D2
Learning Techniques for Integrating Formalized Knowledge and Network Knowledge for Training Reduction and Performance Improvement

Daniel Bogdoll[5], Antoine Detailleur[3], Ludger van Elst[7], Tobias Gleißner[4], Tim Joseph[5], Johann Kelsch[6], Tobias Latka[2], Mohsin Munir[7], Syed Tahseen Raza Rizvi[7], Jörg Reichardt[1], Matthias Reuse[9], Stefan Rudolph[2], Alexander Sagel[3], Gurucharan Srinivas[6], Vera Stehr[9], Tino Werner[9], and Julian Wörmann[3]

[1]Continental AG
[2]Elektronische Fahrwerksysteme GmbH
[3]fortiss GmbH
[4]Fraunhofer FOKUS
[5]FZI Research Center for Information Technology
[6]German Aerospace Center (DLR e.V)
[7]German Research Center for Artificial Intelligence (DFKI GmbH)
[8]OFFIS e.V.
[9]Valeo Schalter und Sensoren GmbH

August 3, 2022

Gefördert durch:

Bundesministerium
für Wirtschaft
und Energie

aufgrund eines Beschlusses
des Deutschen Bundestages

Contents

Chapter 1

Introduction

1.1 KI Wissen Project Description

The research project KI Wissen develops methods for integrating existing knowledge into the data-driven AI functions of autonomous vehicles. The goal of the project is to create a comprehensive ecosystem for the integration of knowledge into the training and safeguarding of AI functions.

Up until now, the development of AI functions has been purely driven by data. However, this data-driven approach requires enormous amounts of data for the training and validation of AI functions, with the collection and processing of this data being very resource-intensive and expensive. In addition to the dependence on extensive amounts of data, data-based AI processes have another weakness: they are still generally black-box models for which the decision-making process cannot be directly reconstructed. Previous research approaches to solving these issues have focused on optimizing the data needed to train and validate AI functions.

The research project KI Wissen addresses the challenge in a novel way and develops methods and tools for integrating knowledge into machine learning. The project consortium, consisting of 16 partners from industry, research and science under the leadership of Continental Automotive GmbH, investigates how existing, traffic-relevant knowledge - in the form of traffic rules, mathematical-physical conditions and also social norms - can be integrated into the data-driven AI functions of autonomous vehicles. By combining conventional data-based AI methods with the knowledge- or rule-based methods developed in the project, the basis for the training and validation of AI functions is largely redefined. This now includes not only data, but knowledge, i.e. data, their relations and information. The further development carried out in the project addresses the central challenges on the way to autonomous driving: the generalization of AI to phenomena with a small data basis, the increase in the stability of the trained AI to disturbances in the data, the data efficiency, the plausibility and the validation of AI-supported functions as well as the increase in functional quality.

A broad variety of methods will be developed and investigated in KI Wissen: methods for knowledge integration, knowledge extraction and knowledge conformity. The methods developed in the project will be evaluated and demonstrated using three defined use cases. Specifically, the following core scientific and technical innovations will be addressed:

Scientific innovations:

- Development of approaches to identify and formalize relevant domain knowledge for L3 to L5 driving functions;
- Development of evaluated approaches for knowledge-based training of AI components;
- Development of learning techniques for training reduction and performance improvement;
- Creation of an ecosystem for knowledge as a basis for efficient training and the safeguarding of AI components based on formalized knowledge.

Technical innovations:

- Development of reusable qualified and formalized modules for application, safeguarding, and network knowledge (e.g., formalized road traffic regulations with exceptions);
- Creation of a tool kit of formalized knowledge and corresponding AI methods for companies and research;
- Creation of demonstrators with knowledge-based AI components.

1.2 Deliverables in the KI Wissen Project

The research project KI Wissen will work on the set goals over a period of three years. It pursues the step-by-step improvement of the methods to be developed in three successive project steps, the project increments. At the end of each project increment, which goes hand in hand with the project milestones, the (interim) results achieved in the work packages are documented in the form of deliverables. The main objective of the deliverable documents is to communicate the achieved results within the project and also beyond it, towards the involved stakeholders, funding body, and other research activities. There are in total seven deliverables which were defined in KI Wissen. In terms of content, they are oriented towards the technical and scientific project innovations:

ID	Title	Release Level	Type
D1	Catalog, characterization, and representation of relevant domain knowledge for L3 to L5 driving functions	public	Document
D2	Learning techniques for integrating formalised knowledge and network knowledge for training reduction and performance improvement	public	Document/ White paper
D3	Methods for extracting knowledge from a data-driven AI function	public	Document
D4	Methods for data-independent validation of the predictions and decisions of an AI function	public	Document
D5	Reusable, qualified and formalized modules of application, safeguarding and network knowledge	VDA LI	Software and specifications
D6	Construction kit of knowledge-based AI methods for companies and research	VDA LI	Software and specifications
D7	Demonstrators with knowledge-based AI components	project-internal	Software and specifications / Demonstration

1.3 Overview

The goal of TP1 is to develop methods and modules (software) to integrate relevant knowledge into data-driven AI functions for autonomous driving. The approaches investigated in TP1 build on the hypothesis that integration of *a priori* domain knowledge into sub-symbolic learning methods can increase the training and data efficiency of a neural network as already explicitly known concepts do not have to be learned laboriously from real-world or synthetic example data. In particular, general or more abstract knowledge can address the inherent weakness of ML algorithms that their reactions to scenarios rarely or not at all occurring in the training data are very difficult to predict. Often, however, precisely such situations are critical and must be mastered in any case. Thus, on the one hand, the incorporation of knowledge can lead to considerable

savings in the generation of training data and the procurement and operation of hardware. On the other hand, it simultaneously increases the quality and reliability of AI-based driving functions in general - but especially in critical traffic situations. The integration of domain knowledge into AI-based driving functions can happen at any of the three levels of an autonomous driving stack: at the level of sensing and environmental perception, at the level of situation understanding and interpretation, and at the level of decision making, planning, and control.

The envisioned integration of knowledge into AI models can be achieved in different ways. One possibility is keep state-of-the-art deep learning architectures, but to redesign the training process. That is, the optimization problem of learning a neural network from example data is adapted in such a way that introduced knowledge is not violated. Another possibility is to adapt AI architectures so that knowledge can be incorporated there directly.

Chapter 2 follows the first option aiming at implicit or explicit integration of knowledge components into the training process of data-driven learning methods. Knowledge-based inputs and optimization targets shall improve the data efficiency, performance and generalization capability of existing architectures and approaches. Concepts for meta-learning methods for the detection of underrepresented instances as well as structural causal models for the physically correct generation of trajectories are presented. Moreover, the integration of predefined object models to reduce interference and false detections is investigated. Further approaches aim at knowledge- and rule-based modeling of reward functions to influence the behavior of reinforcement learning agents in planning scenarios.

Chapter 3 addresses the second option and describes concepts for novel architectures that are capable of incorporating *a priori* knowledge directly. Specifically, a concept is presented that uses knowledge in form of priors about locations where objects may appear or vanish or about potential occlusions to enhance traffic scene prediction. Another concept aims an architecture that includes knowledge about lidar hardware for finding optimal configurations and misalignment states of lidar sensors. Two concepts aim at improved interpretation of the environment: One approach uses knowledge graphs about traffic scenes as background knowledge as priors for pedestrian detection and segmentation, another one focuses on explicit pairwise relationships for improving detection methods for objects like traffic signs. Finally, two architectures for integrating rule-based or normative knowledge into driving functions are being described. The first approach aims at checking whether a vehicle is currently in a desirable state. The second architecture uses a rule-engine to find optimal driving trajectories or distill behavior constrained by rules into neural driving policy. This approach will allow to integrate knowledge about scene structures and also to generate interpretable descriptions of the current scene.

The concepts described in this document are the building blocks for implementation, evaluation, and improvement planned for the subsequent project increments in 2022 and 2023. Each concept will support at least one of the use cases that guide KI Wissen, namely pedestrian detection (UC1), lane change (UC2), and rule exception (UC3). The concepts are designed for integration

into the overall project demonstrator (TP4) as well as for refinement by specific knowledge manifestations developed in the respective complementary work packages of TP1.

Chapter 2

AP1.1: Concepts for training methods that are based on *a priori* knowledge

2.1 Introduction

The goal of this work package is the implicit or explicit integration of knowledge components into the training process of data-driven learning methods. Additional, knowledge-based inputs and optimization targets shall improve the data efficiency, performance and generalization capability of existing architectures and approaches. In particular, the consideration of external knowledge is intended to allow the system to better handle critical traffic situations that cannot otherwise be resolved with reliable accuracy due to the lack of a sufficient and representative data base.

In this work package, various methods for enriching the training procedure with knowledge are investigated and developed. One principle approach is to extend the optimization function, which is minimized during the learning of a neural network, by further regularization terms and constraints. These additional terms guide the learning process via an increased loss for predictions that do not correspond to formalized knowledge. This approach can also be applied selectively to individual layers of the network. Another approach is to utilize knowledge in the data processing and generation process, such that existing learning pipelines can be used efficiently. Further aspects concern the extension of the training strategies by means of knowledge transfer paradigms, e.g., Active Learning or Transfer Learning, as well as the adaptive adjustment of training parameters.

In the following, the partners involved present their concepts regarding the consideration and integration of knowledge during the training process. In detail, meta-learning methods that support the detection of underrepresented

instances (Section 2.2) as well as structural causal models for the physically correct generation of trajectories are proposed (Section 2.3). Furthermore, the integration of predefined object models to reduce interference and false detections is investigated (Section 2.4). Further approaches aim at knowledge- and rule-based modeling of reward functions to influence the behavior of reinforcement learning agents in planning scenarios (Sections 2.5 and 2.6).

2.2 fortiss - Knowledge Guided Few-shot Learning

In our concepts, we employ the meta-learning paradigm in order to guide the training process. We predominantly consider knowledge components concerning the organization and augmentation of the data. Combined with meta-learning, we propose knowledge augmented training strategies that allows for few-shot learning as well as reduction in training time. In this way, we aim for systems that are more invariant to changes in environmental and situational conditions and that can seamlessly adapt to underrepresented scenarios. While we focus on the perception use case, due to the versatility of the meta-learning framework the general principles can also be applied to planning use cases.

2.2.1 Introduction

Meta-learning or *Learning to learn* allows acquiring knowledge about how to solve different tasks and using this knowledge to solve new or related tasks. Classical machine learning models are trained based on the assumption that the model is applied in a scenario that follows the same statistical laws as the training data. Conversely, meta-learning models are trained with the aim to infer these very laws on demand. Let us illustrate this difference on an example from automotive applications. Consider a model that predicts the trajectory of traffic participants from a sequence of video frames. In classical machine learning, we assume that the considered traffic participants in the real world do not differ significantly from those in the data used for training the model. Therefore, our model hopefully captures the visual features associated with the considered traffic participants, as well as their movement patterns. Now, let us consider a situation where the independence assumption between training and testing data does no longer hold - also referred to as domain shift. This could happen, for instance, because the training data was collected from highway motion captures. Hence, the learned visual features and motion patterns are predominantly inferred from objects like trucks or passenger cars and are unsuitable to recognize or track streetcars, bicycles, or pedestrians.

Solving the problem of motion prediction requires learning certain semantic principles that are inherent to the problem irrespective of the considered traffic participants. These principles may encompass learning low-level visual features

that are not unique to certain traffic participant classes, or the mere fact that some kind of visual features needs to be learned in the first place. Meta-learning refers to all machine learning models that aim at acquiring such inherent principles rather than learning a particular data distribution. Typically, this is achieved by dividing the training procedure into two stages. The second stage, referred to as the *adaptation stage*, is essentially identical to the training procedure of classical machine learning. That is to say, a set of training samples is presented to the model to make the model behave in the same way during deployment, as the training data suggests. During the first stage, the so-called *meta-learning stage*, the model is trained with the objective to carry out the adaptation stage as effortless and using as few training samples as possible.

The merits of such an approach are that the meta-knowledge transferred to new tasks is very versatile. Parameter initialization is one of the main goals of meta-learning and is often employed in few-shot scenarios. The learning algorithm learns here how to quickly adapt to only a limited number of samples [1]. However, meta-learning also comprises learning of embeddings. In this case, network weights are predicted as a simple forward pass from the support set to the parameter space [2]. Curriculum learning, i.e. learning the order of samples for training purposes, or learning data augmentation policies [3] are further directions pursued by meta-learning.

As stated in the project description, fortiss explores the capability of neural networks to incorporate external knowledge from heterogeneous sources in the scope of AP 1.1. Practically, this requires systems that remain highly adaptable in face of limited data sets or sparse annotations. Meta-learning is seen as one of the most promising candidates to maintain such adaptability. In meta-learning based systems, external knowledge is usually provided in two different forms. During the meta-learning stage, the system must be taught what principles are inherent to the problem independent of the particular task or situation. This is carried out indirectly, most commonly, by dividing the training set into subsets, such that each subset corresponds to one particular task/situation [4]. In the automotive scenario mentioned before, to name an example, each subset might contain training data for one particular vehicle type. In the adaptation stage, training data provide situation-specific insights to the system. This requires knowledge about the expected statistical properties of the desired application scenario.

In both the meta-learning and the adaptation stage, incorporating external knowledge is not limited to selecting and organizing the training data. Data augmentation and pre-processing are also utilized to integrate knowledge into the system. For instance, image filters are used to emulate specific weather conditions while cropping and zooming can come in handy to increase adaptability to different scales and distances. To account for possible occlusion patterns, region masking can be used.

Main contributions/improvements

- Employ external knowledge components in order to build suitable training data organization and augmentation strategies.
- Utilize the meta-learning framework to learn robust data-driven models from only a few annotated samples.
- Exploit the adaptability of meta-learned models with the focus on tracking traffic participants under varying environmental conditions.

2.2.2 Baseline

At fortiss, we are emphasizing the detection and tracking of pedestrians within AP 1.1. In practice, this task is often challenged by occlusion, since objects such as cars or traffic signs can obfuscate visual detection of pedestrians (Use Case 1). More generally, fortiss aims at developing pedestrian detection and tracking models that adapt to different conditions regarding

- weather,
- surroundings,
- traffic situation,
- occlusions.

This section provides an overview over recent developments that will serve as baselines for our efforts. First, we consider how meta-learning is used to make machine learning more adaptable to particular situations. Additionally, we review the most commonly employed frameworks for object detection as a generalization of pedestrian detection. We then proceed to discuss works that employ meta-learning to render object detection more versatile. Finally, we briefly sketch how meta-learning can be used in the context of Reinforcement Learning, that is in a planning scenario.

At this point, we would like to briefly point out that meta-learning is related to other learning paradigms, e.g., transfer learning, domain adaptation, and continual learning. In contrast to meta-learning, these learning strategies do not incorporate a meta-objective and thus achieve knowledge transfer differently. However, as a very versatile methodology, meta-learning can be used to advance the aforementioned learning approaches. We refer the interested reader to [5] for an exhaustive overview.

Meta-Learning

Most meta-learning frameworks typically assume the existence of tasks $\mathcal{T}_1, \mathcal{T}_2, \ldots$ to which the model is applied. The exact definition of *task* varies across different works in the literature. Generally, a task encompasses a data distribution and an objective function $\mathcal{L}_i$, depending on the said distribution. In *Model-Agnostic*

Meta-Learning (MAML) [1], the set of tasks are to be solved by a parametrized function f_θ trained by gradient descent. The parameter vector θ is optimized in an alternating manner. In the first part of each iteration, an updated parameter vector θ_i' is computed for each task $\mathcal{T}_i$ by making a step of size α in the direction of the steepest descent of $\mathcal{L}_i(\theta)$,

$$\theta_i' \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_i(\theta).$$

In the second part of each iteration, an update of θ is computed by making a β-sized step in the steepest descent direction of the task loss sum $\sum_i \mathcal{L}_i(\theta_i')$, i.e.,

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_i \mathcal{L}_i(\theta_i').$$

Note that θ_i' depends on θ and the aim of the optimization procedure is to get θ to a point where, for any possible task $\mathcal{T}$, it can be easily updated to a task-specific optimum θ' using only a few data samples and gradient steps. MAML is a versatile method since it impacts only the training procedure. It aligns with the core goal of AP1.1 to leverage training methods for improving neural networks performance. Any gradient descent algorithm is compatible with MAML as long as the problem exhibits a subdivision into tasks.

Optimization for fast adaptation as proposed in MAML is not the only approach to meta-learning. An alternative is to use an intermediate representation that captures the inter-task properties of the problem. In *metric based* meta-learning [4, 6], for instance, the input data is mapped to an embedding where samples from the same class are located close together, while samples from different classes are located far apart. This embedding is trained during the meta-learning stage. During the adaptation stage, a simple task-specific proximity-based classifier is trained that classifies the embeddings based on their distances from each other.

Rather than computing an embedding based on the metric, the intermediate representation can also consist of features that are then reweighted during the adaptation stage [7].

Classical Object Detection

In *You Only Look Once* (YOLO) [8, 9], the core of the object detector is a convolutional neural net (CNN) for which the input image is segmented into $S \times S$ cells of equal size. For each cell, YOLO tries to detect whether it belongs to one out of B bounding boxes and to which one of C predefined classes the object in each box belongs. The output of the CNN is a three-dimensional tensor with size $S \times S \times (5B + C)$ so that each cell gets assigned a subset of the B bounding boxes and the C object classes. It is trained in an end-to-end manner, with an objective function that penalizes the location of the bounding boxes and the object class probabilities for each cell. Further one-stage detectors include Single Shot Multibox Detector (SSD) [10], Fully Convolutional One-stage object detection (FCOS) [11], and RetinaNet [12].

In contrast to one-stage detectors, which estimate objectiveness and class labels on a dense grid, two-stage detectors rely on region proposals in order to reduce the number of potential object locations. Most prominent approaches are FasterRCNN [13] and MaskRCNN [14]. The first stage consists of a Region Proposal Network (RPN), that outputs multiple candidate region proposals and an objectness score. The second stage performs classification based on the feature maps and the candidate region proposals. Efficiency is improved via building the RPN such that it shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals.

In recent years, transformers [15] have played an increasingly important role in the design of object detectors. A prominent example is the *DEtection TRansformer* (DETR) [16] that interprets object detection as a set prediction problem. DETR uses the *Hungarian* algorithm to compute the cost of a set of predicted bounding boxes. The DETR architecture is composed of four components. The input image is fed to the *backbone* CNN that creates a lower-resolution activation map. This activation map is then flattened and processed by a transformer encoder-decoder block that yields N embeddings, where N is the number of boxes to be predicted. Finally, these embeddings serve as the input to prediction networks that compute the bounding box coordinates and the predicted class label from them. It is worth pointing out that a bounding box can contain no object at all in which case it gets assigned an *empty* class label.

Meta-learning in Object Detection

The performance of an object detector is to a large degree determined by its implicit object classification capabilities. However, while the object classes can vary from one deployment scenario to another, the capability of detecting the presence of *any* object is the requirement for essentially all possible object detection scenarios. This indicates that meta-learning can be potentially exploited to create a detection framework with the capability to adapt to novel object classes.

One-stage detectors are generally preferred over frameworks based on region proposals since unseen object classes tend to cause deterioration of the region proposal quality [17]. Many meta-detectors are thus based on the concept of *feature reweighting*. This is for instance the case in Meta-YOLO [18]. To this end, the authors propose a framework composed of a *feature extractor* and a *reweighting module*. The feature learner is based on the detection architecture in YOLO, which means that each cell in the input image gets assigned a vector containing m *feature maps*. The reweighting module generates for each input image a vector of weights that are applied to the feature maps in order to calculate the bounding box and the predicted object class learned in the adaptation stage.

Meta-DETR [17] also follows a reweighting-based approach, but builds upon the transformer architecture of DETR. Input images are first mapped to task-specific class encodings and processed by the *correlational aggregation module*. The aim of this module is to leverage correlations between different classes. These

features serve then as an input to a class-agnostic transformer encoder-decoder architecture that predicts the respective object locations.

Other object detectors based on meta-learning include Meta-RCNN [19] and Meta-Det [20].

Meta-learning in Reinforcement Learning

Meta-learning has been also successfully applied in Reinforcement Learning (RL), most notably in the context of motion planning in dynamic environments. The authors in [5] give some examples where meta-objectives concerning the pursued goals, the received rewards, and the considered environments are incorporated into RL frameworks.

Besides in the original paper on MAML [1], meta-learned policies are also explored in [21]. A trajectory planning application is presented in [22], where the authors combine meta-learning with RL to develop a strategy for decision making for lane change maneuvers on highways. Again, a variant of MAML is used to improve adaptation speed for new scenarios, e.g. traffic density, road geometries, or traffic environments. The nearly same goals, however from a more general non-automotive perspective, are also investigated in [23].

