IEEE.

Bay, Herbert, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. 2008. "Speeded-up Robust Features (SURF)." *Computer Vision and Image Understanding* 110 (3): 346–59.

Bellemare, Marc G, Yavar Naddaf, Joel Veness, and Michael Bowling. 2013. "The Arcade Learning Environment: An Evaluation Platform for General Agents." *Journal of Artificial Intelligence Research* 47: 253–79.

Bengio, Yoshua, Nicholas Léonard, and Aaron Courville. 2013. "Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation." *arXiv Preprint arXiv:1308.3432*.

Berner, Christopher, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, et al. 2019. "Dota 2 with Large Scale Deep Reinforcement Learning." *arXiv Preprint arXiv:1912.06680*.

Bernreiter, Lukas, Lionel Ott, Juan Nieto, Roland Siegwart, and Cesar Cadena. 2021. "PHASER: A Robust and Correspondence-Free Global Pointcloud Registration." *IEEE Robotics and Automation Letters* 6 (2): 855–62.

Berrou, Claude, Alain Glavieux, and Punya Thitimajshima. 1993. "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes. 1." In *Proceedings of ICC'93-IEEE International Conference on Communications*, 2:1064–70. IEEE.

Biber, Peter, and Wolfgang Straßer. 2003. "The Normal Distributions Transform: A New Approach to Laser Scan Matching." In *IROS*.

Bojarski, Mariusz, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, et al. 2016. "End to End Learning for Self-Driving Cars." *arXiv Preprint arXiv:1604.07316*.

Brachmann, Eric, Tommaso Cavallari, and Victor Adrian Prisacariu. 2023. "Accelerated Coordinate Encoding: Learning to Relocalize in Minutes Using RGB and Poses." In *CVPR*, 5044–53.

Brachmann, Eric, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. 2017. "DSAC - Differentiable RANSAC for Camera Localization." In *CVPR*.

Brachmann, Eric, Frank Michel, Alexander Krull, Michael Ying Yang, Stefan Gumhold, et al. 2016. "Uncertainty-Driven 6d Pose Estimation of Objects and Scenes from a Single RGB Image." In *CVPR*.

Brachmann, Eric, and Carsten Rother. 2018. "Learning Less Is More-6d Camera Localization via 3d Surface Regression." In *CVPR*.

Brown, Tom, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, et al. 2020. "Language Models Are Few-Shot Learners." In *NeurIPS*, 1877–1901. PMLR.

Brubaker, Marcus A, Andreas Geiger, and Raquel Urtasun. 2013. "Lost! Leveraging the Crowd for Probabilistic Visual Self-Localization." In *CVPR*, 3057–64.

Buciluǎ, Cristian, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. "Model Compression." In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Bujanca, Mihai, Paul Gafton, Sajad Saeedi, Andy Nisbet, Bruno Bodin, Michael FP O'Boyle, Andrew J Davison, et al. 2019. "SLAMBench 3.0: Systematic Automated Reproducible Evaluation of SLAM Systems for Robot Vision Challenges and Scene

Understanding." In *ICRA*.

Burnett, Keenan, David J Yoon, Yuchen Wu, Andrew Z Li, Haowei Zhang, Shichen Lu, Jingxing Qian, et al. 2023. "Boreas: A Multi-Season Autonomous Driving Dataset." *IJRR* 42 (1-2): 33–42.

Caesar, Holger, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. 2020. "nuScenes: A Multimodal Dataset for Autonomous Driving." In *CVPR*.

Cai, Xudong, Yongcai Wang, Zhe Huang, Yu Shao, and Deying Li. 2024. "VOLoc: Visual Place Recognition by Querying Compressed Lidar Map." In *ICRA*.

Camposeco, Federico, Andrea Cohen, Marc Pollefeys, and Torsten Sattler. 2019. "Hybrid Scene Compression for Visual Localization." In *CVPR*.

Cao, Bingyi, Andre Araujo, and Jack Sim. 2020. "Unifying Deep Local and Global Features for Efficient Image Search." In *ECCV*.

Carlevaris-Bianco, Nicholas, Arash K Ushani, and Ryan M Eustice. 2016. "University of Michigan North Campus long-term vision and lidar dataset." *IJRR*.

Carmona, Juan, Carlos Guindel, Fernando Garcia, and Arturo de la Escalera. 2021. "eHMI: Review and Guidelines for Deployment on Autonomous Vehicles." *Sensors* 21 (9): 2912.

Casas, Sergio, Cole Gulino, Simon Suo, Katie Luo, Renjie Liao, and Raquel Urtasun. 2020. "Implicit Latent Variable Model for Scene-Consistent Motion Forecasting." In *ECCV*, 624–41. Springer.

Casas, Sergio, Cole Gulino, Simon Suo, and Raquel Urtasun. 2020. "The Importance of Prior Knowledge in Precise Multimodal Prediction." In *IROS*.

Casas, Sergio, Wenjie Luo, and Raquel Urtasun. 2018. "IntentNet: Learning to Predict Intention from Raw Sensor Data." In *CoRL*, 947–56. PMLR.

Casas, Sergio, Abbas Sadat, and Raquel Urtasun. 2021. "MP3: A Unified Model to Map, Perceive, Predict and Plan." In *CVPR*, 14403–12.

Chai, Yuning, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. 2020. "MultiPath: Multiple Probabilistic Anchor Trajectory Hypotheses for Behavior Prediction." In *CoRL*.

Chang, Chia-Ming, Koki Toda, Xinyue Gui, Stela H Seo, and Takeo Igarashi. 2022. "Can Eyes on a Car Reduce Traffic Accidents?" In *Proceedings of the 14th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 349–59.

Chaurasia, Abhishek, and Eugenio Culurciello. 2017. "LinkNet: Exploiting Encoder Representations for Efficient Semantic Segmentation." In *2017 IEEE Visual Communications and Image Processing*

(VCIP), 1–4. IEEE.

Checchin, Paul, Franck Gérossier, Christophe Blanc, Roland Chapuis, and Laurent Trassoudaine. 2010. "Radar Scan Matching SLAM Using the Fourier-Mellin Transform." In *Field and Service Robotics*, 151–61. Springer.

Chen, David M, Georges Baatz, B Girod, R Grzeszczuk, K Koser, SS Tsai, R Vedantham, et al. 2011. "City-Scale Landmark Identification on Mobile Devices." In *CVPR*.

Chen, Xiaozhi, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. 2017. "Multi-View 3D Object Detection Network for Autonomous Driving." In *CVPR*.

Chen, Xieyuanli, Thomas Läbe, Andres Milioto, Timo Röhling, Jens Behley, and Cyrill Stachniss. 2022. "OverlapNet: A Siamese Network for Computing LiDAR Scan Similarity with Applications to Loop Closing and Localization." *Autonomous Robots*, 1–21.

Chen, Zhao, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. "GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks." In *ICML*.

Chen, Zhuoyuan, Xun Sun, Liang Wang, Yinan Yu, and Chang Huang. 2015. "A Deep Visual Correspondence Embedding Model for Stereo Matching Costs." In *ICCV*.

Choy, Christopher, Wei Dong, and Vladlen Koltun. 2020. "Deep Global Registration." In *CVPR*.

Cieslewski, Titus, Siddharth Choudhary, and Davide Scaramuzza. 2018. "Data-Efficient Decentralized Visual SLAM." In *ICRA*.

Clark, Kevin, Minh-Thang Luong, Urvashi Khandelwal, Christopher D Manning, and Quoc V Le. 2019. "BAM! Born-Again Multi-Task Networks for Natural Language Understanding." In *ACL*.

Codevilla, Felipe, Antonio M Lopez, Vladlen Koltun, and Alexey Dosovitskiy. 2018. "On Offline Evaluation of Vision-Based Driving Models." In *ECCV*, 236–51.

Collobert, Ronan, and Jason Weston. 2008. "A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning." In *ICML*.