2.2.3 Concepts

As already outlined in the introduction, fortiss aims at incorporating prior knowledge in the training process via training data organization and augmentation. The meta-learning paradigm of learning to learn allows utilizing the knowledge components put into data generation with the aim of (1) generalization to underrepresented tasks based on only a few data samples (2) reduction in training/adaptation time due to constraint search spaces.

Knowledge Components

In the following, we outline the targeted knowledge components. Since the focus of our work is on robust pedestrian detection, most of the considered knowledge components refer to the behavior of pedestrians. As mentioned in the preceding section, meta-learning can be straightforwardly applied to planning problems. Analogous to the components used to advance perception, planning problems also benefit from data organization and augmentation. In this context, rather than object detectors, the meta-learned model represents a policy, that can be quickly adapted to new environments and goals.

World knowledge

W1 Pedestrians are often occluded by other traffic participants, for instance parked vehicles, traffic signs or vegetation. The view angle and the mounting position of the camera as well as sizes of cars and obstacles lead to distinct occlusion patterns and occlusion ratios which are encountered in

urban driving scenarios [24]. Knowledge about the occurrence of perturbations can be used to generate a distribution of tasks that take these variations into account. Meanwhile, the same observations are beneficial to design augmentation strategies that reflect observed occlusion patterns.

W2 The behavior of pedestrians in certain scenarios is connected to forthcoming actions of the same individuals. People interacting with the driver or simply observing the traffic are more likely to cross the road and therefore pose an increased safety risk [25, 26]. This observation is again useful to organize the data and to encourage object detectors to focus on these rarely observed patterns. From a classification perspective, not only single objects but the appearance of the whole scene, i.e., the combination of traffic participants, traffic signs and environmental conditions like crosswalks and intersections could possibly point to upcoming critical scenarios.

Physical knowledge

P1 Detecting small size pedestrians is an important prerequisite in order to perform safe and comfortable driving maneuvers. Depending on the current velocity of the ego-vehicle, and the intrinsic parameters of the camera used, the size of pedestrians at a certain distance is calculated. The size is associated with action options relevant for brake or evasion maneuvers. Again, this domain knowledge can be used in both scenarios, i.e., defining tasks distributions that reflect this property as well as scaling and zooming of objects as a promising augmentation strategy.

P2 Many augmentation strategies applied in image-based object detection or classification consist of various image transformations, e.g. rotation, masking, and shuffling of pixels, just to name a few. In contrast to generic object detection, traffic participants usually exhibit much more specific variations due to physical constraints. As an example, pedestrians are not flipped upside down, or rotated to a larger extend. We use this type of knowledge to narrow down the search space of automatically determined augmentation policies.

P3 The appearance and location of objects in subsequent frames can be utilized in order to predict potential occlusions and to narrow down the search space for object detection. This prior knowledge is especially useful if one considers the adaptation stage where the learner has to quickly adapt to new environments and appearances, e.g. in a tracking scenario.

Extracted knowledge/concepts

E1 As another type of knowledge source, we aim to examine the possibility of using knowledge extracted from the model itself. A promising source are attention maps that reveal information about the relation between object parts and channel features. In the literature of pedestrian detection

and tracking, attention mechanisms have been frequently used in order to guide the detection process [27, 28, 29, 30]. Furthermore, meta-learning pedestrian detection often involves feature re-weighting, i.e., the weighting of channel features according to the underlying class [20]. There could be potential with regard to the adaptation/updating of specific channel features, which could possibly further reduce the training and adaptation time. The feasibility must be further examined in the course of the next project increment.

Occluded Pedestrian Detection (UC1)

Robust pedestrian detection is key to reacting in an adequate way, from gentle and comfortable braking maneuvers to emergency brakes to avoid collisions. We aim at early detection of traffic participants, even in highly occluded scenarios.

Robust detection of traffic participants

Meta-learning object detectors such as Meta-YOLO, Meta-RCNN [19], Meta-Detr [17] or MetaDet [20] serve as baseline methods. These frameworks are designed for generic object detection in still images and require certain adaptations for the pedestrian detection use case.

In particular, we need to adapt the training to the automotive use case. The intended approach is to use existing datasets for autonomous driving, e.g. BDD100k [31], Cityscapes [32], Citypersons [24] or JAAD [33], and organize the data to handle few-shot object detection tasks. This is done following the knowledge components **W2** and **P1**. During the meta-learning stage, base classes are treated as few-shot instances such that the algorithm learns how to deal with only a limited number of samples. In the adaptation stage, the detector adapts to underrepresented classes. Additionally, it can be helpful to exploit spatiotemporal dependencies, as the framework will be likely applied to video data.

Since the application scenario is already restricted to the case of pedestrian detection, adaptation should be easier to carry out than in the more general object detection setting. However, under certain circumstances, it may turn out that meta-learning is not necessary, especially when the behavior and appearance of pedestrians do not change significantly from one task to another.

Pedestrian Detection under severe occlusions

Object detection algorithms are prone to drop of performances when dealing with severe occlusions. The main reason is that Faster-RCNN [13], YOLO [8], and even DeTR [16] rely on bounding boxes labels. Hence, their inductive bias is to assume complete perception of the object in most images. We focus on mitigating this side effect by data augmentation.

We follow a systematic strategy and apply Differentiable Automatic Data Augmentation (DADA) [3]. It investigates a large set of transformations with

defined ranges. On top of included transformations, we focus on cutout that mimics the occlusion problem. It masks a square part of an image in black. The augmentation policy is tested on existing autonomous driving datasets. The success criterium is the detection precision under severe occlusions. In particular, we give higher importance to pedestrians in the far-field. We use the knowledge components described in **W1** and **P1, P2** in order to constrain the set of augmentation policies. The size of cars and obstacles (**W1**) as well as the size of pedestrians (**P1**) are good proxies for the parameter of the shearing transformation. The soundness of the augmentation ranges are verified by **P2**, in particular the rotation and the shear.

We expect that adapting data augmentation to driving scenario constraints increases the detection performance in varying environments. Data augmentation procedures are time and energy-consuming. Yet, a priori knowledge reduces training and exploration time.

Scene classification

Meta-learning has been extensively used in few-shot learning problems. In the aforementioned concepts, we focused on few-shot detection, i.e., to predict the bounding boxes and class labels of *individual* objects. According to the knowledge component outlined in **W2**, the interaction of traffic participants (pedestrian interacting with driver) in combination with the observed traffic situation (pedestrian approaching/crossing the street) can be related to specific criticality levels. The extreme imbalance in the training data might require consideration as a few-shot problem that could be tackled via meta-learning. In particular, this problem can be modeled as a few-shot classification problem. In this context, a special variant of algorithms could be particularly suitable here. Prototypical Networks [2] and/or matching networks [4, 6] are few-shot classification methods involving metric learning. The advantage of these methods is that they are usually fully amortized, i.e., they do not need to be fine-tuned in an adaptation stage which could be very beneficial with regard to the demonstration. After training the network, the classification decision can be simply done based on distance comparisons to the learned prototypes. In this way, an assessment of criticality may also be achieved.

We plan to investigate the potential of the JAAD dataset [26, 25] since this source contains several real-world sequences where pedestrians are crossing the street in varying environments and with different interactions.

Occluded Pedestrian Tracking (UC1)

A complementary research direction focuses on the tracking challenge. In this setting, object detection and assignment are performed on each frame of the video. It gives us the opportunity to take advantage of changes in appearance and location. The MAML framework is extended to the tracking task by wrapping around an object detector [34]. The few-shot learning tasks are defined by instance detection [35].

As the first step during inference, the initialization of the tracked object is crafted manually. This method shows its limitations in an autonomous driving scenario where multiple pedestrians need to be detected at different initial timesteps. We implement an alternative procedure where initialization happens at each timestep through multi-object detection. The outputs from the tracker and the detector are combined, by pooling for example, for accurate and up-to-date pedestrian detection. In order to limit the size of the neural network, pruning techniques discard past detector networks. In a first attempt, we skip online adaption that learns at each timestep of inference. Datasets are particularly adapted to the tracking task when they contain driving videos and multi-object bounding boxes. Examples are BDD100K, Cityscapes and Waymo. A performance baseline for the tracking algorithm is its object detection equivalent. More advanced results include automatization capabilities and multi-object handling. The prominent knowledge component is **P3** because the tracking task takes advantage of the temporal context.

Fast occluded pedestrian detection is facilitated by a time-structured algorithm where object dynamics have a representation in the neural network latent space. Real-time tracking imposes light architecture. Also, datasets are bigger than for object detection, hence training is longer.

Behaviour Planning (UC2)

The concept of meta-learning can be readily applied to planning problems tackled via Reinforcement Learning. Given some (simulated) trajectories, meta-learning can be used to improve the model and to adapt to new scenarios, environments, or tasks (e.g. traffic participants with varying behaviour in speed, acceleration or road conditions) that are underrepresented in the data. Analogous to the object detection use case, concepts derived for UC1 about general pipelines how to augment the data, or how to organize the data are thus transferable to UC2 scenarios that use RL instead of image-based object detectors. As already outlined in the baseline section, MAML [1] and its variants can be used to guide the learning process towards fast policy adaptation based only on a few representative samples. We plan to examine, how these concepts transfer to the automotive use case and if reliable trajectories can be determined.

Demonstration

In general, we plan to train and test our concepts on real world data. Existing datasets like BDD100k, Cityscapes/Citypersons, and JAAD serve as the backbone for prototyping and implementation. The applicability to simulated environments has to be investigated. Solely using simulated data requires generation of scenarios comparable to real world datasets. However, one of the strengths of meta-learned detection models is the adaptability to new environments. The extent to which this property enables adaptation to simulated data remains to be investigated.

Another concern regarding the provided AD stack is the possibility for online

adaption. Instance tracking benefits from an online adaptation stage in order to focus on the specific object. Provided that the framework allows online training, adaption can be achieved in only a few model updates, thus runtime is affected only in a minor extend. Alternatively, template based methods might be an option to avoid the need for online adaptation.

In the following, we briefly sketch the targeted functionality for multi-object tracking:

1. Determine object of interest that has to be tracked (either via predefined BB or motion detection), e.g. pedestrian or even any other object (stroller, segway, e-scooter). Bounding box is reprensted in green. Meta-learned object detector allows to quickly adapt to these classes.

2. After initialization use knowledge to determine augmentations (occlusion patterns in order to detect object even under heavy occlusions) and to determine search region (orange box) in next frame.

3. Track the object until it gets occluded; model should be robust against heavy occlusion.

4. If object is fully occluded, meta-learned object detector allows to quickly adapt to new object that occludes the object of interest. Track the new object and use this tracking information to determine search region of object of interest.

5. If object is partly visible again, re-track the object of interest (due to augmentation strategies, only small portion of the object suffices).

2.2.4 Summary

Within the scope of AP1.1, fortiss tackles the problem of object detection under varying conditions via a meta-learning approach. Different knowledge components are used to organize and augment the training data. The proposed meta-learned models are able to acquire shared features across different tasks, such that they can effortlessly adapt to new tasks. While the focus of our approaches is on the perception use case, the overall framework is very versatile and can be straightforwardly extended to other problem settings.

2.3 EFS - A Causal Approach to Knowledge Integration

In autonomous driving, we face a challenging environment for machine learning approaches due to the complex environment and the scarcity of examples of rare events. Therefore, it is beneficial to integrate prior knowledge into the training algorithms. Here, we focus on an approach based on causal models used to augment training data with correct vehicle dynamics.

2.3.1 Introduction

In this section, we describe our used knowledge sources, intended representation and proposed integration concept. As we participate both in TP1 (Knowledge Integration) and TP3 (Knowledge Conformity), our proposed concepts are closely intertwined with each other in the sense that the very same knowledge source is both integrated during an AI's training process and besides used to check the conformity of an AI's consumed input data as well as its predictions with regard

to that knowledge. A high-level overview of the interplay between our concepts (TP1&3) is depicted in Fig. 2.1.

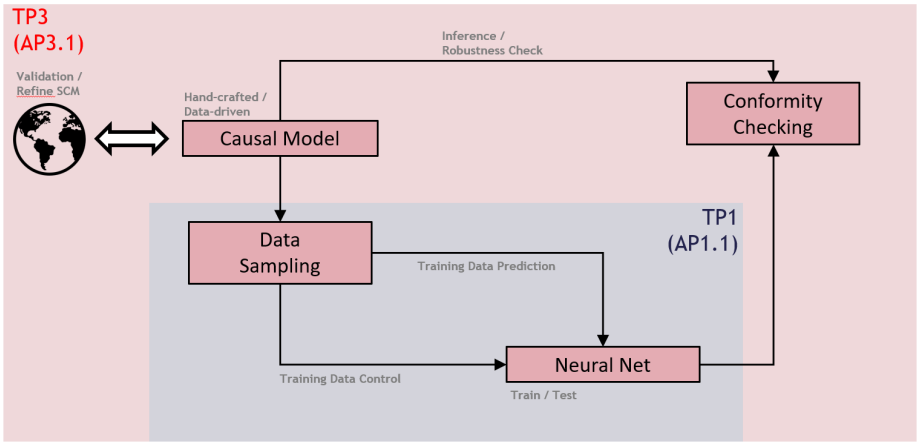


Figure 2.1: High-Level Overview of our Knowledge Integration (TP1) and Conformity (TP3) Concepts.

There, we see the components of the concept depicted. They rely on the causal model which will be created as part of TP3 based on prior knowledge and collected data and will be validated and refined over the duration of the project. Furthermore, as part of TP3, this causal model will be used to check the conformity of the outputs of the machine learning models with the knowledge. In this section, we focus on the knowledge integration (TP1) which is depicted in the blue area. There, we see that the causal model is used to sample data by actively modifying the mechanisms of the causal model for data augmentation. We are planning to generate this additional data for two types of models. A lane change predictor trained in a supervised fashion and a reinforcement learner able to control the ego vehicle.

As we will discuss in the following, a large part of our knowledge integration concept relies on the structural causal model (SCM) framework [36]. Based on this, we represent the physical-mathematical knowledge regarding vehicle dynamics in terms of an SCM. Knowledge formalized in this way is then used to generate cause-effect-aware new samples from previously given examples. Eventually, these additional data can be used to train machine learning models more sample efficiently by using the knowledge provided through the SCM.

Regarding AP1.1, we aim for the following contributions:

- Augmentation of training data with physical knowledge represented by SCM using the ladder of causation [36].
- Integration of physical knowledge represented by SCMs in reinforcement and supervised learning algorithms.

The remainder of this section is structured as follows: First, we highlight areas and approaches from the state-of-the-art that will be used as building blocks for research in this project in Sec. 2.3.2. Afterwards, we discuss our concept build around physical knowledge and its formal representation in terms of SCMs in Sec. 2.3.3. Finally, in Sec. 2.3.4, we summarize and conclude our proposed knowledge integration concept.

2.3.2 Baseline

The concept so far is related to three areas from the state-of-the-art.

- Structural causal models (SCM): Regarding the representation of knowledge we rely on structural causal models [36]. These models represent causal relationships between variables of a system and allow to reason within this system. The types of reasoning possible with this framework are described as the ladder of causation containing associations, interventions and counterfactuals which will be used within the project. These models have been successfully used for data augmentation in RL tasks [37]

- Reinforcement learning (RL): Regarding the control of the ego vehicle, we will rely on reinforcement learning. Currently, we have especially seen model-based approaches showing inspiring results on continuous control tasks [38], classical board games [39] and formerly model-free dominated atari [40]. Similar approaches have as well been applied to the autonomous driving domain [41].

- Continual Learning (CL): Regarding the integration of knowledge in machine learning models that are trained in a supervised fashion, we rely on continual learning [42]. We will investigate especially the applicability of gradient-based methods [43, 44, 45] since they yield the possibility of positive backward transfer.

2.3.3 Concepts

As briefly outlined before, we will use SCMs as primary knowledge representation. These will be used to augment the previously given real-world[1] and simulated[2] examples. Therefore, we give a small introduction in the possibilities using SCMs and the ladder of causation.

By definition, causal reasoning is the process of drawing conclusions from a causal model, similar to the way probability theory reasons about the outcomes of random experiments. However, since causal models are thought of as the data-generating process, they contain more information than probabilistic ones

[1] As a starting point we will rely on the HighD [46] and Automatum dataset [47]

[2] Since it is intractable to collect real-world on-policy data for reinforcement learning within this project, we will use simulation-based data. However, the developed concepts and methods can at every time be used with data collected in the real world.

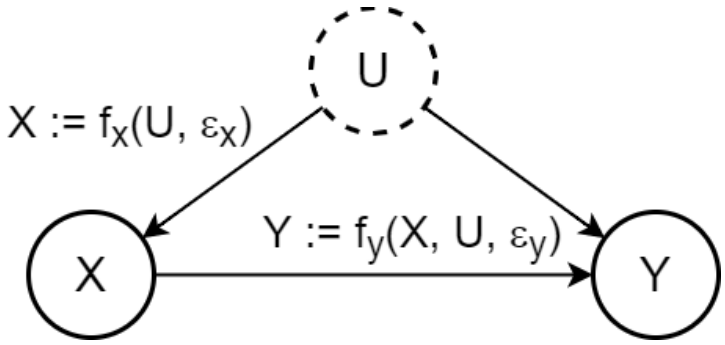


Figure 2.2: An SCM consists of both a set of causal mechanisms (f_X and f_Y) and a DAG modeling the flow of causation between variables (directed arrows). The shown SCM is composed of a cause (X) and its effect (Y), which are both confounded by a latent variable (U). Nodes associated with the exogenous noise variables ϵ_X and ϵ_Y are omitted for the sake of clarity.

and are thus more powerful, as they allow to analyze the effects of interventions or distribution changes on target variables [48].

Questions posed to the causal model are phrased as causal queries. They are at the heart of causal reasoning and can be differentiated into three levels of causation of increasing complexity [36]:

- *Associations* (correlations),
- *Interventions* (actively changing causal mechanisms),
- *Counterfactuals* (retrospective reasoning).

The just described *Causal Hierarchy* is often referred to as Pearl's ladder of causation or causal hierarchy (see Fig. 2.3):

On top of that, there are different types of causal reasoning [36]:

- *Prediction* (reasoning forward in time),
- *Abduction* (reasoning from evidence to explanation),
- *Transduction* (reasoning through common causes),
- *Induction* (from experience to causal knowledge).

One of the strengths of causal reasoning is its capability to assess the causal effect of one variable on another from observational data alone, without the need of collecting experimental data.

In the following, we refer to the SCM-representation of our vehicle dynamics model as the *Vehicle-SCM*. A high-level version of such a Vehicle-SCM is depicted in Fig. 2.4. There, individual vehicle-state-related variables are subsumed by

Level (Symbol)	Typical Activity	Typical Questions	Examples
1. Association $P(y\|x)$	Seeing	What is? How would seeing X change my belief inY?	What does a symptom tell me about a disease? What does a survey tell us about the election results?
2. Intervention $P(y\|do(x), z)$	Doing, Intervening	What if? What if I do X?	What if I take aspirin, will my headache be cured? What if we ban cigarettes?
3. Counterfactuals $P(y_x\|x', y')$	Imagining, Retrospection	Why? Was it X that caused Y? What if I had acted differently?	Was it the aspirin that stopped my headache? Would Kennedy be alive had Oswald not shot him? What if I had not been smoking the past two years?

Figure 2.3: The *Causal Hierarchy*. Questions at level i can be answered only if information from level i or higher is available [49].

the s_t variable. The same argument applies to various, independent action variables a_t. In the Vehicle-SCM, the current state and action are modeled as parental nodes both pointing in yet another node representing the vehicle's next state, as the latter causally emerges from the former ones. The way the next state's value $s_{t+\Delta t}$ results from the actual values of its parental nodes (s_t, a_t) is specified by a function and is distinctive of the chosen vehicle dynamics model M. Interestingly, vehicle trajectories can be conceived of as being generated by repetitive application of the Vehicle-SCM as indicated in Fig. 2.4.

During the project, we plan to use the ladder of causation to generate additional training data. This will be done with two types of machine learning algorithms. The first one is a lane change predictor with the purpose to predict if exo vehicles will switch lanes in the near future. The second type are model-based RL algorithms. For both types of algorithms the previously collected data will be processed with the SCM to generate additional data that helps to increase the robustness of the algorithm. This process can be applied iteratively by adding newly generated data for specific scenarios and utilize continual learning to avoid catastrophic forgetting. The augmentations can have different forms and we want to highlight two exemplary use cases here: they can be small changes to the longitudinal and lateral acceleration of the exo vehicles to enrich the data. Alternatively, they can be changes in the behavior of vehicles in front of the ego to a hazard breaking allowing the algorithm to address their scarcity in real data.