Crawshaw, Michael. 2020. "Multi-Task Learning with Deep Neural Networks: A Survey." *arXiv Preprint arXiv:2009.09796*.

Cui, Henggang, Vladan Radosavljevic, Fang-Chieh Chou, Tsung-Han Lin, Thi Nguyen, Tzu-Kuo Huang, Jeff Schneider, and Nemanja Djuric. 2019. "Multimodal Trajectory Predictions for Autonomous Driving Using Deep Convolutional Networks." In *ICRA*.

Cummins, Mark, and Paul Newman. 2008. "FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance." *IJRR*.

Dai, Jifeng, Kaiming He, and Jian Sun. 2016. "Instance-Aware Semantic Segmentation via Multi-Task Network Cascades." In *CVPR*.

Daimler AG. 2015. "The Mercedes-Benz F 015 Luxury in Motion." 2015. https://media.mbusa.com/releases/the-mercedes-benz-f-015-luxury-in-motion.

Dasgupta, B, and BN Chatterji. 1996. "Fourier-Mellin Transform Based Image Matching Algorithm." *IETE Journal of Research* 42 (1): 3–9.

Dauner, Daniel, Marcel Hallgarten, Andreas Geiger, and Kashyap Chitta. 2023. "Parting with Misconceptions about Learning-Based Vehicle Motion Planning." In *CoRL*, 1268–81. PMLR.

Davison, Andrew J. 2018. "FutureMapping: The Computational Structure of Spatial AI Systems." *arXiv Preprint arXiv:1803.11288*.

Davison, Andrew J, Ian D Reid, Nicholas D Molton, and Olivier Stasse. 2007. "MonoSLAM: Real-Time Single Camera SLAM." *TPAMI* 29 (6): 1052–67.

Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. "ImageNet: A Large-Scale Hierarchical Image Database." In *CVPR*, 248–55. IEEE.

Deschênes, Simon-Pierre, Dominic Baril, Vladimír Kubelka, Philippe Giguere, and François Pomerleau. 2021. "Lidar Scan Registration Robust to Extreme Motions." In *2021 18th Conference on Robots and Vision (CRV)*, 17–24. IEEE.

DeTone, Daniel, Tomasz Malisiewicz, and Andrew Rabinovich. 2018. "SuperPoint: Self-Supervised Interest Point Detection and Description." In *CVPR Workshops*, 224–36.

Dewan, Ayush, Tim Caselitz, and Wolfram Burgard. 2018. "Learning a Local Feature Descriptor for 3D LiDAR Scans." In *IROS*.

Douze, Matthijs, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. "The FAISS Library." https://arxiv.org/abs/2401.08281.

Du, Juan, Rui Wang, and Daniel Cremers. 2020. "DH3D: Deep Hierarchical 3D Descriptors for Robust Large-Scale 6DoF Relocalization." In *ECCV*.

Dusmanu, Mihai, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. 2019. "D2-Net: A Trainable CNN for Joint Detection and Description of Local Features." In *CVPR*.

Ebadi, Kamak, Lukas Bernreiter, Harel Biggie, Gavin Catt, Yun Chang, Arghya Chatterjee, Christopher E Denniston, et al. 2023. "Present and Future of SLAM in Extreme Environments: The DARPA SubT Challenge." *T-RO*.

Engel, Jakob, Vladlen Koltun, and Daniel Cremers. 2018. "Direct Sparse Odometry." *TPAMI*. https://cvg.cit.tum.de/_media/spezial/bib/engel_et_al_pami2018.pdf.

Engel, Jakob, Thomas Schöps, and Daniel Cremers. 2014. "LSD-SLAM: Large-Scale Direct Monocular SLAM." In *ECCV*.

Fähse, Thomas. 2021. "Making Dashcam Videos GDPR Compliant Using Machine Learning." 2021. https://towardsdatascience.com/making-dashcam-videos-gdpr-compliant-f9832883fe94.

Fan, Lue, Yuxue Yang, Yiming Mao, Feng Wang, Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. 2023. "Once Detected, Never Lost: Surpassing Human Performance in Offline LiDAR Based 3D Object Detection." In *ICCV*, 19820–29.

Filliat, David. 2007. "A Visual Bag of Words Method for Interactive Qualitative Localization and Mapping." In *ICRA*.

Floros, Georgios, Benito van der Zander, and Bastian Leibe. 2013. "OpenStreetSLAM: Global Vehicle Localization Using OpenStreetMaps." In *ICRA*.

Frossard, Davi, Simon Suo, Sergio Casas, James Tu, Rui Hu, and Raquel Urtasun. 2020. "StrObe: Streaming Object Detection from LiDAR Packets." In *CoRL*.

Frossard, Davi, and Raquel Urtasun. 2018. "End-to-End Learning of Multi-Sensor 3D Tracking by Detection." In *ICRA*, 635–42. IEEE.

Gallego, Guillermo, Tobi Delbrück, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, et al. 2020. "Event-Based Vision: A Survey." *TPAMI* 44 (1): 154–80.

Gallier, Jean H, and Jocelyn Quaintance. 2020. *Differential Geometry and Lie Groups: A Computational Perspective*. Vol. 12. Springer Nature.

Gálvez-López, Dorian, and Juan D Tardós. 2012. "Bags of Binary Words for Fast Place Recognition in Image Sequences." *T-RO*.

Gao, Jiyang, Chen Sun, Hang Zhao, Yi Shen, Dragomir Anguelov, Congcong Li, and Cordelia Schmid. 2020. "VectorNet: Encoding HD Maps and Agent Dynamics from Vectorized Representation." In *CVPR*.

Gehrig, Daniel, and Davide Scaramuzza. 2024. "Low-Latency Automotive Vision with Event Cameras." *Nature* 629 (8014): 1034–40.

Geiger, Andreas, Philip Lenz, Christoph Stiller, and Raquel Urtasun. 2013. "Vision Meets Robotics: The KITTI Dataset." *IJRR* 32 (11): 1231–37.

Geiger, Andreas, Julius Ziegler, and Christoph Stiller. 2011. "StereoScan: Dense 3D Reconstruction in Real-Time." In *IV*.

Geyer, Jakob, Yohannes Kassahun, Mentar Mahmudi, Xavier Ricou, Rupesh Durgesh, Andrew S Chung, Lorenz Hauswald, et al. 2020. "A2D2: Audi Autonomous Driving Dataset." *arXiv Preprint arXiv:2004.06320*.

Gordo, Albert, Jon Almazán, Jerome Revaud, and Diane Larlus. 2016. "Deep Image Retrieval: Learning Global Representations for Image

Search." In *ECCV*.

Gulino, Cole, Justin Fu, Wenjie Luo, George Tucker, Eli Bronstein, Yiren Lu, Jean Harb, et al. 2024. "Waymax: An Accelerated, Data-Driven Simulator for Large-Scale Autonomous Driving Research." *NeurIPS* 36.

Guo, Michelle, Albert Haque, De-An Huang, Serena Yeung, and Li Fei-Fei. 2018. "Dynamic Task Prioritization for Multitask Learning." In *ECCV*.

Guo, Xiaoxin, Zhiwen Xu, Yinan Lu, and Yunjie Pang. 2005. "An Application of Fourier-Mellin Transform in Image Registration." In *The Fifth International Conference on Computer and Information Technology (CIT'05)*, 619–23. IEEE.

Han, Wei, Zhengdong Zhang, Benjamin Caine, Brandon Yang, Christoph Sprunk, Ouais Alsharif, Jiquan Ngiam, Vijay Vasudevan, Jonathon Shlens, and Zhifeng Chen. 2020. "Streaming Object Detection for 3-D Point Clouds." In *ECCV*.

Hartley, Richard, and Andrew Zisserman. 2003. *Multiple View Geometry in Computer Vision*. Cambridge university press.

Hashimoto, Kazuma, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. "A Joint Many-Task Model: Growing a Neural Network for Multiple NLP Tasks." In *EMNLP*.