In the course of this project's lifespan, we will test and evaluate all our proposed concepts on Use Case 2 (Complex Lane Change). In particular, we will focus on its following sub use cases (SUC):

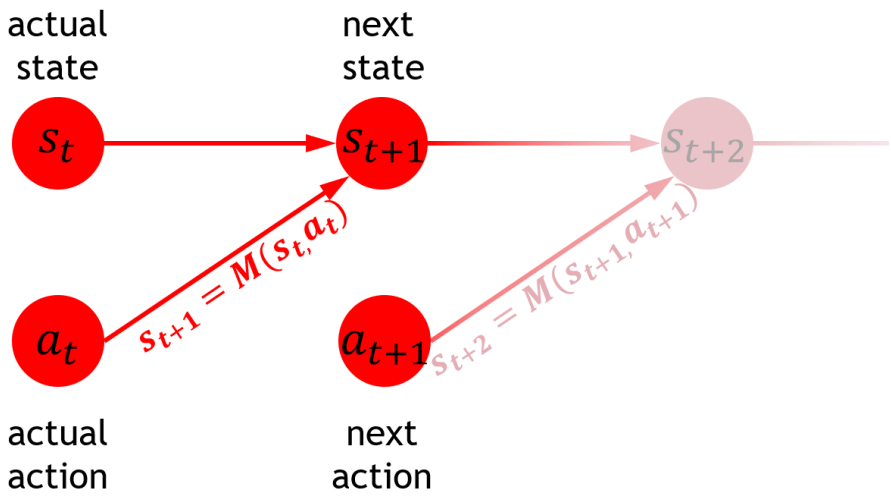


Figure 2.4: A high-level version of our Vehicle-SCM and its application in the process of vehicle trajectory generation. The vehicle's current state-action-pair (s_t, a_t) is mapped via a functional causal mechanism, the vehicle dynamics model $M(s_t, a_t)$, onto the vehicle's next state s_{t+1}, while obeying the laws of physics encoded in $M(s_t, a_t)$. Subsequent vehicle states are obtained by sequential application of the vehicle dynamics model.

- **SUC-2.4**: *Lane Change in Multi-Lane Road*
- **SUC-2.11**: *Lane Change Prediction of Exo-Vehicles in Multi-Lane Road*

Concretely, we assume two different AI-systems, each specialized in solving a single SUC:

- In **SUC-2.4**, an AI *controls* an ego-vehicle in such a way that it is able to weave its own way through a multi-lane road traffic by performing a lane change on its own and in such a way that it neither results in an accident with other traffic participants nor in undercutting safety-margins.
- In **SUC-2.11**, an AI *predicts* another traffic participant's intention for a lane change in the near future. For this purpose it is assumed that the AI makes its prediction conditioned on the consumption of spatiotemporal information on the past trajectories of the surrounding traffic participants. The AI-system shall then assign probabilities to all vehicle drivers in a scene for whether they will perform a *Lane Change Left*, *Lane Change Right* or *No Lane Change*.

2.3.4 Summary

Summarizing our concept for the knowledge integration we will rely on physical knowledge represented by SCMs. The knowledge will be integrated in common machine learning paradigms of supervised learning and reinforcement learning.

2.4 Valeo - Enhancing 3D Perception via Object Model Integration

Valeo aims to enhance the crucial task of 3D object detection for autonomous driving by leveraging object models as a-priori knowledge. An appropriate fitting score will be developed to utilize model matching in the loss function during training time. Various representations are possible for integration of those object models ranging from simple point clouds to more sophisticated gaussian mixture models. At the start the focus will be set to car objects and thus to use case 2.

2.4.1 Introduction

The task of 3D object detection is a crucial part in the pipeline of autonomous driving. The correct operation of the subsequent modules such as motion prediction, planning and steering depends on the quality of the perception results. Therefore, in this AP1.1 Valeo aims to increase the performance of 3D object detection with the integration of object models as a-priori knowledge into the training procedure. Since the focus is set on car objects, Valeo will apply this work mostly on use case 2. The key points of the concept are:

- Extension of 3D object detector by object models as a-priori knowledge.
- Comparison of predictions with object models via matching score.
- Inclusion of matching score into loss function during training time.
- Focus on lane change use case 2.

2.4.2 Baseline

For the planned contribution a supervised 3D object detection network is needed as baseline model. Therefore, Valeo decided for the PointPillars [50] network. This network is fast and lightweight and thus has the best chance to be successfully integrated into a demonstration car. It is a one stage voxelization approach. An illustration of its architecture can be seen in figure 2.5. The input pointcloud is first transformed into a discrete grid of pillars. This pillar grid is fed into the network. First, for each pillar a feature vector is created from all points inside via an MLP. This results in a feature map or pseudo image. Thus, the

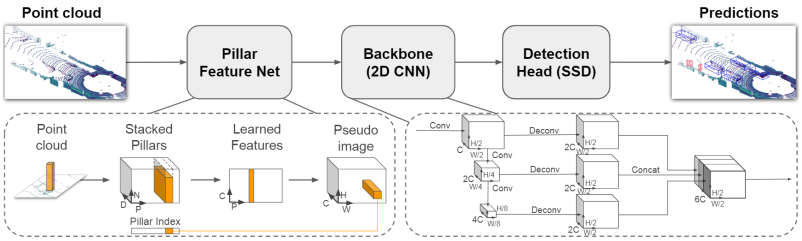


Figure 2.5: Sketch of PointPillars architecture taken from [50]. The point cloud is transformed into a grid of pillars and a pseudo image is created, thus further processing by two dimensional convolutions is possible.

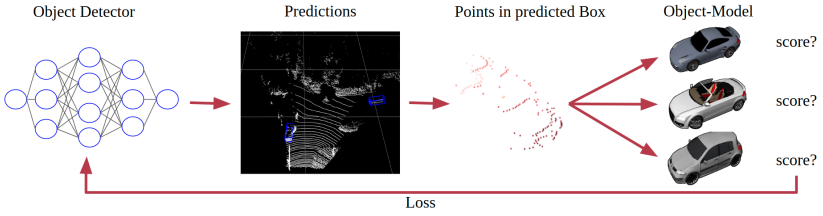


Figure 2.6: Illustration of planned contribution. The points inside the predicted bounding boxes of an 3d object detector are compared with a-priori object models. The matching score is added to the loss function during training time.

dimensionality is reduced and a series of two dimensional convolutions can be applied. Finally, a single shot detection head outputs the box predictions.

For an initial comparison of the planned contribution and extensions to the PointPillars network with the baseline method Valeo plans to use the public datasets Kitti [51] and nuScenes [52]. A similar approach that could be used for reference was not found in the current state of the art yet, thus the comparison with the baseline model must suffice. As metric the mean average precision will be utilized as it is common for the task 3d object detection.

2.4.3 Concepts

To increase the quality and the robustness of the box predictions, Valeo plans to incorporate a-priori knowledge into the training process. More specifically Valeo aims to utilize the shape of objects as additional knowledge for the network. Thus, a supervised 3D object detection network, that predicts the bounding boxes of objects on point cloud data, is used as a foundation. As already mentioned above Valeo plans to use PointPillars as such a foundation. As illustrated in figure 2.6 the object detector predicts the bounding boxes surrounding the objects of interest as narrow as possible. Valeo wants to use these predicted bounding boxes to cut out the points belonging to single objects and compare them to a

selection of appropriate object models. These object models form the a-priori knowledge. The fitting of a prediction and an object model is measured by an appropriate metric and results in a score, that measures how likely the points inside the prediction fit the shape of an object model. This score summed over all the predictions of one batch can then be added to the loss function during training time. Thus, the network learns to predict objects that fit the given object models, hopefully preventing false positive detections and resulting in tighter box predictions in general.

The amount of object models each prediction is compared with shall be kept as small as possible for the initial implementation and depends on the object model representation. For the object models several representations are possible. Valeo takes three options under consideration:

- mesh grid,
- dense point cloud,
- gaussian mixture model.

Dependent on the representation the similarity score can be formulated in different ways. In case of a discrete mesh grid a comparison via a mean square error is possible. If a point cloud is utilized the similarity could be measured by distance metrics. Here, most common are the chamfer distance and the earth mover distance. In case of a gaussian mixture model for representation the likelihood the predicted points belonging to the gaussian mixture density can be used. Which of these representations is best suited for the task is not known yet and remains a point to be determined in the future by appropriate experiments.

In the course of the project Valeo plans to restrict the object detection and the proposed idea of matching object models to cars only. Car objects are the most suitable for the initial proof of concept, since they are rigid and do not change their shape, in contrast to, for example, pedestrians. As a source for such car models Valeo considers two different variants. First, object models could be taken from the ShapeNet dataset [53], where CAD models for diverse objects, among others cars, are publicly available. The major disadvantage here is that these car models are not as realistic. For example, there are models with rear spoilers in the data set, which are rarely seen in actual road traffic. Valeo will use this option most likely for the mesh grid and point cloud representations. Then, for the initial implementation of the proposed concept only one average car model is necessary. In the further course of this work the amount of used car models can be increased. The other option is to extract object models from real world datasets like nuScenes or Kitti. This second option will be utilized in case of a gaussian mixture model representation, since this has to be learned from the LiDAR scans beforehand. Here, several object models will be necessary for the different relative view angles between sensor and object.

Since the focus will be set to car objects, Valeo aims to show the functionality of the proposed concept on use case 2. Here, a critical lane change shall be performed. Thus, a tight prediction of other cars is crucial for this use case. For

the demonstration of the proposed method the in section 2.4.2 mentioned real world dataset could be used to show an increase of the performance metric. The usage of simulated data is not as straight forward, since according data has to be generated not only for the demonstration but also for the training of the network. Otherwise the domain gap would cause a problem. This problematic of a domain gap might also occur if the proposed method is tested in an demonstration vehicle without appropriate training data.

2.4.4 Summary

In this AP1.1 Valeo aims to increase the performance of a 3D object detector, namely PointPillars, by utilizing an object model matching of the predictions with predefined object models into the loss function during training time. Which representation for the object models as well as the matching metric works best, remains an open question, that will be answered by experiments in the future. The focus is set to car objects and use case 2, since car objects suit best for an initial proof of concept and the tight prediction of cars is crucial for a successful lane change.

2.5 OFFIS - (Hierarchical) Reinforcement Learning with knowledge-infused reward shaping

We consider reinforcement learning (RL) which is enhanced with different types of knowledge. In a first paper, we studied the impact of a physics-guided RL approach for a car-following maneuver where we augment the reward function with a regularization term that penalizes any deviance of the distance of the ego-vehicle and the leading vehicle from the optimal, velocity- and acceleration-dependent longitudinal distance to the leading vehicle.
We are currently working on a second paper where we have urban scenarios and again consider a car-following maneuver but followed by an overtaking maneuver, enhanced with incoming cars on the left lane and pedestrian detection with potential occlusion. We propose knowledge-infused RL in order to encourage the agent to respect several (soft) constraints arising from physical knowledge, traffic rules and world knowledge.
We already outlined a multitask reinforcement learning approach that partitions an overall task into several sub-tasks and that learns sub-task-specific sub-policies. For accomplishing the overall task, an option policy has to learn to execute the respective sub-policies in the correct order and for the correct amount of time. Again, we infuse each single sub-policy with sub-task-specific knowledge. We also aim at letting the agent learn the sub-tasks automatically. We plan to apply our approach to autonomous driving scenarios concerning occlusion and

overtaking as well as relevant edge cases in the sense of intentional rule violations.

2.5.1 Introduction

We propose to infuse RL with knowledge (e.g., [54]) by reward shaping (e.g. [55]), i.e., enhancing the reward function with penalty/regularization terms that represent constraints arising from different types of knowledge. The main motivation for considering RL models is that they do not require real data which are yet hardly available but produce simulated data themselves during the training process. Additionally, deep RL models where the policy that maps the states onto the action space is represented by a deep neural network are promising candidates for providing sophisticated decisions. The main reason is that they compute deep state abstractions so that features can be represented in a way that exceeds the possibilities of hand-crafted, rule-based algorithms.

We already showed that a physics-guided approach where the agent is encouraged to keep an optimal safety distance increases the safety of autonomous driving in contrast to a simple agent that is solely trained w.r.t. a collision penalty. The extensions on which we are currently working are expected to be able to cope with rather complex traffic situations, including overtaking, occlusion or intentional rule violation (covering all three use cases) in a more reliable fashion than standard RL approaches or rule-based policies, especially in challenging situations. The planned approach to partition maneuvers into sub-maneuvers which are handled by individual sub-policies in a multitask RL approach (cf. [56], [57]) should improve the safety further. This is because each sub-policy will be trained to become an expert for the respective sub-task. In contrast, an overall policy which may would be very hard to train such that it works well for each sub-maneuver where one additionally had the issue of contradicting side constraints. We also consider different road conditions where additional physical knowledge concerning the vehicle dynamics is integrated.

Main contributions/expected improvements:

- Train the RL agent to respect knowledge like keeping an optimal safety distance, smooth lane changing, deciding whether to overtake or respecting potentially occluded pedestrians.
- Replace the training of an overall agent by training several sub-agents that have the potential to make reliable decisions on their respective sub-task.
- Learn an agent that decides which sub-policy to execute at which time.
- Achive successful driving behaviours not only on dry and undamaged roads but also with worse road conditions.
- Cover all Use Cases and combinations of them in complex traffic scenarios.

2.5.2 Baseline

We will not repeat RL in detail here but only briefly.

A RL problem is generally represented by an MDP (Markov decision problem, see e.g. [58]), i.e., a tuple $(\mathcal{S}, \mathcal{A}, T, r, \gamma)$ where $\mathcal{S}$ is the state space containing all relevant information and where $\mathcal{A}$ is the action space. One assumes either a deterministic transition model $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, i.e., performing action a in state s leads to state $T(s, a) = s'$, or a probabilistic transition model $\tilde{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_{\geq 0}$, i.e., performing action a in state s leads to state s' with the probability or, for continuous states, with the density $\tilde{T}(s, a, s')$. The reward function is represented by r and $\gamma \in]0, 1]$ is an optional discount factor that discounts future rewards. The goal is to learn a policy, parameterized by some parameter ϕ, which is either deterministic, i.e., $\pi_\phi : \mathcal{S} \to \mathcal{A}$, indicating that in state s, one performs action $\pi_\phi(s) = a$, or probabilistic, i.e., $\tilde{\pi}_\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}_{\geq 0}$, so the probability/density for choosing action a in state s is $\tilde{\pi}_\phi(s, a)$. For techniques how to find an optimal policy, see e.g. [59]. We mainly focus on soft actor-critic RL, see [60], since it allows for faster training.

Multitask RL in contrast partitions the overall MDP problem into sub-problems [61]. One advantage is that both the value functions as well as the sub-policies can be shared if considering a task consisting of several sub-tasks. Learned feature representations are also easier for the sub-problems and allow the combined value function to be representable as a sum of these individual terms [61]. Options [62], [63] are actions over a period of time which here define the current sub-task (see also [56]). Then a sub-policy is trained for each sub-task and an overall option policy decides which sub-policy becomes active at which time step. The concatenation of these sub-policies according to the partition of the overall task into these sub-tasks forms a hierarchical policy. There are purely unsupervised strategies [63], [64], but also "minimally supervised" strategies [57]. In the latter, one assumes that the structure in terms of sub-tasks (and therefore, the structure of the hierarchical policy in terms of a sub-policy sequence) is given but the time steps in which one switches from the current to the subsequent sub-task and sub-policy have to be learned.

2.5.3 Concepts

1.) As for the approach we already finalized, we considered the following formula from [65] which relates the desired acceleration with the desired velocity v_0, acceleration constraints and a minimum jam-avoiding distance s^* by

$$acceleration = a_m \left[1 - \left(\frac{v}{v_0} \right)^4 - \left(\frac{s^*(v, \Delta v)}{s} \right)^2 \right] \tag{2.1}$$

for the current distance s between the ego-agent and the leading agent, the current velocity v of the ego-agent and the velocity difference Δv of the ego and

the leading agent.

[65] then computed the minimum jam-avoiding distance

$$s^*(v, \Delta v) = s_0 + \max\left(0, vT + \frac{v\Delta v}{2\sqrt{a_m b}}\right) \tag{2.2}$$

where s_0 is a static minimum distance for small velocities, T is the save time headway, b is the desired deceleration a_m the maximum acceleration. This target distance entered in the penalty term

$$r = -\frac{|s - \mathtt{ts}|}{\mathtt{ts}} - \frac{|s - \mathtt{ts}|}{2s} \tag{2.3}$$

for the reward function with $\mathtt{ts} = s_0 + \max(0, vT)$ being the target separation at the given speed.

The states consist of the distance of the ego and the leading vehicle, their speed difference and the speed of the ego-agent. The action space is $[-1, 1]$ where -1 represents maximum deceleration and 1 maximum acceleration. The policy is stochastic and parameterized by a neural network with a single hidden layer with 500 neurons. As for the transition model, we, due to restricting ourselves to straight roads, used Euler's method for forward simulation of both the distances and the velocities based on the current velocity resp. the acceleration which the policy selected in the current time step. The reward is $-r_0$ if a collision occurs for the reference RL model where a suitable r_0 is derived via parameter search. The reward is

$$r^{PG} = -r_0 + r \tag{2.4}$$

for the physics-guided RL model. As for training the policy, we applied the SAC algorithm which models the $Q-$function by a neural network with again one hidden layer with 500 neurons. We stopped training after 1 million iterations.

We trained the model for different deceleration patterns of the leading agent (including abrupt braking which corresponds to the sudden appearance of occluded pedestrians in UC 1), performed the simulations again for different penalization values for collisions (and also considered less sophisticated reward terms for the physics-guided RL for comparison) and repeated all simulations for perturbed perception where the inputs are multiplied with random multipliers from a uniform distribution on $[0.9, 1.1]$, leading to a total of 128 different configurations. The simulations have been done in Python.[3]

A clear limitation of this approach is the restriction to a simple car-following scenario on straight roads. However, this paper was intended to be a proof of concept, therefore we are working on the mentioned extensions.

[3] See `https://www.mdpi.com/1996-1073/14/22/7572` for more details.

2.) As for the urban overtaking scenario, we aim at extending the paper above by including yet to be selected penalty terms and by training the agent so that it performs whole overtaking maneuvers, i.e., approaching the leading vehicle, performing the lane change, passing the other vehicle and changing back to the right lane. We also integrate additional features like occluded pedestrians and incoming vehicles on the left lane. As for the incoming vehicles, the ego-agent should learn when the safe headway is too small so that it waits until the incoming vehicle has passed before starting to overtake.

The simulations are to be done in CARLA, so we are currently working at both defining the penalty terms for the RL models and on integrating pedestrian recognition in CARLA. More precisely, we aim at integrating visual data in the form of 100×100 images for the ego-agent from which the states have to be perceived where the state space consists of the longitudinal distance between the vehicles, the relative velocity between the vehicles, the velocity of the agent vehicle, the lateral distance of the vehicles and the angle of rotation of the agent vehicle. The action space consists of three float values indicating acceleration, left or right turn.

The reward function will consist at least of a stage reward with encourages an evolvement in the stages of the scenario (beginning, approaching, lane change, overtaking, again lane change), a collision penalty and a positional reward. The stages are integrated into the state by a unit vector, i.e., e_j represents that the agent is in stage j. As for the knowledge-infused agent, the reward is penalized additionally by an acceleration term which becomes negative if acceleration is necessary but if the agent does not accelerate. Other penalty terms include steering penalties if the wrong steering is executed or if steering is executed if no steering is desired. These penalty terms get fixed values. Another potential penalty term will encourage smooth steering and penalize different steering maneuvers among adjacent time steps.

3.) As for the hierarchical RL approach, we define a set of tasks (options), based on [56], but aim at extending it, at least with different road conditions (e.g. icy roads), so that one essentially requires to train one agent for each combination of options and road conditions.

We intend to use the following state space $\mathcal{S}$ resp. shared action space $\mathcal{A}$ for the sub-policies. As for the state space, we require (maybe a subset of) the following information: **i)** The roads/drivable area; **ii)** the speed limit (for each coordinate of the drivable area or cell if it is represented as a grid); **iii)** stop sign locations, traffic lights, other relevant traffic signs (including no-passing zones); **iv)** the types, positions, velocities and orientations of the other traffic participants; **v)** the details of the ego-vehicle (width, length, maximum speed, maximum acceleration/deceleration, maybe even tire details or weight); **vi)** allowed orientation for each lane/passage; **vii)** road conditions; **viii)** weather

conditions (including lightning conditions); **ix)** Lengths and widths of other vehicles; **x)** of course, the destination (and also required waypoints/way-areas; for example, it would be unnecessary to overtake shortly before reaching the destination and it would be somewhat mistaken to overtake shortly before the exit of a highway so that it would be missed). The state space $\mathcal{S}$ is therefore given by the cartesian product of the possible ranges of the individual variables. Optionally, we may learn an abstraction $\psi : \mathcal{S} \to \mathcal{F}$ of the state space into some feature set $\mathcal{F}$. As for the action space, we only consider the relevant actions that the ego-vehicle can perform in our scenarios, i.e., accelerating, decelerating and steering. Actions like blinking etc. are ignored for now. Therefore, we have a continuous action space $\mathcal{A} = [a_{min}, a_{max}] \times [0, 2\pi] \times \{-1, 1\}$ where the first component is the acceleration interval given by the maximal possible acceleration a_{max} and the negative maximal possible deceleration a_{min} while the second component defines the steering degrees. The third component defines the driving orientation (backwards, forward).