Hausler, Stephen, Sourav Garg, Ming Xu, Michael Milford, and Tobias Fischer. 2021. "Patch-NetVLAD: Multi-Scale Fusion of Locally-Global Descriptors for Place Recognition." In *CVPR*, 14141–52.

Havlena, Michal, and Konrad Schindler. 2014. "Vocmatch: Efficient Multiview Correspondence for Structure from Motion." In *ECCV*.

Hays, James, and Alexei A. Efros. 2008. "Im2gps: Estimating Geographic Information from a Single Image." In *CVPR*.

He, Kaiming, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. "Mask R-CNN." In *ICCV*.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification." In *ICCV*.

———. 2016. "Deep Residual Learning for Image Recognition." In *CVPR*, 770–78.

He, Li, Xiaolong Wang, and Hong Zhang. 2016. "M2DP: A novel 3D point cloud descriptor and its application in loop closure detection." In *IROS*.

Herau, Quentin, Nathan Piasco, Moussab Bennehar, Luis Roldao, Dzmitry Tsishkou, Cyrille Migniot, Pascal Vasseur, and Cédric Demonceaux. 2024. "SOAC: Spatio-Temporal Overlap-Aware Multi-Sensor Calibration Using Neural Radiance Fields." In *CVPR*, 15131–40.

Hoffer, Christian. 2017. "Ships Fooled in GPS Spoofing Attack Sug-

gest Russian Cyberweapon." 2017. https://www.newscientist.com/article/2143499-ships-fooled-in-gps-spoofing-attack-suggest-russian-cyberweapon/.

Hofmann-Wellenhof, Bernhard, Herbert Lichtenegger, and Elmar Wasle. 2007. *GNSS–Global Navigation Satellite Systems: GPS, GLONASS, Galileo, and More*. Springer Science & Business Media.

Hu, Anthony, Zak Murez, Nikhil Mohan, Sofia Dudas, Jeffrey Hawke, Vijay Badrinarayanan, Roberto Cipolla, and Alex Kendall. 2021. "FIERY: Future Instance Prediction in Bird's-Eye View from Surround Monocular Cameras." In *ICCV*, 15273–82.

Hu, Anthony, Lloyd Russell, Hudson Yeo, Zak Murez, George Fedoseev, Alex Kendall, Jamie Shotton, and Gianluca Corrado. 2023. "GAIA-1: A Generative World Model for Autonomous Driving." *arXiv Preprint arXiv:2309.17080*.

Hu, Yihan, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, et al. 2023. "Planning-Oriented Autonomous Driving." In *CVPR*, 17853–62.

Hwangbo, Jemin, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. 2019. "Learning Agile and Dynamic Motor Skills for Legged Robots." *Science Robotics* 4 (26).

Indelman, Vadim, Luca Carlone, and Frank Dellaert. 2015. "Planning in the Continuous Domain: A Generalized Belief Space Approach for Autonomous Navigation in Unknown Environments." *IJRR* 34 (7): 849–82.

Irschara, Arnold, Christopher Zach, Jan-Michael Frahm, and Horst Bischof. 2009. "From Structure-from-Motion Point Clouds to Fast Location Recognition." In *CVPR*.

Jaderberg, Max, Karen Simonyan, Andrew Zisserman, et al. 2015. "Spatial Transformer Networks." In *NIPS*.

Jegou, Herve, Matthijs Douze, and Cordelia Schmid. 2010. "Product Quantization for Nearest Neighbor Search." *TPAMI* 33 (1): 117–28.

Jégou, Hervé, Florent Perronnin, Matthijs Douze, Jorge Sánchez, Patrick Pérez, and Cordelia Schmid. 2011. "Aggregating Local Image Descriptors into Compact Codes." *TPAMI* 34 (9): 1704–16.

Jocher, Glenn. 2020. *YOLOv5 by Ultralytics* (version 7.0). https://doi.org/10.5281/zenodo.3908559.

Johnson, Jeff, Matthijs Douze, and Hervé Jégou. 2017. "Billion-Scale Similarity Search with GPUs." *arXiv Preprint arXiv:1702.08734*.

Joubert, Niels, Tyler GR Reid, and Fergus Noble. 2020. "Developments in Modern GNSS and Its Impact on Autonomous Vehicle Architectures." In *IV*.

Jumper, John, Richard Evans, Alexander Pritzel, Tim Green, Michael

Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, et al. 2021. "Highly Accurate Protein Structure Prediction with AlphaFold." *Nature* 596 (7873): 583–89.

Kaiser, Łukasz, Aidan N Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. 2017. "One Model to Learn Them All." *arXiv Preprint arXiv:1706.05137*.

Kanai, Takayuki, Igor Vasiljevic, Vitor Guizilini, Adrien Gaidon, and Rares Ambrus. 2023. "Robust Self-Supervised Extrinsic Self-Calibration." In *IROS*, 1932–39. IEEE.

Kashani, Alireza G, Michael J Olsen, Christopher E Parrish, and Nicholas Wilson. 2015. "A Review of LiDAR Radiometric Processing: From Ad Hoc Intensity Correction to Rigorous Radiometric Calibration." *Sensors* 15 (11): 28099–128.

Kendall, Alex, and Roberto Cipolla. 2016. "Modelling Uncertainty in Deep Learning for Camera Relocalization." In *ICRA*.

Kendall, Alex, Yarin Gal, and Roberto Cipolla. 2018. "Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics." In *CVPR*.

Kendall, Alex, Matthew Grimes, and Roberto Cipolla. 2015. "PoseNet: A Convolutional Network for Real-Time 6-DoF Camera Relocalization." In *ICCV*.

Kendall, Alex, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. 2019. "Learning to Drive in a Day." In *ICRA*.

Kendall, Alex, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. 2017. "End-to-End Learning of Geometry and Context for Deep Stereo Regression." In *ICCV*, 66–75.

Kerbl, Bernhard, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. "3D Gaussian Splatting for Real-Time Radiance Field Rendering." *ACM Transactions on Graphics* 42 (4): 1–14.

Kettwich, Carmen, Andreas Schrank, and Michael Oehl. 2021. "Teleoperation of Highly Automated Vehicles in Public Transport: User-Centered Design of a Human-Machine Interface for Remote-Operation and Its Expert Usability Evaluation." *Multimodal Technologies and Interaction* 5 (5): 26.

Kingma, Diederik P, and Jimmy Ba. 2015. "Adam: A Method for Stochastic Optimization." *ICLR*.

Klein, Georg, and David Murray. 2007. "Parallel Tracking and Mapping for Small AR Workspaces." In *ISMAR*, 1–10. http://ieeexplore.ieee.org/document/4538852/.

Klokov, Roman, and Victor Lempitsky. 2017. "Escape from Cells: Deep Kd-Networks for the Recognition of 3D Point Cloud Models." In *ICCV*.

Knopp, Jan, Josef Sivic, and Tomas Pajdla. 2010. "Avoiding Confusing Features in Place Recognition." In *ECCV*.

Komorowski, Jacek. 2021. "Minkloc3D: Point Cloud Based Large-Scale Place Recognition." In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 1790–99.

Kuglin, Charles D. 1975. "The Phase Correlation Image Alignment Method." In *IEEE Int. Conf. On Cybernetics and Society, 1975*, 163–65.

Kümmerle, Rainer, Bastian Steder, Christian Dornhege, Michael Ruhnke, Giorgio Grisetti, Cyrill Stachniss, and Alexander Kleiner. 2009. "On Measuring the Accuracy of SLAM Algorithms." *Autonomous Robots* 27 (4): 387.

Levinson, Jesse, Michael Montemerlo, and Sebastian Thrun. 2007. "Map-Based Precision Vehicle Localization in Urban Environments." *RSS*. http://www.roboticsproceedings.org/rss03/p16.pdf.

Levinson, Jesse, and Sebastian Thrun. 2010. "Robust Vehicle Localization in Urban Environments Using Probabilistic Maps." In *ICRA*, 4372–78. IEEE.

Lewis, Patrick, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, et al. 2020. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks." *NeurIPS* 33: 9459–74.

Li, Junyi, Jie Chen, Ruiyang Ren, Xiaoxue Cheng, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2024. "The Dawn After the Dark: An Empirical Study on Factuality Hallucination in Large Language Models." *arXiv Preprint arXiv:2401.03205*.

Li, Mengtian, Yu-Xiong Wang, and Deva Ramanan. 2020. "Towards Streaming Perception." In *ECCV*.

Li, Run. 2023. "Complete PPK Workflow for DJI Enterprise Drones." 2023. https://web.archive.org/web/20240619231528/https://enterprise-insights.dji.com/blog/ppk-post-processed-kinematics-workflow.

Li, Yunpeng, Noah Snavely, Dan Huttenlocher, and Pascal Fua. 2012. "Worldwide Pose Estimation Using 3d Point Clouds." In *ECCV*.

Li, Zhiqi, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Yu Qiao, and Jifeng Dai. 2022. "BEVFormer: Learning Bird's-Eye-View Representation from Multi-Camera Images via Spatiotemporal Transformers." In *ECCV*, 1–18. Springer.

Liang, Ming, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. 2019. "Multi-Task Multi-Sensor Fusion for 3D Object Detection." In *CVPR*.

Liang, Ming, Bin Yang, Wenyuan Zeng, Yun Chen, Rui Hu, Sergio Casas, and Raquel Urtasun. 2020. "PnPNet: End-to-End Perception and Prediction with Tracking in the Loop." In *CVPR*.

Lin, Chen-Hsuan, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. 2021. "BARF: Bundle-Adjusting Neural Radiance Fields." In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5741–51.

Lindenberger, Philipp, Paul-Edouard Sarlin, and Marc Pollefeys. 2023. "LightGlue: Local Feature Matching at Light Speed." In *ICCV*.

Linegar, Chris, Winston Churchill, and Paul Newman. 2015. "Work Smart, Not Hard: Recalling Relevant Experiences for Vast-Scale but Time-Constrained Localisation." In *ICRA*.

Lipson, Lahav, Zachary Teed, and Jia Deng. 2021. "RAFT-Stereo: Multilevel Recurrent Field Transforms for Stereo Matching." In *2021 International Conference on 3D Vision (3DV)*, 218–27. IEEE.

Liu, Liu, Hongdong Li, and Yuchao Dai. 2017. "Efficient Global 2d-3d Matching for Camera Localization in a Large-Scale 3d Map." In *ICCV*.

Liu, Zhe, Shunbo Zhou, Chuanzhe Suo, Peng Yin, Wen Chen, Hesheng Wang, Haoang Li, and Yun-Hui Liu. 2019. "LPD-Net: 3D Point Cloud Learning for Large-Scale Place Recognition and Environment Analysis." In *CVPR*.

Long, Jonathan L, Ning Zhang, and Trevor Darrell. 2014. "Do Convnets Learn Correspondence?" In *NIPS*.

Low, Weng Fei, and Gim Hee Lee. 2023. "Robust e-NeRF: NeRF from Sparse & Noisy Events Under Non-Uniform Motion." In *ICCV*, 18335–46.

Lowe, David G. 2004. "Distinctive Image Features from Scale-Invariant Keypoints." *IJCV* 60 (2): 91–110. https://doi.org/10.1023/B:VISI.0000029664.99615.94.

Luiten, Jonathon, Aljosa Osep, Patrick Dendorfer, Philip Torr, Andreas Geiger, Laura Leal-Taixé, and Bastian Leibe. 2021. "HOTA: A Higher Order Metric for Evaluating Multi-Object Tracking." *IJCV* 129: 548–78.

Luminar Technologies. 2024. "A Platform for Safety and Autonomy. Built from the Chip-Level up. (Luminar Iris and Sentinel)." 2024. https://web.archive.org/web/20240000000000*/https://www.luminartech.com/technology#iris.

Luo, Katie, Zhenzhen Liu, Xiangyu Chen, Yurong You, Sagie Benaim, Cheng Perng Phoo, Mark Campbell, Wen Sun, Bharath Hariharan, and Kilian Q Weinberger. 2024. "Reward Finetuning for Faster and More Accurate Unsupervised Object Discovery." *NIPS* 36.

Luo, Wenjie, Alexander G Schwing, and Raquel Urtasun. 2016. "Efficient Deep Learning for Stereo Matching." In *CVPR*.

Luo, Wenjie, Bin Yang, and Raquel Urtasun. 2018. "Fast and Furious: Real Time End-to-End 3D Detection, Tracking and Motion Forecasting with a Single Convolutional Net." In *CVPR*, 3569–77.

Lynen, Simon, Bernhard Zeisl, Dror Aiger, Michael Bosse, Joel Hesch, Marc Pollefeys, Roland Siegwart, and Torsten Sattler. 2020. "Large-Scale, Real-Time Visual–Inertial Localization Revisited." *IJRR* 39 (9): 1061–84.

Ma, Junyi, Jun Zhang, Jintao Xu, Rui Ai, Weihao Gu, and Xieyuanli Chen. 2022. "OverlapTransformer: An Efficient and Yaw-Angle-Invariant Transformer Network for LiDAR-Based Place Recognition." *RA-L* 7 (3): 6958–65.

Ma, Wei-Chiu, Shenlong Wang, Marcus A Brubaker, Sanja Fidler, and Raquel Urtasun. 2017. "Find Your Way by Observing the Sun and Other Semantic Cues." In *ICRA*, 6292–99.

Maddern, Will, Geoffrey Pascoe, Matthew Gadd, Dan Barnes, Brian Yeomans, and Paul Newman. 2020. "Real-Time Kinematic Ground Truth for the Oxford Robotcar Dataset." *arXiv Preprint arXiv:2002.10152*.

Maddern, Will, Geoffrey Pascoe, Chris Linegar, and Paul Newman. 2017. "1 Year, 1000 Km: The Oxford RobotCar Dataset." *IJRR* 36.

Manivasagam, Siva, Ioan Andrei Bârsan, Jingkang Wang, Ze Yang, and Raquel Urtasun. 2023. "Towards Zero Domain Gap: A Comprehensive Study of Realistic LiDAR Simulation for Autonomy Testing." In *ICCV*.

Mao, Jiageng, Minzhe Niu, Chenhan Jiang, Hanxue Liang, Jingheng Chen, Xiaodan Liang, Yamin Li, et al. 2021. "One Million Scenes for Autonomous Driving: ONCE Dataset."

Mao, Jiageng, Junjie Ye, Yuxi Qian, Marco Pavone, and Yue Wang. 2023. "A Language Agent for Autonomous Driving."

Marcu, Ana-Maria, Long Chen, Jan Hünermann, Alice Karnsund, Benoit Hanotte, Prajwal Chidananda, Saurabh Nair, et al. 2023. "LingoQA: Video Question Answering for Autonomous Driving." *arXiv Preprint arXiv:2312.14115*.

Martinez-Covarrubias, Julieta. 2018. "Algorithms for Large-Scale Multi-Codebook Quantization." PhD thesis, University of British Columbia.

Mentzer, Fabian, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. 2018. "Conditional Probability Models for Deep Image Compression." In *CVPR*.

Meyer, Gregory P, Ankit Laddha, Eric Kee, Carlos Vallespi-Gonzalez, and Carl K Wellington. 2019. "LaserNet: An Efficient Probabilistic 3D Object Detector for Autonomous Driving." In *CVPR*, 12677–86.

Mikolajczyk, Krystian, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, Jiri Matas, Frederik Schaffalitzky, Timor Kadir, and Luc Van Gool. 2005. "A Comparison of Affine Region Detectors." *IJCV*.

Mildenhall, Ben, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2021. "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis." *Communications of the ACM* 65 (1): 99–106.