We suggest the following penalty terms:

- Jam-avoiding distance for dry roads
- Jam-avoiding distance for icy/snowy/overflood roads
- Distance to other vehicles when waiting in a queue or stopping
- Off-road loss [66]
- Yaw loss [67]
- Speed limit penalty
- Penalties for non-comfortable decelerations and accelerations (however, in the case of an emergency brake, the safety goal clearly beats the comfortability goal so the penalty here has to be rather small)
- Penalty for crossing a red traffic light (may even be infinite, this should not prevent the agent from moving since this forbidden maneuver can easily be avoided by stopping)
- Penalty for crossing a stop sign without stopping
- Penalty for a too low lateral safety distance when overtaking
- A very high collision penalty

These penalty terms are not used for all sub-agents but only for those for which they make sense. For example, the agent that learns the lane change may not require the yaw loss since lane changing requires to modify the steering angle. Similarly as in [56], there will be extra rewards when accomplishing an intermediate goal, i.e., when finishing a sub-task.

The fully unsupervised MCTS approach proposed in [56] is too ambitious for starting with. The term "fully unsupervised" indicates that the agent does not even get information about the current sub-task (e.g., lane changing, braking etc.) but has to find the suitable task by itself. Therefore, we intend to start with the partially observed approach as in [57]. In this approach, we provide the agent with the sketches during training, i.e., the agent knows the ordering of the sub-tasks, but it has to learn the time steps when going over to the next sub-task. The sub-policies can be trained individually using information from a shared scenario as suggested in [57]. More precisely, let π_{θ_1},...,π_{θ_K} be the sub-policies, each parametrized with some θ_k. Then, having sampled a trajectory $H^{(i)}$, the overall parameter θ for the concatenated policy $\Pi^{(i)}$ corresponding to $H^{(i)}$ will be updated using a gradient step as in Eq. 2 of [57].

As for the option policy $\rho : \mathcal{S} \to \mathcal{O} \times \mathcal{R}$ for the set $\mathcal{O}$ of sub-tasks and the set $\mathcal{R}$ of road conditions, we plan to use curriculum learning similarly to [57] in order to sample suitable sketches. This certainly can be done for length-one sketches, however, we have to ensure that the sketch sequences are meaningful. Maybe it is possible to extract sketches from valid ego-trajectories which are simulated according to knowledge constraints if such data are available.

We propose the following selection of scenarios and will implement at least a proper subset of them:

1. **Overtaking on dry roads:** Similarly as in [56], but including more challenging situations like
   - A vehicle in front of the ego-vehicle on the overtaking lane which is only insignificantly faster than the overtaken vehicle (example: trucks on a highway); the goal is to learn to respect the safety distance to the leading vehicle during overtaking.
   - A wide vehicle which partially covers the left lane (example: a tractor on a country road); the goal is to learn a suitable lateral distance as well as a decision whether there is enough space on the left lane to execute the overtaking maneuver.
   - Overtaking a cyclist in town while there are traffic participants on the opposite lane; the goal is to learn to know when overtaking is possible (if there is just another cyclist on the opposite lane) and when not (if there is for example a sufficiently wide car so that no suitable lateral safety distance could be respected).
   - Standard overtaking in a town or on a country road but with no passing areas; the goal is to learn that overtaking is not possible here, although a pure lifeness reward would be higher if it did.
2. **Overtaking with challenging road conditions:** Essentially the same sub-scenarios as in 1.), but the goal is to learn when it is better not to overtake.

3. **Rule exception:** Learn the necessity to violate the traffic rules (based on UC 3):
    - Simple setting where there is an obstacle on the road; the goal is to learn to react in time before hitting it/needing to drive backwards in order to change the lane.
    - Stopped vehicle that takes the role of an obstacle in a no passing area (learn to execute the rule exception), followed by a slowly moving vehicle (learn not to overtake here).
    - Extreme (but realistic) case: Very slowly moving vehicle in a no passing zone, e.g., a caterpillar, tractor, road sweeper, tank etc. (what do we expect here from the AI?).
4. **Rule exception with occlusion:** Assume that there is a large obstacle blocking the road so that there is a considerable occluded area. When avoiding this obstacle, the agent should learn to be cautious since there may be a pedestrian/cyclist crossing the road behind the obstacle.
5. **Rule exception with challenging road conditions:** While one may learn that under certain road conditions, one should not overtake, the case is different here since there is no other option than to change the lane and "overtake" the obstacle. This maneuver has to be accurately learned even for icy roads.
6. **Overtaking with occlusion:**
    - Overtaking scenario in a town where a pedestrian/cyclist may appear behind a large parking vehicle so that the overtaken agent may have to evade a collision themselves and performs a lane change.
    - Overtaking with a bend in front which is an exemplar for a large occluded area. The agent has to learn to measure if there is a sufficiently long safe headway including the physical knowledge that during overtaking, a vehicle on the opposite lane may appear at the curve so that the decrease of the safe headway can be computed. One may also include world knowledge that a rowdy who violates the speed limit can appear so that the safe headway may even decrease more.

The hierarchical approach has the potential to enable both active learning and set-based learning (AI verification) that OFFIS announced in the VHB. Set-based learning can clearly be applied by checking whether there exist states where the individual sub-policies take actions that violate any (knowledge) constraints. Similarly, one could apply it to the option policy which may be even more interesting, i.e., we check in a verification context in which situations it would produce wrong decisions concerning the sub-task and in a conformity context in which situations it would produce decisions that (while may indeed leading to high rewards) contradict human decisions, especially based on

world knowledge, for example overtaking shortly before reaching the destination.

As for active learning, one could add training loops that solely concern the option policy ρ. This is essentially a classificator which is trained in an unsupervised manner by MCTS in [56] resp. by standard policy learning as in [57] (essentially by learning how long to stay in the current sub-task). Therefore, it should also be possible to produce a set of (valid) states which are presented to the option policy in order to let it decide what to do. Then, active learning would apply where the option policy is very unsure about the decision (uncertainty sampling) while wrong decisions should be detected during the overall training process since the corresponding rewards would be very low. Although the agent makes many trials in RL, the sketch set is that large that it may receive a good reward but that the decision is still suspicious since it differs from human decisions. Of course, this requires a human in the loop which can provide the agent with the correct decision at that time.

The limitation of the hierarchical RL approach is that the design of the particular regularized reward functions and the regularization parameters may be difficult and that the sub-tasks have to be carefully defined. Since we consider deep RL, i.e., the policies are represented by deep neural networks, it can still be hard to understand the underlying decision making process. Also, the approach cannot cope with environment changes like the national delta in KI-DL where one switches between driving on the right resp. on the left lane for example.

The results will be demonstrated using CARLA.

2.5.4 Summary

We consider knowledge-infused RL approaches in which additional penalty terms appear in the reward function in order to encourage the agent to respect certain types of knowledge. A proof of concept where an optimal safety distance is to be kept has already been successfully completed.

We are currently working on a more sophisticated approach that can handle urban overtaking scenarios with incoming vehicles on the left lane and with potentially occluded pedestrians. We have already planned a hierarchical RL approach in which the overall task is divided into sub-tasks and accomplished by learning sub-task-specific sub-policies. With this strategy, we aim at overcoming the issue of contradicting constraints and in order to simplify the training for the sub-tasks. We expect that, once fully trained, the latter approach will be able to produce policies that can cope with complex traffic scenarios with difficult conditions.

A Python implementation of the proof of concept has already been realized. The subsequent approaches will be demonstrated in CARLA. Since we aim at

using reinforcement learning, we do not need real data. However, it is evidently necessary that the data that we simulate (already during training) are realistic in the sense that they at least do not contradict physical laws. This has to be guaranteed at several stages like the initial scenario configuration (for example, vehicles' positions cannot be just randomly generated since they may would stack onto the same position), during side constraints and penalty terms in the reward function (for example, speed limits should be realistic) or during the transition from one time step to another so that physical vehicle models have to be respected when determining the new position of a vehicle based on the previous position and the executed control action.

2.6 FZI-TKS: Knowledge infused Reinforcement Learning

In AP1.1, FZI TKS develops training methods for integrating additional inputs for rule-based control of training progress in the presence of multimodalities in neural network outputs. Here, we focus on the reward function of reinforcement learning as our interface.

2.6.1 Introduction

Here we present our first concept on the intergration of a priori knowledge in the training process of an AI. We want to include two types of knowledge:

- Rule Slack: How do human agents actually perform, provided with concrete traffic rules?
- Rule Engine: How can an agent derive appropriate actions presented with situations that include rule exceptions?

Introduction: Knowledge in Reinforcement Learning

To integrate additional input, such as knowledge, into the training of reinforcement learning systems, the reward function is a promising interaction point. General rule knowledge can be extracted from dataset analysis to be incorporated. To integrate rule knowledge based on the current situation, an interpretable representation of the environment is necessary. Chen et al. [68] show how a non-interpretable, vector-based latent space can be used to also output an interpretable mask on how the model perceives its environment next to the control commands, as shown in Figure 2.7. A graph-based latent space representation is also possible, as shown in [69].

In the following, we present two approaches to integrate knowledge into the reward function. Rule Slack is what we call typical human behavior, while strict rules apply to allow for more relaxed rules to handle rare situations better. A

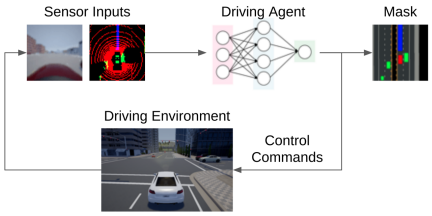


Figure 2.7: Interpretable Reinforcement Learning as proposed by Chen et al. [68]

rule engine is what we call a system that can identify conflicting rules and thus dynamically adapt the reward function based on a hierarchical setting.

2.6.2 Baseline

Rule Slack

Driving is restricted by various traffic rules, aiming to guide drivers maneuvering in a safe way. However, with corner-cases or dilemma situations, the driving environment appears highly complex. Depending on the situation, rules are not equally important or might even be contrary. Therefore, utilizing these generic rules e.g. in activation functions of an reinforcement agent, is short-sighted. A value of importance is inevitable for every rule.

The idea of assigning such values to traffic rules was developed in [70][71] in order to overcome dilemma situations and generate human-like behavior. A traffic rule is formalised as an signal temporal logic (STL) formula, which enables learning a satisfaction margin for any given rule. This margin or robustness slackness is calculated by observing expert behavior.
With their slackness they incorporated the human value of each STL and therefore enriched generic traffic rules. In combination with a model predictive control (MPL) they achieved significantly better results and received more human-like agents.

Rule Engine

The issue of conflicting rules in road traffic regulations ("Straßenverkehrsordnung" in Germany) is well known. The KI-WISSEN consortium loosely defines *rule exceptions* as follows:

> "A provision is a rule exception if it **prevents or modifies the application of another provision** in a specific individual case due to the occurrence of an unusual circumstance or the absence of an otherwise usual circumstance. The usualness is determined by the statistical frequency of the circumstance in comparison to the

frequency of the factual situation which is regulated by the provision from which the exception is intended to deviate. As soon as the statistical frequency is worthy of discussion, it is no longer a rule exception."

Thus, a hierarchical order of rules is implied by rule exceptions. Censi et al. [72] have shown a framework that can utilize such constructs as shown in Figure 2.8 to incorporate them into classical planning components.

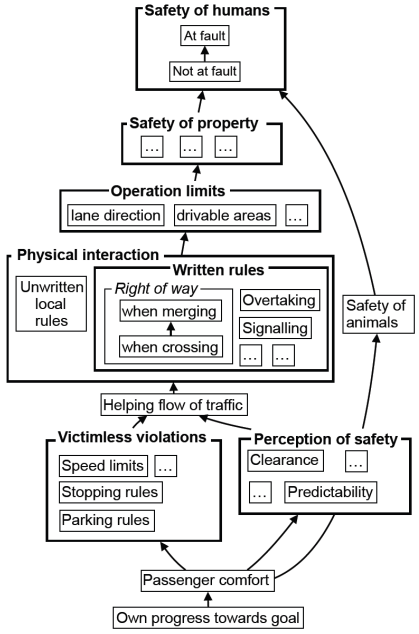


Figure 2.8: Rule graph as proposed by Censi et al. [72]

2.6.3 Concepts

Rule Slack

At hand we want to extract and learn the robustness slackness of three traffic rules:

- Distance keeping
- Speed limitation
- Lane keeping

Each slack is a percentage, that indicates how strongly a rule is followed among every driver in the dataset. Thus, we receive dynamic human interpretation of traffic rules instead of static rules. Later we utilize these slacks within a reinforcement learning agent, via its reward function. This should contribute knowledge to the RL agent and enforce a human-like driving style.

The scores are calculated from the *Waymo Motion Dataset*. It consists of 103,354 segments each containing 20 seconds from urban to suburban terrain, which were recorded on various locations, in six different cities among the US. Waymo offers detailed object lists with map data which enables the calculation of the slacks mentioned above. More details about each individual rule is featured below

1. Distance keeping is implemented similar to the three second rule [73]. The n'th drivers position P_0^n is multiplied by three times its velocity vector $\vec{v}_n$ to receive his future position P_1^n, which forms the minimum distance $t_n = (P_0^n, P_1^n)$. A violation occurs if there exists another driver P_0^k $(k \neq n)$ inside t_n. Then, $t_k = (P_0^n, P_0^k)$ denotes collision trajectory i.e., path to the closest car in between. Afterwards, we calculate the robustness slackness RS^{dist} by dividing the collision trajectory with the drivers minimum distance:

$$RS_n^{dist} = \frac{t_k}{t_n} \tag{2.5}$$

The equation ensures that $RS^{dist} \epsilon [0, 1]$ in every state of the scenario.

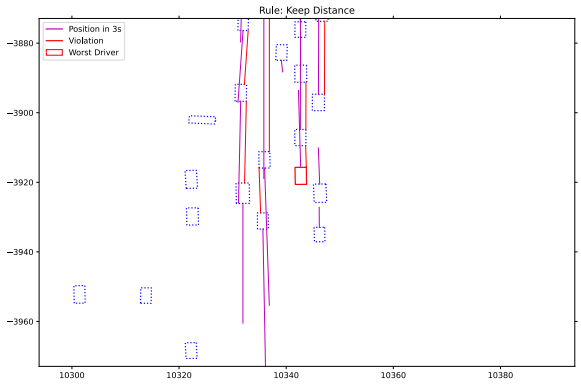


Figure 2.9: Visualization of rule calculation.

For the plotted scenario in Figure 2.9, the total slack among all drivers was ≈ 0.80063.

2. To extract the current speed limit for individual drivers we only take middle

lanes into count, since they solely contain limits. A lane l_n is assigned to every driver d_k, in respect to the minimum distance (euclidean distance). We filter out vehicles that exceed a certain distance threshold to avoid misclassification, and also omit vehicles below a certain speed threshold to avoid including parked or congested vehicles. Afterwards speedlimit slack RS^{speed} is calculated:

$$RS_k^{speed} = min(1, \frac{limit_n}{\vec{v}_k}) \tag{2.6}$$

Limit denotes the speed limit of lane n that is assigned to driver d_k whereas $\vec{v}_k$ is the velocity of driver k.

3. The implementation regarding lane keeping is still work in progress, details will follow.

These extractions of rule slack within the Waymo dataset can be applied to German traffic since the settings, including speed limits, are very similar. Thus, we can incorporate that knowledge into the reward function of an RL agent to achieve a more human-like behaviour, including implicit rule exceptions, instead of always strictly following the rules.

Settings. Waymo dataset contains a tremendous amount of data which complicates the above steps. Therefore, we calculate scores each second instead of using the whole dataset, which provides ten steps per second. Furthermore, parallelism is applied when operating on the speed limit slack to accelerate poor run-time. Nevertheless, those hyperparameter can easily be tuned by providing flags, when running these scripts.

Rule Engine

In close alignment with AP 1.2 the concept of AP 1.1 is to utilize Reinforcement Learning as the Machine Learning baseline methodology. To integrate additional knowledge into the training, we will utilize the reward function as an interface. To be able to interpret the environment to apply rules, a model shall be trained that is able to generate interpretable outputs of the environment, as shown in Figure 2.10.

The actor is supposed to suggest multiple trajectories each step. Combined with the interpretable environment representation, the rule engine should be able to interpret situations ("situational awareness") and evaluate each trajectory based on a small set of selected rules. Thus, the reward can be adjusted more dynamically compared to state-of-the-art models. One popular example is the crossing of solid lines, as shown in Figure 2.11

SotA RL agents typically get punished statically for divergence from the center line. Their reward functions typically

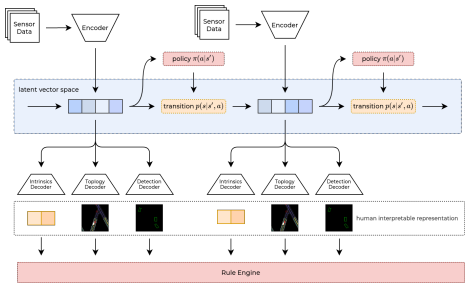
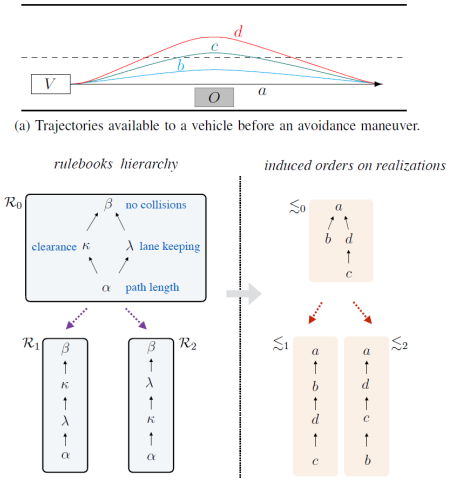


Figure 2.10: Proposed architecture to generate a scene representation that can be interpreted by a rule-engine.

2.6.4 Summary

In AP1.1, FZI-TKS proposes a strong extention of the often little regarded reward function in reinforcement learning for autonomous driving, with knowledge about driving behavior and conflicting traffic rules.



(a) Trajectories available to a vehicle before an avoidance maneuver.

(b) Rulebook hierarchy and induced hierarchy on realizations order.

Fig. 5: Example involving an avoidance maneuver.

Figure 2.11: Example from an applied rule engine on a classical planning system [72]

Chapter 3

AP1.2: Concepts for architectures that can incorporate *a priori* knowledge

3.1 Introduction

The goal of work package AP1.2 is *to develop architectures* for functionalities in autonomous driving that a) learn their behavior from example data using Deep Learning approaches and b) in addition are able to integrate *a priori* knowledge. This *a priori* knowledge can be physical knowledge, world knowledge or normative knowledge. The goal of this use of a priori knowledge can be to increase the efficiency and quality of the learning procedures, but also to make the results of Deep Learning procedures interpretable. Complementary to AP1.1, in which the *a priori* knowledge is exclusively used in the training procedures for neural networks, in this work package approaches are to be developed that require completely new architectures for the integration of the knowledge.

There are two fundamental challenges in integrating knowledge into learning procedures based on neural networks: First, there is the representational gap between the sub-symbolic formalisms on which Deep Learning methods are based and the symbolic representational formalisms in which *a priori* knowledge typically exists. Symbolically represented knowledge is not readily usable in neural networks, nor are—especially inner—layers of neural networks interpretable symbolically. The second challenge arises from the multitude of existing knowledge representations and associated procedures for normative, physical and world knowledge. These include rule systems, probabilistic graphical models, semantic graphs, but also mathematical representations (such as partial differential equations) and simulation models. Typically, these different formalisms are neither

compatible with each other nor easily interchangeable, but rather optimized for specific types of knowledge and usage scenarios.

In principle, there are three possibilities to close the representation gap outlined above:

- Transforming representations for *a priori* knowledge into sub-symbolic representations ("embeddings"), which can then be used in adapted neural network structures.
- Simulation of symbolic representations and processes by neural networks.
- Hybrid architectures, which combine input and output levels of the two approaches in a problem-specific way or fuse results.

Examples of all three approaches exist in the literature in various application domains and for a variety of formalisms for representing *a priori* knowledge (see, for example, [74], [75], [76]).

The objective of this document is to give an overview of the concepts developed by the partners of AP1.2 within the first phase of KI Wissen: In Section 3.2, an architecture for long-term traffic scene prediction is presented. This architecture is capable of including knowledge in form of priors about locations where objects may appear or vanish as well as about potential occlusions. Section 3.3 describes an architecture that includes knowledge about lidar hardware for finding optimal configurations and misalignment states of lidar sensors. Two architectures aiming at improved interpretation of the environment are presented in Section 3.4 and Section 3.5. While the first approach uses knowledge graphs about traffic scenes as background knowledge as priors for pedestrian detection and segmentation, the latter focuses on explicit pairwise relationships for improving detection methods for objects like traffic signs. In Sections 3.6 and 3.7, two architectures for integrating rule-based or normative knowledge into driving functions are being described. The approach in Section 3.6 aims at checking whether a vehicle is currently in a desirable state. The architecture in Section 3.7 uses a rule-engine to find optimal driving trajectories or distill behavior constrained by rules into neural driving policy. This approach will allow to integrate knowledge about scene structures and also to generate interpretable descriptions of the current scene.

All in all, the described concepts comprise methods and procedures to embed various types of knowledge directly into the architecture of data-driven AI to improve data efficiency, functional goodness, and traceability for functionalities that are essential for the use cases that guide KI Wissen, namely pedestrian detection (UC1), lane change (UC2), and rule exception (UC3). With regard to embedding the approaches in the context of the overall project, each section also outlines initial approaches for integration into the project demonstrator.

3.2 Continental - Long Term Traffic Scene Prediction

3.2.1 Goals

Safe and comfortable driving needs anticipatory planning. Anticipatory here means that we are able to incorporate expectations about the future motion of dynamic agents in the surrounding of the vehicle and in particular their interactions with each other and the static environment. Since we generally do not have access the the motion plans of the agents in our environment and agents rarely actively signal their intents unequivocally, we are forced to make inferences about their future behavior from past observations. Within AP 1.2, our goal is to derive a probabilistic long term traffic scene prediction system that facilitates anticipatory planning. By "traffic scene", we understand the entirety of all traffic participants, their interactions among each other and with the static and dynamic environment.

We want to achieve this within the framework of a Bayesian filter architecture that implements the paradigm of sequential data assimilation [77]. We choose this architecture for its firm grounding in probability theory, its computational efficiency as well as its flexibility in implementation. The Bayes filter architecture permits implementations with modular learnable components for all conditional probability distributions involved, most importantly for motion- and observation models. We restrict ourselves to interpretable state representations that facilitate validation and verification.

Ideally, the final system will be trainable end-to-end within the architectural constraints given.

The prediction system shall make "long term" forecasts, i.e. over a time span that goes beyond what can be predicted purely based on the extrapolation of kinematic states (about $2s$). Concretely, we're aiming for prediction horizons between $5 - 10s$.

The prediction system shall be probabilistic to correctly reflect the inherent uncertainties, both statistical (aleatoric) and systematic (epistemic).

The prediction system shall make multi-object multi-modal predictions. Traffic participants typically have several options of future movement from a given start state. For example, when approaching a crossing, a vehicle may turn right, go straight or turn left. Each of these options represents a well localized mode of future behavior for a single agent. When several agents are present in a scene, their individual motion options do not only combine combinatorily to form a multi-modal multi-object option space, but the agents' interactions also influence the joint probabilities of each of the modes representing a possible scenario evolution. Figure 3.1 illustrates this with two vehicles at a four-arm crossing. In particular, we note that the multi-object distribution cannot factorize into a product of single object distributions as this would imply independence of motion between different object. Our system shall be flexible enough to represent a non-factorizing join multi-modal multi-object distribution over possible scene

evolutions.

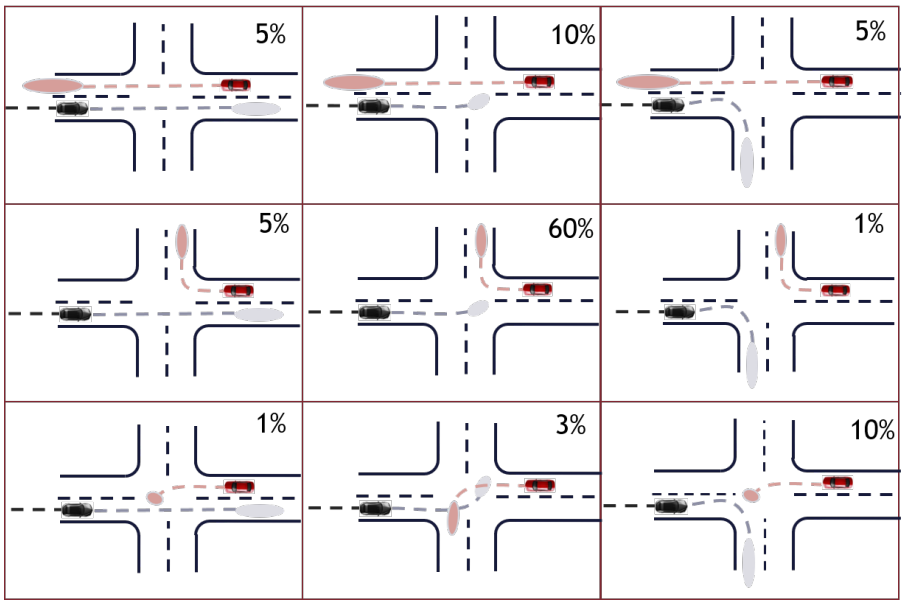


Figure 3.1: An illustration of the multi-modal nature of probabilistic multi-object traffic scene representations.

Ultimately, the architecture shall also include the ability to model the influence rules and regulations have on future behavior of traffic participants such that predictions conform to both legal and physical constraints as well as reflect the statistics of observed traffic. Figure 3.2 shows an example of a scene in which an alteration of the traffic signs should lead to a drastic alteration of predicted trajectories at otherwise identical initial conditions.

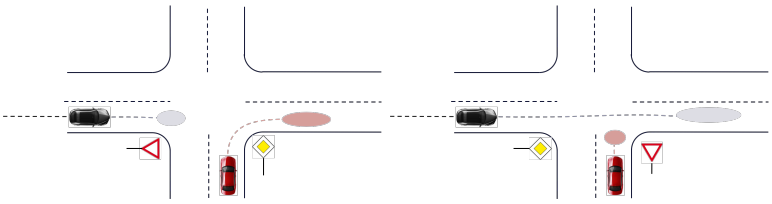


Figure 3.2: An example of how traffic rules shall influence predictions at otherwise identical dynamical situation

For this goal, we will leverage the scaffold of a state of the art multi-object (MOT) [78] multi-hypotheses tracker (MHT) [79]. Tracking algorithms are generally used to infer a system's *current* state from *past* observations, i.e. we are interested in a quantity that is localized in time and configuration

space. For a prediction system, we are naturally interested in a system's *future* evolution, i.e. we are interested in a quantity that extends in time and space. We will later see how the architecture we describe provides the facility for this by modifications of the state space and the types of observations we consider. We will introduce the trajectory representation developed in AP1.3 as single object state representation that is combined into scenes by our multi-object tracker. We will further introduce "pseudo observations" that denote putative future observations into the formalism and leverage the ability to model multi-hypotheses and thus multi-modal distributions of the tracking architecture.

3.2.2 Tracking Architecture

Below, we outline the basic architecture of the Bayes Filter and indicate at which points knowledge can be integrated. In particular, we highlight the multi-hypotheses aspect of the tracker that will provide a convenient representation for a multi-modal multi-object probability distributions for the current state of a traffic scene [79]. It is this representation that we will eventually use to represent distributions over future traffic scene scenarios. In our description and notation, we closely follow [80] and we encourage the reader to also consult this source and the excellent introductory course on multi-object tracking [81] as we can only repeat its main aspects here.

In traffic, we are dealing with a variable and a priori unknown number of agents that appear, move and disappear from the scene. Agents/objects are observed through noisy measurements. Here, we restrict ourselves to the standard point target model [82], i.e. each tracked agent is represented by a point in configuration space $\mathbf{x} \in \mathbb{R}^{d_x}$. The measurements we obtain are vectors in an observation space $\mathbf{o} \in \mathbb{R}^{d_o}$. Due to inadequacies of the detection algorithms and sensors or simply occlusions, detections $\mathbf{o}$ can be either clutter (false alarms), misdetections, or actual detections.

In order to deal with the resulting uncertainties with respect to the number of objects present in the scene, we consider a Random Finite Set (RFS) approach to tracking [83], i.e. the key variables of interest are sets and the state density we estimate through the Bayes filter is a density over sets. This is a natural choice since the any multi-object state density must be invariant under a reordering of the objects being tracked and sets provide this permutation invariance. The same is true for object detections that generally come as a set of observations without a prior assignment of individual detections to individual tracked objects.

The structure of the Bayes update will remain familiar, except that now we are dealing with sets of objects X and sets of observations O in our update equations running at time intervals δt [80]:

$$P_{t+\delta t|t}(X) = \int P(X|X')P_{t|t}(X')\mathrm{d}X' \tag{3.1}$$

$$P_{t+\delta t|t+\delta t}(X) \propto P(O|X)P_{t+\delta t|t}(X) \tag{3.2}$$

The integrations and products here appear innocuous but are not. We will hence adhere to an established state of the art representation of sets and distributions

over sets that will allow us to keep these updates tractable. The discussion below assumes a general familiarity with tracking in state space models [84, 85, 86]. As we are describing a single interaction of the filter, we will drop the time index for notational clarity.

A key element of our filter will be the Bernoulli Random Finite Set (BRFS). A BRFS X contains at most one element. The cardinality distribution is a Bernoulli distribution with parameter q, i.e. the set is empty with probability $1-q$ and contains one element with probability q. The probability of density of the BRFS is then

$$P(X) = \begin{cases} q \cdot P(\mathbf{x}) & \text{if } X = \{\mathbf{x}\} \\ 1-q & \text{if } X = \emptyset \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

where $P(\mathbf{x})$ is a density associated with the element of the set. We will use this concept to describe both observations associated to objects as well as tracked objects themselves. It is the beauty of this approach that we are free to define and use any $P(\mathbf{x})$. We will later that the ability to describe the state of a single object through $P(\mathbf{x})$ is enough to describe a multi-object scene through the random finite set approach.

Observation Model

To get started, we first describe the observation model $P(O|X)$, i.e. the likelihood for a set of observations given a set of tracked objects. We denote with $X = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ the set of $|X| = n$ targets currently tracked [80].

We denote with $O = O^c \uplus O_1 \cdots \uplus O_n$ the set of all current detections O with a *given* associations as either clutter O^c or to one of the tracked objects $O_i,\ i \in 1, \cdots, n$. The symbol $\uplus$ denotes disjoint union, which means that the independent sets O^c and O_i are mutually disjoint but together form the set O. The likelihood for the set of detections associated to tracked object i is then a BRFS

$$P(O_i|\mathbf{x}_i) = \begin{cases} P_d(\mathbf{x}_i)P(\mathbf{o}_i|\mathbf{x}_i) & \text{if } O_i = \{\mathbf{o}_i\} \\ 1 - P_d(\mathbf{x}_i) & \text{if } O_i = \emptyset \\ 0 & |O_i| > 1 \end{cases} \tag{3.4}$$

Here, $P_d(x_i)$ denotes the prior probability of detection. It allows to specify the overall sensitivity of a detector as well as circumstantial evidence such as temporary occlusions that preclude detections. Modeling $P(O_i|\mathbf{x}_i)$ as a BRFS allows to classify object i as misdetected, i.e. $O_i = \emptyset$ or associated with detection $\mathbf{o}_i$. It also enforces that only a single detection can be assigned to a tracked object. By construction, all detections $\mathbf{o}_j$ that are not associated with a tracked object are part of the set of clutter detections O^c. If the object is detected, we expect the observation distributed as $P(\mathbf{o}_i|\mathbf{x}_i)$.

The density of clutter detections over the space of observations is given by $c(\mathbf{o}$, the corresponding likelihood for a set of clutter detections is then given as

$$P(O^c) = e^{-\lambda_c} \prod_{\mathbf{o} \in O^c} c(\mathbf{o}) \equiv e^{-\lambda_c}[c(\cdot)]^{O^c}$$

where we have introduced the short-hand notation $[c(\cdot)]^{O^c}$ and the mean clutter intensity

$$\lambda_c = \int c(\mathbf{o}) \mathrm{d}\mathbf{o}$$

as the expected number of clutter detections over the entire observation range. Again, this distribution allows us to model the characteristics of sensors and detection algorithms in terms of their false alarm rate [87].

If we knew the associations of detections to either clutter or objects, we could immediately write down the observation likelihood using the above expressions. However, in MOT these are uncertain about this and thus the particular assignments have to be marginalized over which leaves us with the following expression for the observation likelihood:

$$\begin{aligned} P(O|X) &= P(O|\{\mathbf{x}_1, \cdots \mathbf{x}_n\}) && (3.5) \\ &= e^{-\lambda_c} \sum_{O^c \uplus O_1 \cdots \uplus O_n = O} [c(\cdot)]^{O^c} \prod_{i=1}^{n} P(O_i|\mathbf{x}_i) && (3.6) \end{aligned}$$

Note how the formalism of BRFS allows to express a sum over all possible associations of detections to tracked objects. We illustrate this sum with an example: Assume we have received $m = 2$ detections $O = \{\mathbf{o}_1, \mathbf{o}_2\}$ and $n = 1$ currently tracked object. The set of observations $O = O^c \uplus O_1$ can be associated to tracked objects in the following ways: $O^c = \{\mathbf{o}_1\}, O_1 = \{\mathbf{o}_2\}$, $O^c = \{\mathbf{o}_2\}, O_1 = \{\mathbf{o}_1\}$, $O^c = \{\mathbf{o}_1, \mathbf{o}_2\}, O_1 = \emptyset$. The case $O^c = \emptyset, O_1 = \{\mathbf{o}_1, \mathbf{o}_2\}$ is ruled out.

$P(O|X)$ will play a central role in our development as we will consider several possible data association hypotheses. With the observation model specified, we now turn to the description of the state density.

State Density

The tracker architecture we chose derives its name from the definition of its multi-object state density: a union of an independent Poisson Point Process (PPP) [88] and a Multi-Bernoulli Mixture (MBM) [80]. The resulting Poisson Multi-Bernoulli Mixture (PMBM) tracker models the state of an entire scene as a probability density over a set of single object states. In particular, it differentiates between undetected, or better yet-to-be-detected, objects and detected objects. The undetected objects are modeled with the PPP component while the detected objects are modeled with the MBM component.

Undetected Objects. It may seem awkward to explicitly model undetected objects in a tracking algorithm, but we'll show that this is quite sensible. The density of undetected objects is modeled as a Poisson Point Process [88] and thus effectively models our expectation for the density of undetected objects:

$$P_{PPP}(X) = e^{\int \mu(x) \mathrm{d}x} [\mu(\cdot)]^X$$

Being a Poisson Point process $\mu(\mathbf{x})$ models the intensity of object detections, i.e. $\mu(\mathbf{x})\mathrm{d}\mathbf{x}$ is the expected number of objects in volume element $\mathrm{d}x$. The density itself is modeled as a Gaussian mixture

$$\mu(\mathbf{x}) = \sum_{i=1}^{N_\mu} w_{\mu,i}\mathcal{N}(\mathbf{x}; \bar{\mathbf{x}}_{\mu,i}, \mathbf{\Sigma}_{\mu,i})$$

and accordingly, the weights of this mixture do not sum to one. Rather, $w_{\mu,i}$ denotes the mean number of undetected objects in the neighborhood of $\bar{\mathbf{x}}_{\mu,i}$. In practice, we use $\mu(\mathbf{x})$ to model the fringes of the sensor range, i.e. the boundary of the field of observation where new objects are about to enter the field of observation or where objects that have just left the field of observation may linger so that, should they re-enter the field of observation they are quickly recognized again.

In general, we will differentiate between two parts of the intensity of undetected objects $\mu(\mathbf{x}) = \lambda_b(\mathbf{x}) + \lambda_u(\mathbf{x})$ where $\lambda_b(\mathbf{x})$ models the birth intensity of new objects, i.e. rate at which new objects appear at the boundary of our field of observation, while $\lambda_u(\mathbf{x})$ is used to retain the state of objects that have just left the field of observation or have an existence probability that is too low to consider them in the set of detected objects. Their differentiation becomes necessary as $\lambda_b(\mathbf{x})$ needs to be transformed according to the movement of the sensor while $\lambda_u(\mathbf{x})$ is subject to the movement of the undetected objects. Both $\lambda_b(\mathbf{x})$ and $\lambda_u(\mathbf{x})$ are Gaussian mixtures with N_b and N_u components respectively such that $N_b + N_u = N_\mu$.

Detected Objects. The detected objects are modeled as a mixture of Bernoulli random finite sets [80]:

$$P_{MBM}(X) \propto \sum_j \sum_{X_1 \uplus \cdots \uplus X_n = X} \prod_{i=1}^{n} w_{j,i} f_{j,i}(X_i)$$

The index j runs over all "global hypotheses", i.e. components of the mixture. A global "global hypothesis" accounts for one particular scenario of n potentially detected objects in X. The $w_{j,i}$ then weigh the individual BRFS within such scenario. The Bernoulli random finite set $f_{j,i}(X_i)$ is defined as

$$f_{j,i}(X) = \begin{cases} r_{j,i}P_{j,i}(\mathbf{x}) & \text{if } X = \{\mathbf{x}\} \\ 1 - r_{j,i} & \text{if } X = \emptyset \\ 0 & \text{otherwise} \end{cases}$$

Here $r_{j,i}$ plays the role of an existence probability, i.e. whether object i part of scenario j and $P_{j,i}(\mathbf{x})$ is the single object state density. Again, note how the summation over X_i accounts for all single object state densities compatible with a set of tracked objects. If object i is not considered part of a global hypothesis j, its corresponding weight will be $w_{j,i} = 1$ and its existence probability will be $r_{j,i} = 0$. The individual $f_{j,i}$ that make a global hypothesis j are termed "local hypotheses".

We will work with a normal distribution for $P_{j,i}(\mathbf{x}) \sim \mathcal{N}(\mathbf{x}; \bar{\mathbf{x}}_{j,i}, \mathbf{\Sigma}_{j,i})$ for the single state density.

Taken together, the full state density of detected and undetected objects is expressed as

$$P(X) = \sum_{Y \uplus W = X} P_{PPP}(Y) P_{MBM}(W)$$

And again, we sum over all possible assignments of objects into detected and undetected objects to account for the underlying uncertainty.

Besides the ability of being able to describe multi-modal distributions over multi-object densities with uncertain number of objects, the above formulation of the state density has another merit: it is conjugate to the previously described measurement model. The value of conjugacy cannot be overestimated as it enables computationally efficient closed form measurement updates [80].

Structure of Hypotheses. It may be instructive to illustrate the notation with an example of the structure of the tracking hypotheses [80]. Consider Figure 3.3. At time t_0 a single object $\mathbf{x}_1$ is tracked and at t_1 we receive a single detection, i.e. $O_1 = \{\mathbf{o}_1^1\}$. This detection can naturally only be associated with object $\mathbf{x}_1$ which consequently has state density $P(\mathbf{x}_1|\mathbf{o}_1^1)$. At time t_2 we receive two object detections $\{\mathbf{o}_1^2, \mathbf{o}_2^2\}$ that give rise to the following data association hypotheses: 1) object $\mathbf{x}_1$ has been misdetected $O_1 = \emptyset$ and we are observing two new objects $\mathbf{x}_2$ and $\mathbf{x}_3$ and we form a new global hypothesis that contains three objects with local hypotheses (state densities) $P(\mathbf{x}_1|\emptyset, \mathbf{o}_1^1)$, $P(\mathbf{x}_2|\mathbf{o}_1^2)$, and $P(\mathbf{x}_3|\mathbf{o}_2^2)$; 2) $O_1 = \{\mathbf{o}_1^2\}$ and $\mathbf{o}_2^2$ gives rise to a single newly tracked object - we thus form a second global hypothesis from local hypotheses $P(\mathbf{x}_1|\mathbf{o}_1^2, \mathbf{o}_1^1)$ and $P(\mathbf{x}_2|\mathbf{o}_2^2)$; 3) $O_1 = \{\mathbf{o}_2^2\}$ and $\mathbf{o}_1^2$ gives rise to a single newly tracked object resulting in third global hypothesis containing local hypotheses $P(\mathbf{x}_1|\mathbf{o}_2^2, \mathbf{o}_1^1)$ and $P(\mathbf{x}_3|\mathbf{o}_1^2)$. Note how object $\mathbf{x}_3$ is not existent (n.e.) in the second global hypothesis and $\mathbf{x}_2$ is not part of the third but $\mathbf{x}_2$ and $\mathbf{x}_3$ must be two different objects, because of the first global hypothesis formed. The cardinality of tracked objects can differ between global hypotheses. Also note how local hypotheses correspond to leaf nodes in the hypothesis tree and are thus a characterization of the *entire* data association history of an object. Finally, note that individual local hypotheses may be part of several global hypotheses. It is only important that a global hypothesis never contains more than one leaf from an object's data association tree.