Miles, Alistair, John Kirkham, Martin Durant, James Bourbeau, Tarik Onalan, Joe Hamman, Zain Patel, et al. 2020. *Zarr-Developers/Zarr-Python: V2.4.0* (version v2.4.0). Zenodo. https://doi.org/10.5281/zenodo.3773450.

Misra, Ishan, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. 2016. "Cross-Stitch Networks for Multi-Task Learning." In *CVPR*.

Misra, Pratap, and Per Enge. 2011. *Global Positioning System: Signals, Measurements, and Performance Revised Second Edition*. Ganga-Jamuna Press.

Moosmann, Frank, and Christoph Stiller. 2013. "Joint Self-Localization and Tracking of Generic Objects in 3D Range Data." In *ICRA*.

Moravec, Hans. 1988. *Mind Children: The Future of Robot and Human Intelligence*. Harvard University Press.

Mourikis, Anastasios I, and Stergios I Roumeliotis. 2007. "A Multi-State Constraint Kalman Filter for Vision-Aided Inertial Navigation." In *ICRA*, 3565–72. IEEE.

Mu, Jesse, Xiang Li, and Noah Goodman. 2024. "Learning to Compress Prompts with Gist Tokens." *NeurIPS* 36.

Mu, Norman, Jingwei Ji, Zhenpei Yang, Nate Harada, Haotian Tang, Kan Chen, Charles R Qi, et al. 2024. "MoST: Multi-Modality Scene Tokenization for Motion Prediction." In *CVPR*, 14988–99.

Müller, Thomas, Alex Evans, Christoph Schied, and Alexander Keller. 2022. "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding." *ACM Transactions on Graphics (TOG)* 41 (4): 1–15.

Mur-Artal, Raúl, Jose Maria Martínez Montiel, and Juan D Tardós. 2015. "ORB-SLAM: A Versatile and Accurate Monocular SLAM System." *IEEE Transactions on Robotics*.

Mur-Artal, Raúl, and Juan D. Tardós. 2017. "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras." *T-RO*. https://arxiv.org/abs/1610.06475.

Murthy, J Krishna, Miles Macklin, Florian Golemo, Vikram Voleti, Linda Petrini, Martin Weiss, Breandan Considine, et al. 2020. "gradSim: Differentiable Simulation for System Identification and Visuomotor Control." In *ICLR*.

Nardi, Luigi, Bruno Bodin, M Zeeshan Zia, John Mawer, Andy Nisbet, Paul HJ Kelly, Andrew J Davison, et al. 2015. "Introducing SLAMBench, a Performance and Accuracy Benchmarking Methodology for SLAM." In *ICRA*.

Nelson, Peter, Winston Churchill, Ingmar Posner, and Paul Newman. 2015. "From Dusk till Dawn: Localisation at Night Using Artificial Light Sources." In *ICRA*.

Novatel Inc. 2015. *An Introduction to GNSS (Second Edition)*.

Oquab, Maxime, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, et al. 2023. "DINOv2: Learning Robust Visual Features Without Supervision." *arXiv Preprint arXiv:2304.07193*.

Ort, Teddy, Krishna Murthy, Rohan Banerjee, Sai Krishna Gottipati, Dhaivat Bhatt, Igor Gilitschenski, Liam Paull, and Daniela Rus. 2019. "MapLite: Autonomous Intersection Navigation Without a Detailed Prior Map." *IEEE Robotics and Automation Letters* 5 (2): 556–63.

Pais, G Dias, Srikumar Ramalingam, Venu Madhav Govindu, Jacinto C Nascimento, Rama Chellappa, and Pedro Miraldo. 2020. "3DRegNet: A Deep Neural Network for 3D Point Registration." In *CVPR*, 7193–203.

Panek, Vojtech, Zuzana Kukelova, and Torsten Sattler. 2023. "Visual Localization Using Imperfect 3D Models from the Internet." In *CVPR*, 13175–86.

Parkinson, Bradford W, Per Enge, Penina Axelrad, and James J Spilker Jr. 1996. *Global Positioning System: Theory and Applications, Volume II*. American Institute of Aeronautics; Astronautics.

Petrenko, Aleksei, Erik Wijmans, Brennan Shacklett, and Vladlen Koltun. 2021. "Megaverse: Simulating Embodied Agents at One Million Experiences Per Second." In *ICML*, 8556–66. PMLR.

Phan-Minh, Tung, Elena Corina Grigore, Freddy A Boulton, Oscar Beijbom, and Eric M Wolff. 2020. "CoverNet: Multimodal Behavior Prediction Using Trajectory Sets." In *CVPR*.

Philion, Jonah, and Sanja Fidler. 2020. "Lift, Splat, Shoot: Encoding Images from Arbitrary Camera Rigs by Implicitly Unprojecting to 3D." In *ECCV*.

Philion, Jonah, Amlan Kar, and Sanja Fidler. 2020. "Learning to Evaluate Perception Models Using Planner-Centric Metrics." In *CVPR*.

Pitropov, Matthew, Danson Evan Garcia, Jason Rebello, Michael Smart, Carlos Wang, Krzysztof Czarnecki, and Steven Waslander. 2021. "Canadian Adverse Driving Conditions Dataset." *IJRR* 40 (4-5): 681–90.

Qi, Charles R, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation."

Qi, Charles R, Li Yi, Hao Su, and Leonidas J Guibas. 2017. "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a

Metric Space." In *NeurIPS*.

Qu, Xiaozhi, Bahman Soheilian, and Nicolas Paparoditis. 2015. "Vehicle Localization Using Mono-Camera and Geo-Referenced Traffic Signs." In *IVS*. IEEE.

Radenović, Filip, Giorgos Tolias, and Ondřej Chum. 2016. "CNN Image Retrieval Learns from BoW: Unsupervised Fine-Tuning with Hard Examples." In *ECCV*.

Radford, Alec, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, et al. 2021. "Learning Transferable Visual Models from Natural Language Supervision." In *ICML*, 8748–63. PMLR.

Radford, Alec, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2023. "Robust Speech Recognition via Large-Scale Weak Supervision." In *ICML*, 28492–518. PMLR.

Radwan, Noha, Abhinav Valada, and Wolfram Burgard. 2018. "VLocNet++: Deep Multitask Learning for Semantic Visual Localization and Odometry." *RA-L* 3 (4): 4407–14.

Redmon, Joseph, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. "You Only Look Once: Unified, Real-Time Object Detection." In *CVPR*, 779–88.

Reid, Tyler GR, Bryan Chan, Ashish Goel, Kazuma Gunning, Brian Manning, Jerami Martin, Andrew Neish, Adrien Perkins, and Paul Tarantino. 2020. "Satellite Navigation for the Age of Autonomy." In *IEEE/ION Position, Location and Navigation Symposium (PLANS)*, 342–52.

Reinke, Andrzej, Matteo Palieri, Benjamin Morrell, Yun Chang, Kamak Ebadi, Luca Carlone, and Ali-Akbar Agha-Mohammadi. 2022. "LOCUS 2.0: Robust and Computationally Efficient Lidar Odometry for Real-Time 3D Mapping." *RA-L* 7 (4): 9043–50.

Rematas, Konstantinos, Andrew Liu, Pratul P Srinivasan, Jonathan T Barron, Andrea Tagliasacchi, Thomas Funkhouser, and Vittorio Ferrari. 2022. "Urban Radiance Fields." In *CVPR*, 12932–42.

Revaud, Jerome, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. 2015. "Epicflow: Edge-Preserving Interpolation of Correspondences for Optical Flow." In *CVPR*, 1164–72.

Rhinehart, Nicholas, Jeff He, Charles Packer, Matthew A Wright, Rowan McAllister, Joseph E Gonzalez, and Sergey Levine. 2021. "Contingencies from Observations: Tractable Contingency Planning with Learned Behavior Models." In *ICRA*.

Rippel, Oren, and Lubomir Bourdev. 2017. "Real-Time Adaptive Image Compression." *ICML*.

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. 2015. "U-Net: Convolutional Networks for Biomedical Image Segmentation." In *Medical Image Computing and Computer-Assisted Intervention–*

MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18, 234–41. Springer.