Prediction Step. The prediction step is straight forward in the linear Gaussian case that we consider here together with a constant survival probability P_s [80]. For the PPP component we find

$$\mu(\mathbf{x}) \leftarrow \lambda_b(\mathbf{x}) + P_s \sum_{i=1}^{N_u} w_{u,i} \mathcal{N}(\mathbf{x}; \mathbf{F}\bar{\mathbf{x}}_{u,i}, \mathbf{F}\mathbf{\Sigma}_{u,i}\mathbf{F}^T + \mathbf{Q})$$

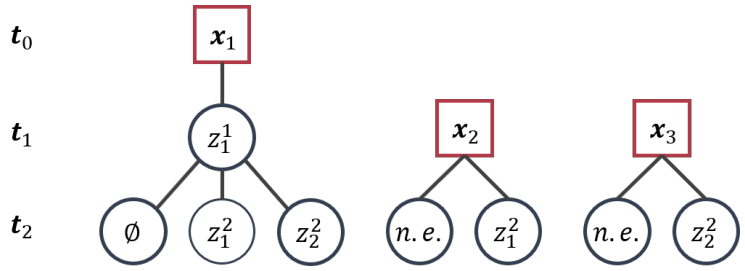


Figure 3.3: The structure of the local hypotheses trees considered in multi-hypotheses tracking. See text for details.

which shows how the survival probability shrinks the weight of the undetected components to zero if they are not re-detected. The intensity of the birth component $\lambda_b(\mathbf{x})$, on the other hand, remains unaltered.

For the MBM component of the state density, similar reasoning applies:

$$r_{j,i} \leftarrow P_s r_{j,i} \tag{3.7}$$
$$P_{j,i}(\mathbf{x}) \leftarrow \mathcal{N}(\mathbf{x}; \mathbf{F}\bar{\mathbf{x}}_{j,i}, \mathbf{F}\mathbf{\Sigma}_{j,i}\mathbf{F}^T + \mathbf{Q}) \tag{3.8}$$

and we see the standard Kalman update and the shrinkage of the existence probability with P_s. The multiplication with the survival probability leads to an exponential decay of the existence probability if an object is not detected again. The existence probability $r_{j,i}$ is reset to 1 every time an object is detected.

Update Step. We could form the posterior density $P(X|O) \propto P(O|X)P(X)$ with ease if the association of individual measurements to tracked objects was known and certain. This is unfortunately not the case. The strategy MHT trackers employ is to form the posterior of all possible data associations weighted by their respective likelihood. This leads of course to a combinatorial explosion of local hypotheses. Hence, we retain only to top k most likely associations in a mixture. Due to the conjugacy of our predictive state distribution with the observation model, the functional form of a MBM combined with a PPP is retained. Finding the most likely data association can be formulated as a linear assignment problem. It can be found with a deterministic algorithm known as the "Hungarian Algorithm" [89]. From this most likely assignment, a list of the next most likely associations can be generated deterministically by "Murty's Algorithm" [90, 91] and finding the top k most likely data associations that make up the global hypotheses in the posterior distribution is computationally cheap.

We next proceed to describe the components of the posterior and how the cost matrix for the assignment problem can be formulated. In [80], authors show that the posterior resulting from our prior state density and observation model is again a mixture of a PPP and MBM distribution. The MBM distribution now has one additional component for every observation in the set of observations that corresponds to a potential newly detected target. These potentially new

objects are assigned a Bernoulli random finite set with existence probability

$$r(\mathbf{o}) = \frac{e(\mathbf{o})}{e(\mathbf{o}) + c(\mathbf{o})}$$

where

$$e(\mathbf{o}) = P_d \int P(\mathbf{o}|x)\mu(x)\mathrm{d}x$$

is the likelihood that the detection $\mathbf{o}$ arises from any of the components in the PPP component of the state density. This is how the birth model $\lambda_b(\mathbf{x})$ and the undetected targets (re-)enter the calculation. Note how the existence probability captures the fact that if there is little support from the PPP prior for this detection compared to the expected clutter intensity, i.e. $e(\mathbf{o}) \ll \mathbf{c}(\mathbf{o})$, the object probably does not exist. The initial state density for this new object is correspondingly calculated as

$$P(\mathbf{x}|\mathbf{o}) = \frac{P_D}{e(\mathbf{o})} P(\mathbf{o}|\mathbf{x})\mu(\mathbf{x})$$

An object i modeled by a BRFS $f_{j,i}$ in global hypothesis j can be missing from the detections for two reasons: The object may not actually exist or it may be undetected. We express this misdetection hypothesis as

$$\rho(\emptyset|\mathbf{x}_i) = w_{j,i}(1 - r_{j,i} + r_{j,i}(1 - P_D))$$

If it is indeed detected, the predicted likelihood for a particular observation $\mathbf{o}$ is given by

$$\rho(\{\mathbf{o}\}|\mathbf{x}_i) = w_{j,i} r_{j,i} P_D P(\mathbf{o}|\mathbf{x}_i)$$

where $P(\mathbf{o}|\mathbf{x}_i)$ refers to the observation likelihood of the predicted single object state $\mathbf{x}_i$. Taken together, the log posterior density for a given particular assignment of a set of object detections $O = O^c \uplus O_1 \cdots \uplus O_n$ to the n detected objects in X is given as:

$$\ln P(O|X)P(X) = \sum_{i, O_i = \{\mathbf{o}_i\}} \ln \rho(\{\mathbf{o}_i\}|\mathbf{x}_i) + \sum_{i, O_i = \{\emptyset\}} \ln \rho(\{\emptyset\}|\mathbf{x}_i) + \sum_{k, \mathbf{o}_k \in O^c} \ln(e(\mathbf{o}_k) + c(\mathbf{o}_k))$$

Here, the first sum accounts for all observations $\mathbf{o}_i$ that are actually associated with a measurement, the second sum accounts for all misdetected objects, and the last accounts for all newly detected objects. The goal is now to maximize this expression with respect to the assignment of individual detections to objects. This can be done by setting up the following cost matrix $\mathbf{C}$ and solving the linear assignment problem $\min_{\mathbf{S}} \mathbf{tr}(\mathbf{S}^T\mathbf{C})$. Table 3.1 shows the general layout of the cost matrix. Assuming we have received m object detections $\mathbf{o}_i$ and are currently tracking n objects, the cost matrix $\mathbf{C} \in \mathbb{R}_+^{m \times (n+m)}$. The first n columns contain the scores for matching observations to already existing objects, while the last m columns contain the scores for considering $\mathbf{o}_i$ as the first detection of a new object. The matrix $\mathbf{S} \in \{0,1\}^{m \times (n+m)}$ is an assignment matrix such that each row sums to 1 and each column sums to either 0 or 1. These constraints enforce

	$\mathbf{x}_1$	$\cdots$	$\mathbf{x}_n$	$\mathbf{x}_{n+1}$	$\mathbf{x}_{n+2}$	$\cdots$	$\mathbf{x}_{n+m}$
$\mathbf{o}_1$	$-\ln \frac{\rho(\{\mathbf{o}_1\}\|\mathbf{x}_1)}{\rho(\emptyset\|\mathbf{x}_1)}$	$\cdots$	$-\ln \frac{\rho(\{\mathbf{o}_1\}\|\mathbf{x}_n)}{\rho(\emptyset\|\mathbf{x}_n)}$	$-\ln(e(\mathbf{o}_1)+c(\mathbf{o}_1))$	∞	$\cdots$	∞
$\mathbf{o}_2$	$-\ln \frac{\rho(\{\mathbf{o}_2\}\|x_1)}{\rho(\emptyset\|\mathbf{x}_1)}$	$\cdots$	$-\ln \frac{\rho(\{\mathbf{o}_2\}\|x_n)}{\rho(\emptyset\|\mathbf{x}_2)}$	∞	$-\ln(e(\mathbf{o}_2)+c(\mathbf{o}_2))$	$\cdots$	∞
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\mathbf{o}_m$	$-\ln \frac{\rho(\{\mathbf{o}_m\}\|\mathbf{x}_1)}{\rho(\emptyset\|\mathbf{x}_1)}$	$\cdots$	$-\ln \frac{\rho(\{\mathbf{o}_m\}\|x_n)}{\rho(\emptyset\|\mathbf{x}_n)}$	∞	∞	$\cdots$	$-\ln(e(\mathbf{o}_m)+c(\mathbf{o}_m))$

Table 3.1: Structure of the cost matrix $\mathbf{C}$ used to solve the data association problem.

that every detection is assigned to either an existing object or a new object and that no object is assigned more than one detection.

For each global hypothesis j in the prior state density, one such cost matrix is formed and we calculate the k most likely assignments. Thus, the space of global hypotheses increases k-fold. The weights $w_{j,i}$ for these new hypotheses are derived from the entries in the cost matrix for a resulting assignment and multiplied to the weights of the original hypotheses.

The update step concludes with a number of house-keeping and model reduction steps. We remove components of small weight $w_{\mu,i}$ from the PPP component since they have been undetected for too long. We keep only the N most likely global hypotheses according to their weight $w_j = \prod_i w_{i,j}$ and we remove all local hypotheses with an existence probability below a threshold or that do not appear in any of the global hypotheses we keep. Local hypotheses of moderate existence probability that would fall victim to this pruning can be added to the PPP component where they remain in a limbo state of "undetected" until they either are detected again or die out.

Figure 3.4 shows the entire component architecture of the tracker [92].

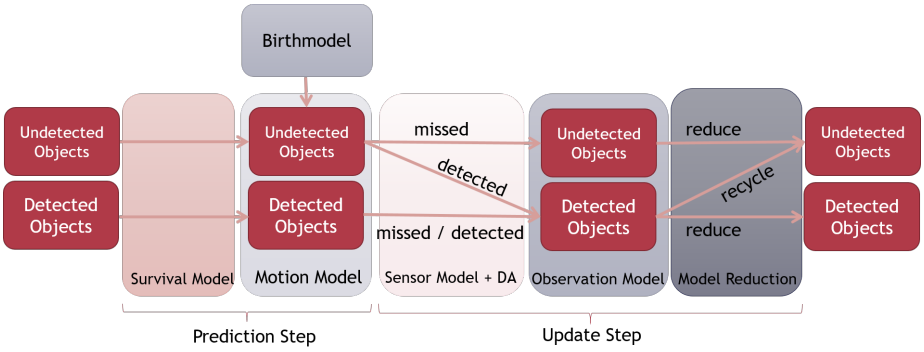


Figure 3.4: Component overview of the proposed Bayesian Filter architecture, after [92]

Integrating Knowledge in the Tracking Architecture

The architecture outlines above already contains a lot of opportunities to include situational knowledge. We'll mention three key components: The birth density $\lambda_b(\mathbf{x})$ models where objects appear. Hence, if a detection classifies different kinds ob objects such as pedestrians, bicycles or cars, we may adapt the birth density accordingly to reflect that pedestrians primarily appear in areas where the field of view intersects with sidewalks. We ma also specify different distributions over initial states for vehicles on different lanes. In right lane traffic, the expected density of vehicles entering the field of view is higher on right lanes.

Similar ideas apply to the survival probability P_s. Oncoming vehicles will leave the field of view much quicker than leading vehicles and we can reflect this in the described framework.

Finally, occlusions can be explicitly modeled and then account for different detection probabilities P_D. This is particularly important as objects that are not re-detected in every frame decrease their probability of existence over time and may thus be lost in tracking.

Figure 3.5 illustrates these ideas.

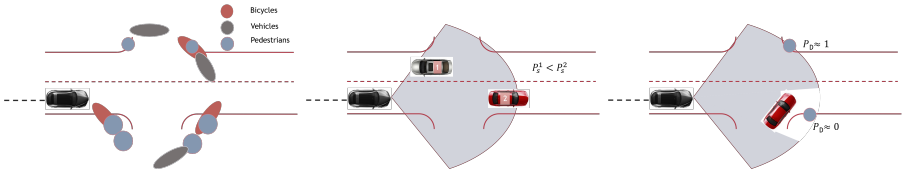


Figure 3.5: Different possibilities to include knowledge into the components of the tracker. **Left:** The static environment shapes the distribution of the birth models. **Center:** Dynamic states influence the survival probability. **Right:** Occlusions shape the detection probability.

3.2.3 Trajectories as Single Object Markov States

In AP1.3 we have motivated and developed the use of object trajectories over a constant time horizon Δt as single object states and as an alternative to the commonly used kinematic states. There, we also derived closed form Kalman update equations that can be integrated seamlessly into the filtering architecture described above. Below, we will make ample reference to this development.

Even when considering trajectories as single object states, we still only absorb past observations and thus are limited to track past movements. However, if the proposed representation is good for past trajectories, then it is certainly valid for future trajectories as well. For future trajectories, we use the notation of $\tau = 0$ corresponding to t and $\tau = 1$ corresponding to $t + \Delta t$ and recall that $\Delta t \gg \delta_t$, i.e. our prediction horizon is much larger then the cycle-time at which we receive new information.

3.2.4 Trajectory Prediction

If we consider a trajectory representation with $n + 1 = 6$ basis functions, we only need 6 linear measurements to fully determine the trajectory over the entire duration of Δt. With position, velocity and acceleration at the start $\tau = 0$, we only need 3 more and the natural choice is position, velocity and acceleration at the end $\tau = 1$. We denote these hypothetical measurements "pseudo-observations" $\mathbf{o}_f$ at $\tau = 1$ to differentiate them from earlier observations $\mathbf{o}$ at $\tau < 1$. This concept of pseudo-observations at a constant time horizon is also motivated by the rich literature on driver modeling [93] and preview control of vehicles [94, 95].

The road topology gives us a finite number of path options in the form of center-lines and we choose to restrict the position of our endpoints to lie on these path options [96, 97, 98]. We further restrict velocities and accelerations at the endpoints to be tangent to the path option. We thus only have to find the position along the path option, the longitudinal velocity and acceleration. This choice deliberately rules out the possibility for pseudo-observations on lane boundaries for example. For the time horizons considered here, there will generally not be enough evidence to distinguish such a hypothesis from either a completed lane change or a keep lane hypothesis. Note that this does not rule out predicting an object's position away from center-lines at intermediate times between t and $t + \Delta t$ - it just rules this out for pseudo-observations at $t + \Delta t$. If an object indeed crosses a lane boundary at $t + \Delta t$, our measurements will certainly lend support to the hypothesis of a lane change at some later point in time $t' \in (t, t + \Delta t)$ when we consider pseudo-observations at $t' + \Delta t$.

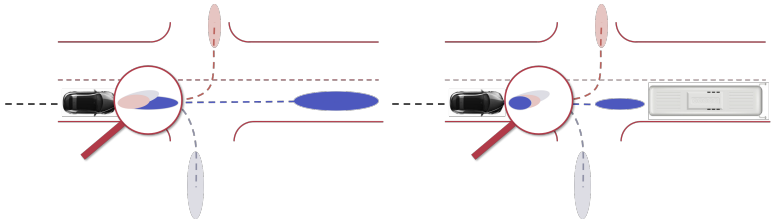


Figure 3.6: The static and dynamic environment shape the long term path options for a traffic participant. The distribution of vehicle states at $t + \Delta t$ is generally multi-modal and influenced not only by the static environment but also by other traffic participants. This multi-modal structure of the long term options shapes the expected state distribution in short time horizons $t + \delta t$.

Figure 3.6 illustrates this concept for a single vehicle in two different environmental contexts. In each environment, there are 3 path options for the vehicle. Naturally, these path options are uncertain in position, velocity and acceleration. However, since we can assume continuity of trajectories in position, velocity and acceleration, the hypothesized object state at $\tau = 1$ and the current vehicle state at $\tau = 0$ fully define a trajectory and we can propagate back the uncertainty of the pseudo observation from $\tau = 1$ to $\tau = \delta t / \Delta t$ where we will receive the

next actual observation of the vehicle state. Notice that due to the multitude of path options, we obtain a *mixture* of Gaussians for the expected distribution of vehicle positions/states in a short time horizon. This means, conditioned on the pseudo observation, we could consider three different motion models

$$P^k_{t+\delta t|t} = \int \mathrm{d}\mathbf{x}_t P(\mathbf{x}_{t+\delta t}|\mathbf{x}_t, \mathbf{o}_{f_k}) P_{t|t} \tag{3.9}$$

The probability which of the path options is likely taken is then evaluated in light of new evidence together with a prior probability $P(\mathbf{o}_{f_k})$ of the path option that can be uniform:

$$P(\mathbf{o}_{f_k}|\mathbf{o}_{t+\delta t}) = \frac{P(\mathbf{o}_{f_k}) \int \mathrm{d}\mathbf{x}_{t+\delta t} P(\mathbf{o}_{t+\delta t}|\mathbf{x}_{t+\delta t}) P^k_{t+\delta t|t}}{\sum_{k'} P(\mathbf{o}_{f'_k}) \int \mathrm{d}\mathbf{x}_{t+\delta t} P(\mathbf{o}_{t+\delta t}|\mathbf{x}_{t+\delta t}) P^{k'}_{t+\delta t|t}} \tag{3.10}$$

A sensible choice for the pseudo observations is to select the furthest point in the path together with velocity and acceleration that is reachable without violating comfort levels [99, 100, 101]. We are free to choose the uncertainties at $\tau = 1$. This approach is similar to [96, 102] but instead of forward generating the hypotheses, we interpolate back from a single step prediction. Another excellent choice is to employ a Gaussian Process conditioned on the past trajectory and the path ahead to provide an estimate of both mean and covariance of our pseudo-observations [103]. The covariances of the pseudo observations must be large enough as to cover the option space plausibly, but small enough as to be discriminative in the near future.

Figure 3.7 illustrates this concept in a toy example. A single vehicle approaches the crossing. The road topology provides three path options. We construct plausible end states as pseudo observations at $t + \Delta t$ with $\Delta t = 5s$ for each of these options together with associated uncertainties. From a common initial condition, we simulate three actual vehicle trajectories along these path options. Note how these actual trajectories deviate from the mean hypothesized trajectories due the holonomic constraints of the vehicle and the cost function of the vehicle controller.

We are now interested at what point along the vehicle trajectory, it is possible to decide which of the options is actually chosen. Figure 3.8 illustrates this. If we only compare the vehicle's position to the expectation from each hypothesis, we reach high confidence at around $1s$ into the scene, (Figure 3.8 top). If we additionally consider the velocity, high confidence is reached at about $0.5s$ (Figure 3.8 middle). Also note how the turn right maneuver is competing with the decelerating straight maneuver early into the scene as both maneuvers are decelerating and the simulated vehicle veers slightly to the left to increase the turn radius before the right turn. The bottom of Figure 3.8 shows the marginal covariances of the hypothesized trajectories at $t = 1s$ into the scene together with the actual measurement of the three simulated trajectories.

In a live system running at sub second cycle times δt the difference between future options may be to small to differentiate between, in particular as current

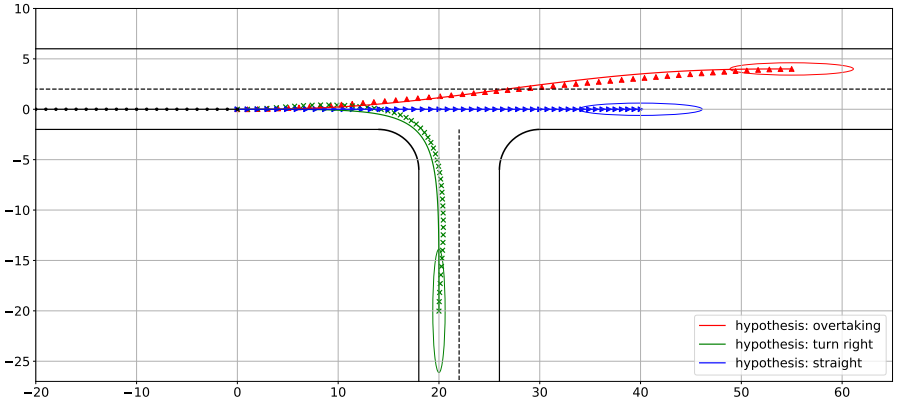


Figure 3.7: A vehicle approaches a crossing at time $t = 0$ with $v_0 = 10m/s$, $a_0 = 0$ with three hypotheses of future motion: a right turn, going straight across and decelerating possibly due a leading vehicle, and changing lanes and accelerating to overtake the leading vehicle. We show 95% confidence ellipses for pseudo-observations at $t + 5s$ and the resulting mean trajectory for each hypothesis. With individual markers, we denote three actual trajectories obtained from simulating a vehicle along these path options.

state estimates and pseudo-observations are updated at this cycle time, too, and by construction, the end of a tracked trajectory is always 100% compatible with all future trajectories resulting form pseudo-observations. Hence, the current vehicle state does not provide any evidence about which of the future options an object is going to take. We can, however, check the compatibility of the past trajectory with a future option at the current point in time. For this, we form a transition trajectory from observations made at $t - \alpha\Delta t$ on the past trajectory and at $t + (1 - \alpha)\Delta t$ on the hypothesis trajectory. Typical values for $\alpha\Delta t$ are in the range of $0.5s$ identified in our preceding analysis. On this transition trajectory, we find the expectation for the current kinematic object state at $\tau = \alpha$. The higher the compatibility between past and future trajectory, the closer this expectation will match the current kinematic vehicle state, this effectively corresponds to an "observation likelihood" for future trajectories. This approach is similar in spirit to multiple model filters [104], but here, the "models" considered are given by the path options.

We can leverage this observation likelihood directly in the framework our multi-hypotheses tracking algorithm. Before the data association step, we augment each local hypothesis (corresponding to a single tracked object) with its possible path-options and pseudo-observations. The entries in the cost matrix in the data association step are then calculated from the observation likelihood of the transition trajectories as described above [80]. I.e., we effectively employ (3.9)

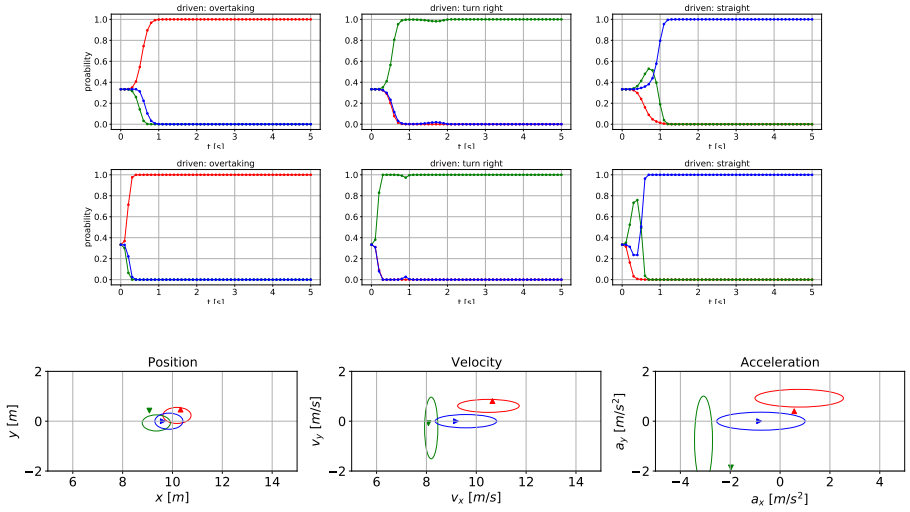


Figure 3.8: **Top:** The posterior probability of each hypothesis from Figure 3.7 for each of the actually driven trajectories when considering only position information. **Middle:** When considering both position and velocity. **Bottom:** Distribution of the hypotheses dynamical variables at $t = 1s$ into the situation together with measured quantities from driven trajectories. Colors and markers correspond to Figure 3.7.

in the calculation of the observation likelihood. The standard data association algorithm (Murty's Algorithm [90, 91]) now produces a ranking of the most likely assignments of object detections to local hypotheses under the constraint that all object detections are assigned and each local hypothesis is assigned at most one observation. Each of these assignments forms a new global hypothesis weighted by its total likelihood and thus provides a multi-modal multi-object representation of the scene. We alter this algorithm slightly by adding the additional group constraint that in each global hypothesis, only one of the possible futures for a local hypothesis can be present. This constraint can be fulfilled trivially for the most likely global hypothesis by considering only the maximum entry in each group of future trajectories. As Murty's algorithm generates alternative assignments from this globally optimal solution, we only need to adapt the exchange rules in order to reflect the additional constraints. The resulting ranking of data associations represents a probability weighted multi-modal multi-object representation of the current traffic scene and its future development. The ranking can then be further pruned by applying additional constraints such as traffic rules. The pruning of highly probable hypotheses that violate traffic rules can be taken as a consideration of an imminent violation of traffic rules.

3.2.5 Models to Generate Pseudo Observations

From the preceding discussion, it is clear that the generation of pseudo-observations that on one hand cover the space of options and on the other hand are distinctive enough to provide evidence for discerning them on a short term horizon is the central component that must be developed. Hence, we need prediction models that predict plausible agent positions at time $t + \Delta t$ given information available at time t.

We approach this step-wise with increasing complexity. With increasing complexity of the situation, we will further increase the complexity of our modeling. For models of higher complexity, we can employ curriculum training by reusing the data sets of simpler situations to start training and possibly also include distilled knowledge gained from simpler models. All models will take the past trajectory of the vehicle to be predicted as input.

As a first level of complexity, we consider a single given path ahead, i.e. the future motion of an agent to be predicted is entirely by the road layout. We can distinguish two different cases: the path free of other agents and the path has other agents on it. For vehicles, this means we must model the typical behavior of slowing down before curves and accelerating out of curves. Such a situation is typically called the "follow road" scenario.

As a second level of complexity, we plan to consider a single given path with other vehicles on it, i.e. the future motion of an agent is dependent both on the path and on the traffic participant in front. Such a situation is typically called the "follow vehicle" scenario. For large separating distances, we expect to recover the "follow road" scenario while for shorter distances, the fact that we represent agents through their past trajectories will prove beneficial as drivers naturally keep a safety distance that is roughly constant in time.

These two scenarios may appear overly simplistic, but they are in fact applicable to all vehicles near the end of the sensor range when only map information about the traffic situation ahead is available. The only sensible prediction for these vehicles is then given by the follow road scenario *marginalized* over the expected distribution of traffic outside the sensor range and will naturally be associated with greater uncertainty than either thee free path or the path with observed agents on it. In fact, the two situations blend into each other as for most vehicles, part of their future path is within the sensor range and part is outside it.

These situations can be modeled by employing one of the many driver models that aim to mimic human driving [105, 93, 106, 107] and are verified to a very high degree of detail [108, 109]. Alternatives are planning, more specifically model predictive contouring approaches [110] or simpler heuristic planners such as timed elastic bands [111, 112].

Predictions from such models under parameter variation can then be distilled into Gaussian mixture models that represent the joint density of past trajectory representation for the vehicle, path representation, and pseudo-observation. Predictions are then obtained straight forwardly from conditioning on the observable information. This approach directly results in uncertainty estimates for the

pseudo observations [103, 113]. An alternative are simple feed forward mixture density networks [114, 115] or normalizing flows [116, 117].

In the development of the trajectory representation, we stressed the ability to perform tracking in the coordinate frame of the ego vehicle given by the fact that we can easily transform our state vector to any new coordinate frame. For the first two levels of complexity, we can leverage this and make predictions for every agent from their respective frame of coordinates, keeping in mind that the observability of the path ahead does not change by this transformation.

Both of these models will assume uni-modal distributions for the pseudo-observation for a given path option. Data for training these models can be obtained from real world recordings or from simulations of vehicle physics, i.e. controlling a vehicle to drive comfortably along a path with one of the described models.

The third level of complexity contains all situations where several vehicles and several path options exist. Here, too, we will increase the complexity by first considering situation without oncoming or cross traffic. Then we increase the number of path options and traffic participants. The simplest situation would then be one vehicle behind another with the option of overtaking and staying behind the vehicle. This is the first situation in which we expect a multi-modal output for the pseudo-observation of a single vehicle.

If we restrict ourselves to models with a known number of components, we could again work with mixture density networks. An alternative would be to use the invariance under time reversal of our trajectory representation and treat the problem as an instance of inverse dynamics. To to this, we recall that all predicted trajectories start at the current kinematic vehicle state. Reversing time means that treating the pseudo-observations as start states and looking back in time, we are seeing trajectories that all converge in one kinematic state - which the vehicle is currently in. Thus, we have formulated an inverse problem where a single end state may arise from multiple start configuration. For solving these, a number of architectures exist in the literature that are trained on the forward process and by construction can also model the inverse process with the same accuracy [118, 119, 120]. This is quite convenient for a number of situations in our application domain. The drawback of these models, however, is that they are currently specified for fixed input sizes and we would like to have a flexibility in the number of paths options and vehicles in the input.

This class of models that can deal best with varying input size due to varying complexity of a situation are transformer models [121, 122, 123, 124]. In fact, they are specified for input as sets and generally add a positional encoding to remove their natural permutation invariance. We of course are looking for precisely this feature. We would encode the available path options and past trajectories of traffic participants as context and generate pseudo-observations one-by-one as commonly done in a language model. The available paths and possibly other vehicles would be encoded as context and the transformer would then be seeded with the path option and vehicle we would like to predict and should output a future pseudo-observation. These would ideally be mixture distributions again so that we can sample the output and build distributions

over entire scenes. In analogy to automatic translation, past trajectories and paths are the words of the text to translate and pseudo-observations are the words of the language into which they are translated.

The attention mechanism of transformer models would make the predictions explainable to a certain extent. In an ideal case, they reveal the dominating influences in the generation of a pseudo-observation.

Finally, we would like to include the influence on traffic rules and signs in the prediction. This seems most easily possible if pseudo-observations could be checked for their legality. The legality of a pseudo-observation is then most easily represented as prior probability $P(\mathbf{o}_{f_k})$.

3.2.6 Connections to other APs and TPs

The closest connection we have is certainly that with AP1.3 from where we have obtained the formulation of the state representation and two equivariance constraints that we can test against the implementation the generation of prediction models for the pseudo-observations: Is a model trained to predict pseudo observations at $\Delta t/2$ consistent with one trained to predict them at Δt when applied twice? And: Are the prediction models equivariant under a change of coordinates?

Further, we will collaborate with AP1.4 in developing a representation of traffic rules that results in effective priors $P(\mathbf{o}_{f_k}$ to represent the legality of pseudo-observations.

Last, if the transformer approach on predicting pseudo-observations for entire traffic scenes proves successful, we can then start at investigating the attention maps to make the predictions explainable. Figure 3.9 illustrates these ideas

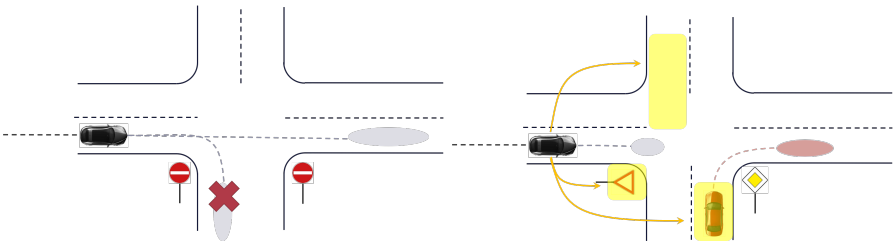


Figure 3.9: **Left:** Traffic rules provide priors on the legality of path options. **Right:** Attention maps can visualize and plausibilize prediction of future trajectories.

3.2.7 Demonstration Concept

The demonstration concept for the traffic scene prediction system is quite straight forward. Input data about object detections is received from the demonstrator and filtered using the proposed multi-object multi-hypothesis tracker. If working

in simulation, we can consider the data association problem solved and need the multi-hypothesis capability only to represent different possible future scenarios.

The path options for agents in the scene are derived from the provided map of the scene.

We can then proceed to demonstrate the prediction capabilities open loop, i.e. we only compare predicted trajectories to observed ones and do not interact with the scene. We'll demonstrate this capability in various use cases of varying complexity in accordance to those defined in AP4.1. In particular, we'll be demonstrating at which point in time we reach high confidence in an agent's future trajectory similar to the example shown in Figures 3.7 and 3.8. This will be done in highway scenarios where we are interested in cut-in and cut-out maneuvers and merge situations with multiple vehicles coordinating and interaction as well as in city scenarios involving crossings of various layouts and roundabouts.

Provided the necessary demonstrator capabilities for open loop operation, we'll further evaluate driving comfort when planning vehicle motion taking our future scenario prediction into account. For this, we'll consider the same use cases as in the open look scenarios and expect to show increased comfort when negotiating complex maneuvers such as lane merges and intersections.

3.3 Valeo - Incorporation of Sensor Hardware Design

3.3.1 Introduction

Reliable perception of the environment is crucial for autonomous driving. The vehicle has to recognize the traffic, obstacles and vulnerable road users with a high confidence in order to induce the right maneuvres, e. g. giving way to other vehicles, dodging other objects or initiating an emergency braking. Different types of sensors have been developed for that purpose, such as camera, ultrasound, radar (radio detection and ranging) and lidar (light detection and ranging). A camera lacks distance information since its image is only a two dimensional projection, and does not work in darkness. Ultrasound works only in the near field range up to 10 metres, for example as parking sensor. The angular and distance resolution of a radar is not very good and not sufficient for, e. g., object detection. These gaps are filled by the lidar, which provides distance information, operates also in complete darkness and covers medium to long distances from about 5 to 60 metres. A large part of the automotive community considers lidar to be essential [125].

A lidar sensor determines the distance of objects by measuring the time of flight of laser beams, which are reflected by an object. Currently, the most commonly used lidar in the automotive industry is the *mechanical spinning* lidar, where a mirror rotating around a vertical axis is used and several vertically aligned laser diodes are used to scan a certain solid angle [125]. The stack of laser diodes can additonally be multiplied in vertical direction to increase the

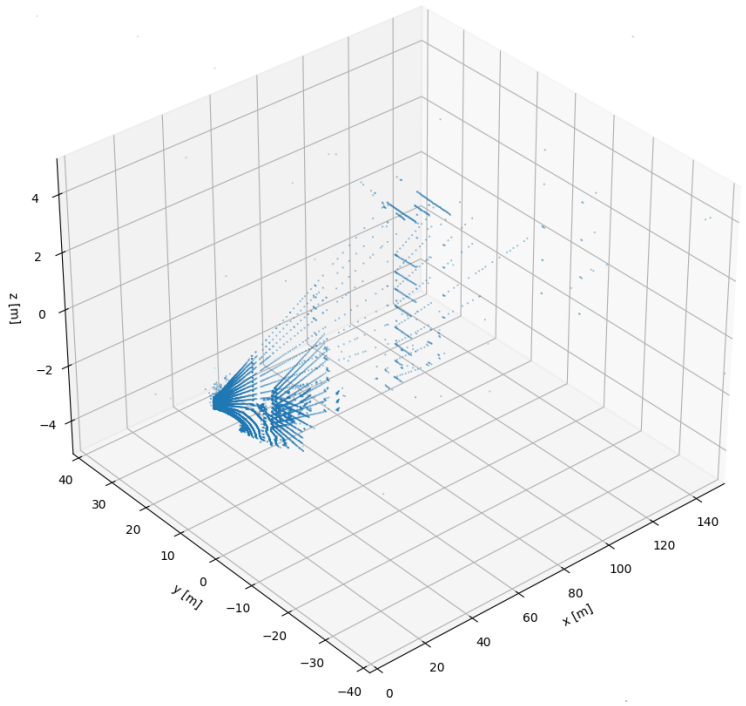


Figure 3.10: A point cloud generated with a Valeo lidar sensor (model *Scala 2*). The scanning angle lies in the xy plane.

field of view further. The result is a point cloud representing the environment of the car. Figure 3.10 shows an example for such a point cloud, recorded with a Valeo lidar sensor, *Scala 2*. Up to now, this is the only lidar sensor in series production [126].

Deep learning for camera-based sensor data has been developed for many years, while deep learning for lidar observations is still in its infancy due to the unique challenges that arise from the sparse point cloud structure. However, this field gains more and more attention not only in autonomous driving, but also in the fields of computer vision and robotics, and algorithms have already been developed for classification, object detection, instance/semantic segmentation and tracking, similar to the classical image processing [127].

The serial production of the Valeo Scala sensor started in 2017. On the basis of past experience it is known that over the years the optical parts of the lidar sensors misalign due to, e. g. evaporation of solvent from the glue which is used to fix the lenses, or vibrations during operation in the car. By manual misalignment of a sensor it could be shown on the test bench that the detection range of a norm object decreases. However, during normal operation of the sensor in a car this effect cannot be easily detected since it is superimposed by

external effects such as the heavily varying surrounding

The reliability of the sensor misalignment detection is a very important point: false positive detections lead to high costs and dissatisfaction of the customer, false negative detections compromise safety. Therefore, the aim is to build a neural network which reliably detects optical misalignment. Since the misalignment caused by the above mentioned reasons happens over years, obtaining training data is difficult, and a lot of data is needed for a statistical analysis, for example to train a neural network. For this reason, it is of high interest to increase the reliability of the misalignment detection not only by increasing the amount and quality of data, but by integrating knowledge of the sensor in the neural network. In addition, if knowledge about the sensor is taken into account in the neural net architecture, the networks can become simpler and faster.

3.3.2 Concept and Architecture

It is planned to use a state of the art convolutional neuronal network for point clouds, such as *SalsaNext* [128]. This neural network is designed for semantic segmentation and will be adapted to our classification problem of sensor misalignment. It projects the point cloud from three dimensions onto a two-dimensional spherical surface, the so-called *range view image* [129]. This projection allows for convolution operations in the following network layers, which are well established in image processing neural networks.

A source of knowledge for possible improvement of artificial intelligence is the internal hardware setup of the lidar sensor. As described above, the rotating mirror and the vertical stacks of laser diodes result in multiple vertically shifted horizontal layers, see figs. 3.11 and 3.12. The *Valeo Scala 1* lidar sensor, for example, has four layers, the *Scala 2* has 16 layers and *Velodyne* manufactures lidars with 16, 32, 64 and 128 layers. This knowledge can be used to arrange the points of the point cloud. *SalsaNext* assumes an unstructured point cloud which needs to be projected to the two-dimensional spherical surface for further processing. However, knowing the layer number and the number of the points in each layer (701 in total in the case of the *Scala 2* sensor) the points can already be arranged in a two-dimensional image. For every point the echo pulse width is recorded, which is the width of the returning echo pulse at a certain threshold voltage. The cartesian coordinates, (x, y, z), distance and echo pulse width are stored as channels in the range view image, resulting in an array of dimension $[N \times M \times 5]$, where N is the total number of layers and M is the total number of scan points in one layer. This approach furthermore avoids discretisation errors which would be caused by a projection.

The point recording sequence of the sensor is also known. First all points belonging to the same horizontal angle are recorded and then the horizontal angle is varied. This knowledge is not used in a normal unstructured point cloud where the actual sensor design is not regarded. This could justify convolutional kernels with a larger size in vertical than horizontal direction, in order to weaken the effect that the car moves while the lidar is scanning. In the case of *Scala 2*,

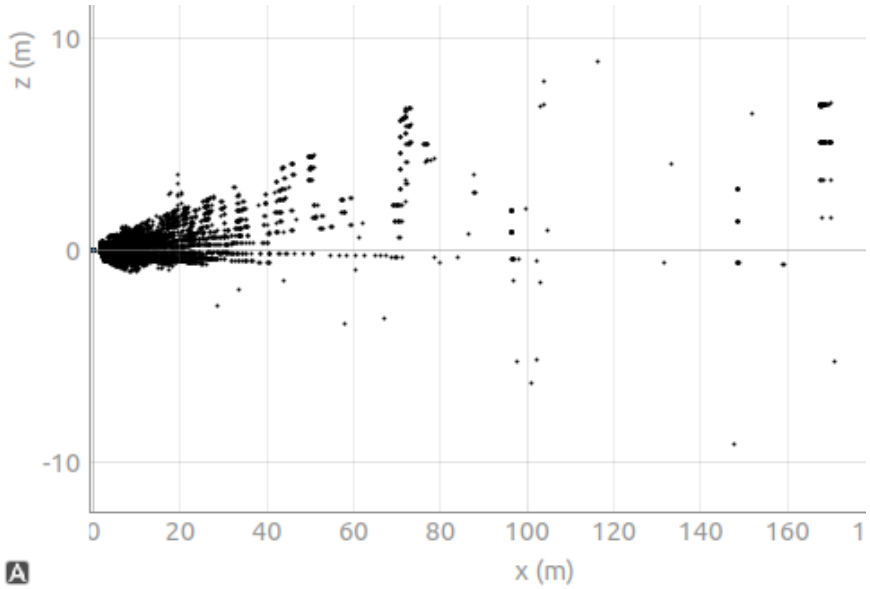


Figure 3.11: Lateral view of a *Scala 2* point cloud. The viewing direction of the sensor is from left to right. The vertical layer structure is clearly visible.