Rosinol, Antoni, Marcus Abate, Yun Chang, and Luca Carlone. 2020. "Kimera: An Open-Source Library for Real-Time Metric-Semantic Localization and Mapping." In *ICRA*, 1689–96. IEEE.

Rosique, Francisca, Pedro J Navarro, Carlos Fernández, and Antonio Padilla. 2019. "A Systematic Review of Perception System and Simulators for Autonomous Vehicles Research." *Sensors* 19 (3): 648.

Rusu, Radu Bogdan, Nico Blodow, and Michael Beetz. 2009. "Fast Point Feature Histograms (FPFH) for 3D Registration." In *ICRA*.

Sadat, Abbas, Sergio Casas, Mengye Ren, Xinyu Wu, Pranaab Dhawan, and Raquel Urtasun. 2020. "Perceive, Predict, and Plan: Safe Motion Planning Through Interpretable Semantic Representations." In *ECCV*.

Sadat, Abbas, Mengye Ren, Andrei Pokrovsky, Yen-Chen Lin, Ersin Yumer, and Raquel Urtasun. 2019. "Jointly Learnable Behavior and Trajectory Planning for Self-Driving Vehicles." In *IROS*, 3949–56. IEEE.

Sarlin, Paul-Edouard, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. 2019. "From Coarse to Fine: Robust Hierarchical Localization at Large Scale." In *CVPR*, 12716–25.

Sarlin, Paul-Edouard, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. 2020. "SuperGlue: Learning Feature Matching with Graph Neural Networks." In *CVPR*, 4938–47.

Sattler, Torsten, Michal Havlena, Filip Radenovic, Konrad Schindler, and Marc Pollefeys. 2015. "Hyperpoints and Fine Vocabularies for Large-Scale Location Recognition." In *ICCV*.

Sattler, Torsten, Bastian Leibe, and Leif Kobbelt. 2011. "Fast Image-Based Localization Using Direct 2d-to-3d Matching." In *ICCV*.

Sattler, Torsten, Will Maddern, Carl Toft, Akihiko Torii, Lars Hammarstrand, Erik Stenborg, Daniel Safari, et al. 2018. "Benchmarking 6DOF Outdoor Visual Localization in Changing Conditions." In *CVPR*, 8601–10.

Sattler, Torsten, Akihiko Torii, Josef Sivic, Marc Pollefeys, Hajime Taira, Masatoshi Okutomi, and Tomas Pajdla. 2017. "Are Large-Scale 3D Models Really Necessary for Accurate Visual Localization?" In *CVPR*.

Sattler, Torsten, Tobias Weyand, Bastian Leibe, and Leif Kobbelt. 2012. "Image Retrieval for Image-Based Localization Revisited." In *BMVC*.

Sattler, Torsten, Qunjie Zhou, Marc Pollefeys, and Laura Leal-Taixe. 2019. "Understanding the Limitations of CNN-Based Absolute Camera Pose Regression." In *CVPR*.

Schonberger, Johannes L, and Jan-Michael Frahm. 2016. "Structure-from-Motion Revisited." In *CVPR*, 4104–13.

Schönberger, Johannes L, Marc Pollefeys, Andreas Geiger, and Torsten Sattler. 2018. "Semantic Visual Localization." *JPRS*.

Schreiber, Markus, Carsten Knöppel, and Uwe Franke. 2013. "Lane-Loc: Lane Marking Based Localization Using Highly Accurate Maps." In *IV*.

Schrittwieser, Julian, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, et al. 2020. "Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model." *Nature* 588 (7839): 604–9.

Sculley, David, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. 2015. "Hidden Technical Debt in Machine Learning Systems." *NIPS* 28.

Sener, Ozan, and Vladlen Koltun. 2018. "Multi-Task Learning as Multi-Objective Optimization." In *NIPS*.

Shacklett, Brennan, Luc Guy Rosenzweig, Zhiqiang Xie, Bidipta Sarkar, Andrew Szot, Erik Wijmans, Vladlen Koltun, Dhruv Batra, and Kayvon Fatahalian. 2023. "An Extensible, Data-Oriented Architecture for High-Performance, Many-World Simulation." *ACM Transactions on Graphics (TOG)* 42 (4): 1–13.

Shan, Tixiao, Brendan Englot, Drew Meyers, Wei Wang, Carlo Ratti, and Daniela Rus. 2020. "LIO-SAM: Tightly-Coupled LiDAR Inertial Odometry via Smoothing and Mapping." In *IROS*, 5135–42. IEEE.

Shannon, Claude E. 1948. "A Mathematical Theory of Communication." *The Bell System Technical Journal* 27 (3): 379–423.

Shen, Yuan, Bhargav Chandaka, Zhi-hao Lin, Albert Zhai, Hang Cui, David Forsyth, and Shenlong Wang. 2024. "Sim-on-Wheels: Physical World in the Loop Simulation for Self-Driving." *ICRA*.

Shi, Shaoshuai, Xiaogang Wang, and Hongsheng Li. 2019. "PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud." In *CVPR*.

Shotton, Jamie, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. 2013. "Scene Coordinate Regression Forests for Camera Relocalization in RGB-D Images." In *CVPR*.

Siegwart, Roland, Illah Reza Nourbakhsh, and Davide Scaramuzza. 2011. *Introduction to Autonomous Mobile Robots*. MIT press.

Sucar, Edgar, Shikun Liu, Joseph Ortiz, and Andrew J Davison. 2021. "iMap: Implicit Mapping and Positioning in Real-Time." In *ICCV*, 6229–38.

Sun, Pei, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard,

Vijaysai Patnaik, Paul Tsui, James Guo, et al. 2020. "Scalability in Perception for Autonomous Driving: Waymo Open Dataset." In *CVPR*, 2446–54.

Tancik, Matthew, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. 2022. "Block-NeRF: Scalable Large Scene Neural View Synthesis." In *CVPR*, 8248–58.

Teed, Zachary, and Jia Deng. 2020. "RAFT: Recurrent All-Pairs Field Transforms for Optical Flow." In *ECCV*, 402–19. Springer.

———. 2021a. "DROID-SLAM: Deep Visual Slam for Monocular, Stereo, and RGB-D Cameras." *NeurIPS* 34: 16558–69.

———. 2021b. "RAFT-3D: Scene Flow Using Rigid-Motion Embeddings." In *CVPR*, 8375–84.

Teichmann, Marvin, Michael Weber, Marius Zoellner, Roberto Cipolla, and Raquel Urtasun. 2018. "MultiNet: Real-Time Joint Semantic Reasoning for Autonomous Driving." In *IV*.

Thrun, Sebastian. 2007. "Simultaneous Localization and Mapping." In *Robotics and Cognitive Approaches to Spatial Mapping*, 13–41. Springer.

Thrun, Sebastian, Wolfram Burgard, and Dieter Fox. 2005. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press.

Thrun, Sebastian, and Michael Montemerlo. 2006. "The Graph SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures." *IJRR*.

Toderici, George, Sean M O'Malley, Sung Jin Hwang, Damien Vincent, David Minnen, Shumeet Baluja, Michele Covell, and Rahul Sukthankar. 2016. "Variable Rate Image Compression with Recurrent Neural Networks." In *ICLR*.

Toderici, George, Damien Vincent, Nick Johnston, Sung Jin Hwang, David Minnen, Joel Shor, and Michele Covell. 2017. "Full Resolution Image Compression with Recurrent Neural Networks." In *CVPR*, 5435–43.

Tolias, Giorgos, Ronan Sicre, and Hervé Jégou. 2016. "Particular Object Retrieval with Integral Max-Pooling of CNN Activations." In *ICLR*.

Tombari, Federico, Samuele Salti, and Luigi Di Stefano. 2010. "Unique Signatures of Histograms for Local Surface Description." In *ECCV*.

Tonderski, Adam, Carl Lindström, Georg Hess, William Ljungbergh, Lennart Svensson, and Christoffer Petersson. 2024. "NeuRAD: Neural Rendering for Autonomous Driving."

Torii, Akihiko, Relja Arandjelovic, Josef Sivic, Masatoshi Okutomi, and Tomas Pajdla. 2015. "24/7 Place Recognition by View Synthesis." In *CVPR*, 1808–17.

Torii, Akihiko, Josef Sivic, Masatoshi Okutomi, and Tomas Pajdla. 2015. "Visual Place Recognition with Repetitive Structures." *TPAMI*, 1–14.

Tran, Richard, Janice Lan, Muhammed Shuaibi, Brandon M Wood, Siddharth Goyal, Abhishek Das, Javier Heras-Domingo, et al. 2023. "The Open Catalyst 2022 (OC22) Dataset and Challenges for Oxide Electrocatalysts." *ACS Catalysis* 13 (5): 3066–84.

Triggs, Bill, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. 1999. "Bundle Adjustment—A Modern Synthesis." In *International Workshop on Vision Algorithms*.

Tufte, Edward R. 2001. *The Visual Display of Quantitative Information*. Second Edition. Cheshire, Connecticut: Graphics Press. https://www.edwardtufte.com/tufte/books_vdqi.

Turki, Haithem, Jason Y Zhang, Francesco Ferroni, and Deva Ramanan. 2023. "SUDS: Scalable Urban Dynamic Scenes." In *CVPR*, 12375–85.

Ulyanov, Dmitry, Andrea Vedaldi, and Victor Lempitsky. 2017. "Improved Texture Networks: Maximizing Quality and Diversity in Feed-Forward Stylization and Texture Synthesis." In *CVPR*, 6924–32.

Uy, Mikaela Angelina, and Gim Hee Lee. 2018. "PointNetVLAD: Deep Point Cloud Based Retrieval for Large-Scale Place Recognition." In *CVPR*.

Valada, Abhinav, Noha Radwan, and Wolfram Burgard. 2018. "Deep Auxiliary Learning for Visual Localization and Odometry."

van den Berg, Jur, Sachin Patil, and Ron Alterovitz. 2012. "Motion Planning Under Uncertainty Using Iterative Local Optimization in Belief Space." *IJRR* 31 (11): 1263–78.

Velodyne Lidar, Inc. 2021. "Velodyne Lidar Announces Next-Generation Velabit™ Sensor, Media Reports." 2021. https://velodynelidar.com/media-coverage/next-generation-velabit/.

Verghese, Simon. 2019. "Bringing 3D Perimeter Lidar to Partners." 2019. https://waymo.com/blog/2019/03/bringing-3d-perimeter-lidar-to-partners/.

Wan, Guowei, Xiaolong Yang, Renlan Cai, Hao Li, Yao Zhou, Hao Wang, and Shiyu Song. 2018. "Robust and Precise Vehicle Localization Based on Multi-Sensor Fusion in Diverse City Scenes." In *ICRA*.

Wang, Haiping, Yuan Liu, Zhen Dong, and Wenping Wang. 2022. "You Only Hypothesize Once: Point Cloud Registration with Rotation-Equivariant Descriptors." In *Proceedings of the 30th ACM International Conference on Multimedia*, 1630–41.

Wang, Jingkang, Siva Manivasagam, Yun Chen, Ze Yang, Ioan Andrei Bârsan, Anqi Joyce Yang, Wei-chiu Ma, and Raquel Urtasun. 2022.

"CADsim: Robust and Scalable in-the-Wild 3D Reconstruction for Realistic and Controllable Sensor Simulation." In *CoRL*.

Wang, Peng, Ruigang Yang, Binbin Cao, Wei Xu, and Yuanqing Lin. 2018. "DeLS-3D: Deep Localization and Segmentation with a 3D Semantic Map." In *CVPR*.

Wang, Sen, Ronald Clark, Hongkai Wen, and Niki Trigoni. 2017. "DeepVO: Towards End-to-End Visual Odometry with Deep Recurrent Convolutional Neural Networks." In *ICRA*, 2043–50. IEEE.

Wang, Shenlong. 2021. "Deep Structured Models for Spatial Intelligence." PhD thesis, University of Toronto (Canada).

Wang, Tsun-Hsuan, Sivabalan Manivasagam, Ming Liang, Bin Yang, Wenyuan Zeng, James Tu, and Raquel Urtasun. 2020. "V2VNet: Vehicle-to-Vehicle Communication for Joint Perception and Prediction."

Wang, Yue, and Justin M Solomon. 2019. "Deep Closest Point: Learning Representations for Point Cloud Registration." In *ICCV*, 3523–32.

Wang, Zirui, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. 2021. "NeRF−−: Neural Radiance Fields Without Known Camera Parameters." *arXiv Preprint arXiv:2102.07064*.

Warburg, Frederik, Soren Hauberg, Manuel Lopez-Antequera, Pau Gargallo, Yubin Kuang, and Javier Civera. 2020. "Mapillary Street-Level Sequences: A Dataset for Lifelong Place Recognition." In *CVPR*, 2626–35.

Waymo LLC. 2024. "Advice Letter 0002 (Tier 2) to the Public Utilities Commission of the State of California," 2024. https://web.archive.org/web/20240703035846/https://www.cpuc.ca.gov/-/media/cpuc-website/divisions/consumer-protection-and-enforcement-division/documents/tlab/av-programs/waymo-llc-cpuc-advice-letter-0002-tier-2--january-2024-passenger-safety-plan-update-january-192024.pdf.

Welzel, Andre, Pierre Reisdorf, and Gerd Wanielik. 2015. "Improving Urban Vehicle Localization with Traffic Sign Recognition." In *ICITS*. IEEE.

Weston, Rob, Matthew Gadd, Daniele De Martini, Paul Newman, and Ingmar Posner. 2022. "Fast-MbyM: Leveraging Translational Invariance of the Fourier Transform for Efficient and Accurate Radar Odometry." In *ICRA*, 2186–92. IEEE.

Wiesmann, Louis, Tiziano Guadagnino, Ignacio Vizzo, Giorgio Grisetti, Jens Behley, and Cyrill Stachniss. 2022. "DCPCR: Deep Compressed Point Cloud Registration in Large-Scale Outdoor Environments." *IEEE Robotics and Automation Letters* 7 (3): 6327–34.

Wikipedia Contributors. 2019. "JPEG - Wikipedia." https://en.wikipedia.org/wiki/JPEG.

———. 2023. “Radio Navigation — Wikipedia, the Free Encyclopedia.” 2023. https://en.wikipedia.org/wiki/Radio_navigation.

———. 2024a. “Deflate — Wikipedia, the Free Encyclopedia.” 2024. https://en.wikipedia.org/wiki/Deflate.

———. 2024b. “Snappy (Compression) — Wikipedia, the Free Encyclopedia.” 2024. https://en.wikipedia.org/wiki/Snappy_(compression).

Wilson, Benjamin, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, et al. 2023. “Argoverse 2: Next Generation Datasets for Self-Driving Perception and Forecasting.” *arXiv Preprint arXiv:2301.00493*.

Wolcott, Ryan W, and Ryan M Eustice. 2014. “Visual Localization Within LIDAR Maps for Automated Urban Driving.” In *IROS*.

———. 2015. “Fast LIDAR Localization Using Multiresolution Gaussian Mixture Maps.” In *ICRA*, 2814–21. IEEE.

Xu, Wei, Yixi Cai, Dongjiao He, Jiarong Lin, and Fu Zhang. 2022. “Fast-LIO2: Fast Direct LiDAR-Inertial Odometry.” *IEEE Transactions on Robotics* 38 (4): 2053–73.

Yan, Yunzhi, Haotong Lin, Chenxu Zhou, Weijie Wang, Haiyang Sun, Kun Zhan, Xianpeng Lang, Xiaowei Zhou, and Sida Peng. 2024. “Street Gaussians for Modeling Dynamic Urban Scenes.” In *ECCV*.

Yang, Anqi Joyce, Sergio Casas, Nikita Dvornik, Sean Segal, Yuwen Xiong, Jordan Sir Kwang Hu, Carter Fang, and Raquel Urtasun. 2023. “LabelFormer: Object Trajectory Refinement for Offboard Perception from LiDAR Point Clouds.” In *CoRL*, 3364–83. PMLR.

Yang, Anqi Joyce, Can Cui, Ioan Andrei Bârsan, Raquel Urtasun, and Shenlong Wang. 2021. “Asynchronous Multi-View SLAM.” In *ICRA*.

Yang, Bin, Runsheng Guo, Ming Liang, Sergio Casas, and Raquel Urtasun. 2020. “RadarNet: Exploiting Radar for Robust Perception of Dynamic Objects.” In *ECCV*, 496–512. Springer.

Yang, Bin, Wenjie Luo, and Raquel Urtasun. 2018. “PIXOR: Real-Time 3D Object Detection from Point Clouds.” In *CVPR*, 7652–60.

Yang, Honghui, Sha Zhang, Di Huang, Xiaoyang Wu, Haoyi Zhu, Tong He, Shixiang Tang, et al. 2024. “UniPAD: A Universal Pre-Training Paradigm for Autonomous Driving.” In *CVPR*.

Yang, Jiawei, Boris Ivanovic, Or Litany, Xinshuo Weng, Seung Wook Kim, Boyi Li, Tong Che, et al. 2023. “EmerNeRF: Emergent Spatial-Temporal Scene Decomposition via Self-Supervision.” *arXiv Preprint arXiv:2311.02077*.

Yang, Weixiang, Qi Li, Wenxi Liu, Yuanlong Yu, Yuexin Ma, Shengfeng He, and Jia Pan. 2021. “Projecting Your View Attentively: Monocular Road Scene Layout Estimation via Cross-View Transformation.” In *CVPR*, 15536–45.

Yang, Yibo, Stephan Mandt, Lucas Theis, et al. 2023. “An Introduc-

tion to Neural Data Compression." *Foundations and Trends® in Computer Graphics and Vision* 15 (2): 113–200.

Yang, Yongxin, and Timothy M Hospedales. 2017. "Trace Norm Regularised Deep Multi-Task Learning." In *ICLR Workshop Track*.

Yang, Ze, George Chen, Haowei Zhang, Ta Kevin, Ioan Andrei Bârsan, Daniel Murphy, Sivabalan Manivasagam, and Raquel Urtasun. 2024. "UniCal: Unified Neural Sensor Calibration." In *ECCV*.

Yang, Ze, Yun Chen, Jingkang Wang, Sivabalan Manivasagam, Wei-Chiu Ma, Anqi Joyce Yang, and Raquel Urtasun. 2023. "UniSim: A Neural Closed-Loop Sensor Simulator." In *CVPR*, 1389–99.

Yao, Yao, Zixin Luo, Shiwei Li, Tianwei Shen, Tian Fang, and Long Quan. 2019. "Recurrent MVSNet for High-Resolution Multi-View Stereo Depth Inference." In *CVPR*, 5525–34.

Yoneda, Keisuke, Hossein Tehrani, Takashi Ogawa, Naohisa Hukuyama, and Seiichi Mita. 2014. "Lidar Scan Feature for Localization with Highly Precise 3-D Map." In *IV*.

You, Yurong, Katie Z Luo, Xiangyu Chen, Junan Chen, Wei-Lun Chao, Wen Sun, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. 2022. "Hindsight Is 20/20: Leveraging Past Traversals to Aid 3d Perception."

You, Yurong, Katie Luo, Cheng Perng Phoo, Wei-Lun Chao, Wen Sun, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. 2022. "Learning to Detect Mobile Objects from Lidar Scans Without Labels." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1130–40.

Zagoruyko, Sergey, and Nikos Komodakis. 2015. "Learning to Compare Image Patches via Convolutional Neural Networks." In *CVPR*.

Zamir, Amir R, Asaad Hakeem, and Richard Szeliski. 2016. *Large-Scale Visual Geo-Localization*. Springer. https://doi.org/10.1007/978-3-319-25781-5.

Zbontar, Jure, and Yann LeCun. 2015. "Computing the Stereo Matching Cost with a Convolutional Neural Network." In *CVPR*.

Zeng, Wenyuan, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. 2019. "End-to-End Interpretable Neural Motion Planner." In *CVPR*, 8660–69.

Zeng, Wenyuan, Shenlong Wang, Renjie Liao, Yun Chen, Bin Yang, and Raquel Urtasun. 2020. "DSDNet: Deep Structured Self-Driving Network." In *ECCV*.

Zhang, Chris, Wenjie Luo, and Raquel Urtasun. 2018. "Efficient Convolutions for Real-Time Semantic Segmentation of 3D Point Clouds." In *3DV*.

Zhang, Jeffrey O, Alexander Sax, Amir Zamir, Leonidas Guibas, and Jitendra Malik. 2020. "Side-Tuning: A Baseline for Network Adap-

tation via Additive Side Networks." In *ECCV*.

Zhang, Ji, and Sanjiv Singh. 2014. "LOAM: Lidar Odometry and Mapping in Real-Time." In *RSS*.

Zhang, Lunjun, Anqi Joyce Yang, Yuwen Xiong, Sergio Casas, Bin Yang, Mengye Ren, and Raquel Urtasun. 2023. "Towards Unsupervised Object Detection from LiDAR Point Clouds." In *CVPR*, 9317–28.

Zhang, Wenxiao, and Chunxia Xiao. 2019. "PCAN: 3D Attention Map Learning Using Contextual Information for Point Cloud Based Retrieval." In *CVPR*.

Zhang, Zhishuai, Jiyang Gao, Junhua Mao, Yukai Liu, Dragomir Anguelov, and Congcong Li. 2020. "STINet: Spatio-Temporal-Interactive Network for Pedestrian Detection and Trajectory Prediction." In *CVPR*.

Zhang, Zichao, and Davide Scaramuzza. 2020. "Fisher Information Field: An Efficient and Differentiable Map for Perception-Aware Planning." *arXiv Preprint arXiv:2008.03324*.

Zhao, Hengshuang, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. 2017. "Pyramid Scene Parsing Network." In *CVPR*.

Zheng, Stephan, Yisong Yue, and Jennifer Hobbs. 2016. "Generating Long-Term Trajectories Using Deep Hierarchical Networks." In *NIPS*.

Zhou, Brady, Philipp Krähenbühl, and Vladlen Koltun. 2019. "Does Computer Vision Matter for Action?" *Science Robotics* 4 (30).

Zhou, Qian-Yi, Jaesik Park, and Vladlen Koltun. 2016. "Fast Global Registration." In *ECCV*.

Zhou, Qunjie, Sérgio Agostinho, Aljoša Ošep, and Laura Leal-Taixé. 2022. "Is Geometry Enough for Matching in Visual Localization?" In *ECCV*, 407–25. Springer.

Zhou, Yao, Guowei Wan, Shenhua Hou, Li Yu, Gang Wang, Xiaofei Rui, and Shiyu Song. 2020. "DA4AD: End-to-End Deep Attention Aware Features Aided Visual Localization for Autonomous Driving." In *ECCV*.

Zhu, Sijie, Linjie Yang, Chen Chen, Mubarak Shah, Xiaohui Shen, and Heng Wang. 2023. "R2Former: Unified Retrieval and Reranking Transformer for Place Recognition." In *CVPR*, 19370–80.

Zhu, Zihan, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R Oswald, and Marc Pollefeys. 2022. "NICE-SLAM: Neural Implicit Scalable Encoding for SLAM." In *CVPR*, 12786–96.

Ziegler, Julius, Henning Lategahn, Markus Schreiber, Christoph G Keller, Carsten Knoppel, Jochen Hipp, Martin Haueis, and Christoph Stiller. 2014. "Video Based Localization for Bertha." In *IV*.