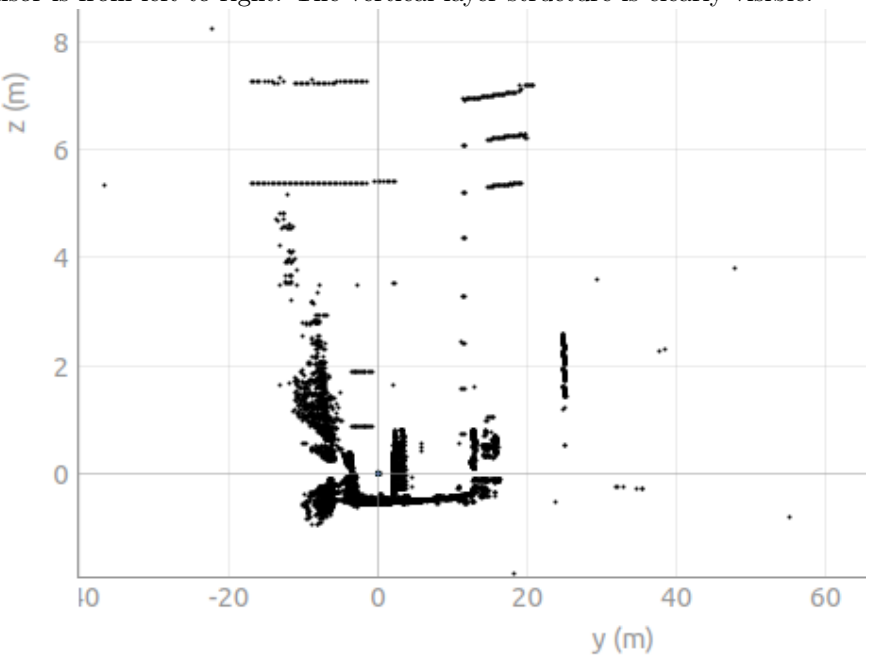


Figure 3.12: View in driving direction of a *Scala 2* point cloud.

which has 16 layers, a filter size of, e. g. $[16 \times 1]$, $[16 \times 3]$ or $[8 \times 1]$, $[8 \times 3]$ or similar could be a promising try. In the case of lidars with a much larger number of layers (e. g. 128) it is not reasonable to create a filter covering all layers, however filter covering half or fourth of it should be tested. The actual filter sizes working best in relation to the number of layers have to be found by testing. Even sized kernels are rarely used in CNNs, however, it has been shown that in combination with the right padding it outperforms $[3 \times 3]$ kernels in image classification and generation [130].

Furthermore, it is known how many photo diodes are in one stack, that means, how many photo diodes are hit with the same laser shot at the same time. This knowledge can also help to optimize the filter kernel size. If the stack contains p photodiodes, a filter size of $[p \times 1]$, $[p \times 3]$, ..., is promsing. It must be tested if it is instead more important to choose the kernel size larger or smaller than the stack size, to cover any kind of effects. It will be interesting to find out if even the knowledge of the divergence of the laser diodes can be used to find an optimal kernel size.

Besides the kernel size, the stride can be adapted to the layer number and stack size. It is worth testing if a stride of the stack size p or a bigger/smaller stride gives a competitive edge. In that case, calculation speed would increase, which would be beneficial for nets working real time in the car.

SalsaNext uses dilated convolutions [131] (also called atrous convolutions), which allow for larger context information without increasing the number of parameters. It is not clear if an adaption of the dilation rate to the stack size is reasonable, however, it should be taken into account. (SalsaNext uses a moderate dilation rate of 1 and 2, respectively.)

Figure 3.13 shows the planned architecture of the neural network. It is derived from SalsaNext [128] and adapted to the classification task. Furthermore, the size of the arrays is adapted to the (variable) input size, depending on the sensor type. N is the number of layers of the input point cloud and M is the number of points per layer. The number of repetitions of the convolution layers in the "repeat" block depends on the number of layers of the input point cloud. (It lies between 4 and 128 layers for the models known.)

The knowledge of the lidar sensor can simply be represented in a file which can be imported by the neural net, for example as a YAML (*YAML Ain't Markup Language*) file. Each lidar sensor model will need its own file with the needed data.

3.3.3 Testing

The misalignment of the optical parts due to the reasons mentioned above usually appears after several years of operation. A trivial way to test the plausibility of detected misalignment is a comparison with the actual age of the sensor or its operating hours. This could be stored internally in the sensor.

Furthermore, it is well-established knowledge that misalignment due to ageing happens gradually and not suddenly. In addition, misaligment is irreversible. A validation of the classification of the network is therefore possible by saving the

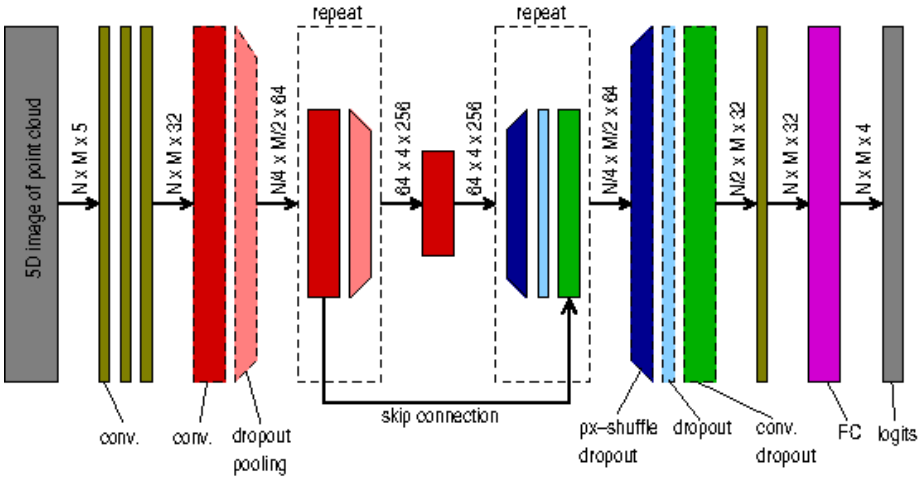


Figure 3.13: The SalsaNext architecture [128] adapted with the knowledge about the spinning lidar hardware. N is the number of layers and M is the number of points in one layer. Additionally, the filter sizes of the convolutional layers are changed depending on the knowledge (not depicted here).

classification results over a specific period and comparing with earlier determined misalignment states. The observation of misalignment over a longer time is in practice feasible, since these effects are supposed to be detected before they become security-relevant.

For tests during development it is possible to mix point clouds belonging to different misalignment states and input them in wrong order, e. g. no misaligment after strong misalignment. However, it has to be taken into consideration, that the time scales between testing and real time operation differ. This can be taken into account by a parameter which can be adapted for the tests.

3.3.4 Use Cases

Assuring proper functioning of the lidar is crucial. The lidar sensor is very important for the lane change use case, but also for the pedestrian detection use case.

Among other things, lidar sensors are used for object detection. Their main field of application is driving on motorways, where it is used for lane detection. In order to be able to induce a lane change, it is important to know if the lane and free space detection of the lidar is reliable. For this purpose, the stage of misalignment (“ok”, “light”, “medium”, “strong”) is sent to the subsequent processing units, which then decide if the lidar data can be used for driving manoeuvre decisions. Furthermore, the driver has to be informed (for example by a warning light in the dashboard) that the lidar needs an inspection. As an

alternative to the misalignment class, a confidence measure, ranging for example from 0 to 100, can be determined and forwarded.

3.3.5 Further knowledge for possible integration

Valeo has a long experience with lidar algorithm development and a lot of knowledge about lidar sensors and their ageing effects exists. The integration of knowledge into the network is, however, the most challenging step. It is planned to iteratively increase the complexity of knowledge. The detailed further line of action depends on the results of the integration of the sensor hardware knowledge described above. If the results are good, it will be interesting to use them to improve semantic segmentation on lidar point clouds.

A further possible approach is the knowledge that optical misalignment increases the spaces between the reflected laser beams in the grid of the scan, due to less overlapping of the area covered by the sender and the receiver, respectively.

Additional knowledge resulting from the bigger distances in the scanning grid is the fact that the range of the lidar decreases, because less laser light from objects at large distances is detected.

Another effect resulting from this cause is that narrow objects, such as reflector posts at the side of the road, are detected in less frames, causing them to appear and disappear with different frequency compared to a sensor with no misalignment. If appropriate labeled training data exist, this knowledge could be used by the network.

3.3.6 Demonstration Concept

The effectiveness of the misalignment detection will be tested with simulated data, e. g. from the TP 4 demonstrator. Point clouds recorded by a Lidar front sensor are directed to the misalignment detection module for analysis. The module will output the level of misalignment and the confidence of the classification. The following modules are able to decide if the Lidar data is reliable or has to be discarded. Depending on the needs of the demonstrator, the Robot Operating System (ROS/ROS2) may be used for communication and a Docker container for encapsulation, ensuring the necessary dependencies.

3.4 DFKI - Applying Knowledge on Visual Primitives

3.4.1 Introduction

Visual perception deals with the recognition and understanding of vision-related primitive stimuli. These visual primitives include colors, edges, depth, movement, and gradient, etc. They play a crucial role in visual perception tasks. Nowadays, Neural Networks are among the most popular choice for solving various problems

in different domains. Neural Networks learn from low-level features to gradually build representations of more abstract concepts.

Perception plays a crucial role in attaining the goal of autonomous driving. An ego-vehicle is generally equipped with a variety of sensors including cameras, LiDAR and Radar. These sensors serve as the senses of an ego-vehicle and therefore enable the capability of perceiving the environment around the ego-vehicle in different spectra. Object detection, and in particular pedestrian detection, has significant importance in the perception spectra as it serves as a critical piece of information for the downstream tasks associated with the autonomous driving pipeline [132].

3.4.2 Literature review

The neural network-based object detection approaches can be divided into two main categories: Single-Stage and Two-stage approaches. Single-stage approaches are generally based on a fully convolutional architecture and consider the object detection problem as a simple regression problem [133]. For a given input image, the Single-stage detectors learn class probabilities and the coordinates of a bounding box encompassing an object. Most popular examples of single-stage detectors include Single Shot MultiBox Detector (SSD) [134] and You Only Look Once (YOLO) [135]. On the other hand, Two-stage approaches are more sophisticated, where each stage specializes in a sub-task which eventually contributes to the final output of the system. The first stage is responsible for identifying the region of interest and the second stage is responsible for the object classification and bounding box regression. The most common examples of two-stage detectors include Faster R-CNN [136] and Mask R-CNN [137]. Both single and two-stage approaches have distinct pros and cons. Most notably, Two-stage approaches yield better detection accuracy than Single-stage approaches as they have specialized stages where the output of the second stage is built on top of the output of the first stage. However, Single-stage approaches are much faster than Two-stage approaches as they don't have an additional stage with supplementary computational overhead.

Pedestrian detection is applied in various vision-based applications ranging from surveillance to autonomous driving. Despite their good performance, it is still unknown how the detection performs on unseen data. Hasan et al. [138] presented a study in quest of generalization capabilities for pedestrian detectors. In their cross-dataset evaluation, they tested several backbones with their baseline detector Cascade R-CNN on famous autonomous driving data sets including Caltech, CityPersons, ECP, CrowdHuman, and Wider Pedestrian. Cross-dataset evaluation is an effective way of evaluating a method on unseen data and validating its generalization capability, otherwise, a method may over-fit on a single dataset. The authors demonstrated that the existing pedestrian detection methods perform poorly when compared with general object detection methods given larger and diverse datasets. A carefully trained state-of-the-art general-purpose object detector can outperform pedestrian-specific detection methods. However, it highly depends on the training pipeline and the selected dataset.

In [138], the authors used large datasets that contain more persons per image. Such general-purpose datasets, usually collected by crawling the web and through surveillance cameras, are likely to have more human poses, appearances, and occlusion cases as compared to pedestrian-specific datasets. In [138], it was also shown that progressively fine-tuning the models from largest general-purpose to smallest target domain dataset, results in performance improvement. The generalization ability of pedestrian detectors has been compromised due to the lack of diversity and density of the pedestrian benchmarks. However, benchmarks such as WiderPerson, Wider Pedestrian, and CrowdHuman provide relatively higher diversity and density.

Some of the recent state-of-the-art pedestrian detectors include Pedestron [138], Center and Scale Prediction (CSP) [139], Adapted Center and Scale Prediction (ACSP) [140], Attribute-aware Pedestrian Detection (APD) [141] and Mask Guided Attention Network (MGAN) [142]. With reasonable visibility conditions, the current state-of-the-art detectors seem to work well with still room for improvement. However, when it comes to occluded cases a robust method is needed which can detect pedestrians even under heavy occlusions.

Pedestron [138] takes the approach of training Cascade R-CNN [143] based two-stage object detector with additional datasets i.e. training model on multiple datasets and fine-tuning it on target dataset. Pedestron achieve state-of-the-art results on Caltech[144] and CityPerson [145] datasets in similar fashion [138]. Pedestron achieves a MR^{-2} of 7.5% and 1.76% in reasonable settings on CityPerson and Caltech datasets respectively [138]. MGAN [142] provides the network with further visibility information on ground truth detections to let the model focus on learning visible area which belongs to pedestrian rather than learning occlusions as a part of the pedestrian.

CSP [139], ACSP [140] and APD [141] are anchor free, end-to-end FCN based approaches towards pedestrian detection which are catching the attention of the community. Anchor-free approaches are similar to CenterNet [146] which was originally proposed for multi-class object detection. CenterNet [146] takes an end-to-end CNN-based approach to predict center heatmap along with a scale instead of traditional bounding box regression in one and two-stage detectors. CSP [139] takes objects as a points-based approach to pedestrian detection with the center as a single point to represent object location along with scale and center offset as additional output to predict the scale of the detection and center point offset respectively. Since the predicted center heatmap output in CSP [139] has reduced resolution compared to its inputs, there is a need for an offset to locate centers precisely. ACSP [140] improves on CSP [139] by adding Switchable Normalization introduced in [147] for better convergence and robustness [140]. APD [141] uses a powerful backbone and adds additional pedestrian-oriented attribute features as output to detect and handle crowded cases in a better way by employing an adaptive NMS based on attribute features [140, 141].

Computer Vision related tasks have been known to be more reliant on merely the features available in an image. However, there have been several efforts to use external background knowledge i.e., text, knowledge graphs, etc. Deng et al. [148] introduced Hierarchy and Exclusion graph which represented the

	Caltech [144]	CityPersons [145]	Euro CityPersons [153]
Images	42,782	2,975	21,795
Persons	13,674	19,238	201,323
Persons/image	0.32	6.47	9.2
Unique persons	1,273	19,238	201,323

Table 3.2: Autonomous driving datasets statistics [138].

semantic relation between any two labels applied to the same object. This graph was then later used for probabilistic classification of visual objects in images. Vondrick et al. [149] used mined text from a large corpus of data to use alongside images to facilitate the prediction of the motives of action. Wu et al. [150] combined information extracted from general knowledge-base with images to better tackle the problem of visual question answering. Lu et al. [151] proposed a model to predict multiple relations among objects in a given image. For this purpose, they employed semantic word embeddings to build language priors from existing text. Hong et al. [152] suggested using co-occurrence information to predict the presence of certain objects in an image that are related to already detected objects in that image.

3.4.3 Pedestrian Detection & Segmentation

Pedestrian detection deals with the identification of pedestrians in the environment around an ego-vehicle. There exist approaches in the literature which perform pedestrian detection only using LiDAR sensors. However, such approaches are usually not popular in the community due to fact that the features obtained from camera images are significantly rich as compared to the ones obtained from LiDAR or Radar. On the other hand, different approaches use LiDAR to incorporate depth information into the image data for the pedestrian detection task. Therefore, the approaches used to perform pedestrian detection mainly using camera images are generally widely adopted. The images from the mounted cameras serve as an input from which individual pedestrians are identified and are enclosed in a bounding box. A variety of solutions have been proposed to effectively identify individual pedestrians in the surrounding environment.

Datasets

The publicly available major datasets for training and testing pedestrian detectors include Caltech [144] pedestrian dataset, CityPersons [145] dataset which is a subset of CityScapes [154] dataset and Euro CityPersons [153] dataset. Other datasets include Wider Pedestrian [155] and CrowdHuman [156] which are employed in some pedestrian detectors as additional data to make models more robust. Table 3.2 shows statistics for Caltech, CityPersons and Euro CityPersons datasets.

Baseline methods for pedestrian detection

Currently, there are two baselines for pedestrian detection: one uses additional data for training while the other does not. These two baselines have to be separate since, using additional data in most cases increases model performance, as the model becomes more general and aware of the larger context, provided the model can learn this information. The current baseline method for pedestrian detection with additional data for training is Pedestron [138] while baseline methods for training without additional data are Adaptive Center and Scale Prediction, APD [141] and MGAN [142]. In the meantime, we intend to proceed with these selected baseline approaches and based on their initial results we will select the optimal approach for the task of pedestrian detection.

Evaluation Settings. There are two different evaluation settings most commonly used in the community for pedestrian detection. The first one is a general setting that addresses visibility as well as detection sizes, it includes Reasonable, Small, Heavy Occlusion, and All sub-settings as shown in Table 3.3. The second one focuses on visibility alone with sub-settings being Reasonable, Bare, Partial, and Heavy as shown in Table 3.4.

Evaluation Criteria. The most common measure used for comparison under above mention settings is Log average Miss rate (LAMR) over False Positive per Image (FPPI). MR^{-2} indicates the Miss rate with $FPPI \leq 10^{-2}$ and it is followed by all baseline methods. Without putting an upper limit to FPPI, methods can achieve lower LAMR by simply predicting a large number of detection which will result in a huge number of false alarms.

Baseline Methods for Traffic Scene Segmentation

Traffic scene segmentation can be seen as a semantic scene segmentation problem. In our case, urban driving datasets like CityScapes [154] can be used as data for training. The current state-of-the-art method for semantic scene segmentation is HRNet-OCR [157]. HRNet-OCR employs a hierarchical multi-scale attention mechanism that allows flexible and faster inference along with improved meanIoU. HRNet-OCR achieves meanIoU of 85.1% and 61.1% on CityScapes test set and Mapillary validation set respectively [157, 158].

Setting	Height	Visibility
Reasonable	[50, ∞]	[0.65, ∞]
Reasonable Small	[50, 70]	[0.65, ∞]
Heavy Occlusion	[50, ∞]	[0.2, 0.65]
All	[20, ∞]	[0.2, ∞]

Table 3.3: General experimental settings.

Setting	Height	Visibility
Reasonable	[50, ∞]	[0.65, ∞]
Bare	[50, ∞]	[0.9, ∞]
Partial	[50, ∞]	[0.65, 0.9]
Heavy	[50, ∞]	[0, 0.65]

Table 3.4: Occlusion experimental settings.

3.4.4 Knowledge Incorporation

Ever since the dawn of neural networks, vision-related tasks have gained significant improvements over the years. Object detection is one of the most popular vision tasks. It has countless applications, the most notable is autonomous driving. Most of the object detection approaches only takes the image information into account missing out on traffic scene-related external knowledge. Traffic scene knowledge is usually provided in the form of a knowledge graph. Which consists of several entities in a traffic scene and their semantic relation with each other.

Knowledge graphs are generally quite enormous sometimes with hundreds of entities. Curating and maintaining a knowledge graph has proved to be a very challenging task. Paulheim [159] performed a survey of existing knowledge graph generation approaches and their refinement processes. This survey also highlighted the trade-off of using certain approaches instead of others.

Knowledge graphs have been involved in a diverse variety of tasks including image, text, and web domains. Dong et al. [160] employed knowledge graphs alongside the extracted web content from analysis of text, structure, annotations, and tabular data to fuse both information sources. The fused information is then later used to perform analysis for factual correctness.

3.4.5 Pipeline Overview

Traffic scene knowledge is generally provided in the form of a knowledge graph, which consists of several entities in a traffic scene and their semantic relation with each other. A traffic scene knowledge graph not only provides us information about the entities in a traffic scene but also includes a complex relationship between those entities. Typical knowledge graphs are very vast and extensive having millions of concepts and their relationships with each other. The massive scale of a knowledge graph ensures to attain generalization between two concepts. The potential scope of generalization extends to the level which also enables us to identify an indirect relationship between concepts i.e. the relation between two concepts without any direct connection between them in a knowledge graph.

Figure 3.14 shows the simplest example of a traffic scene knowledge graph. It consists of 7 entities connected to each other with relations like drives, contains, walks, etc. Some entities have a direct relationship like “Pedestrian” has a “walks on” relationship with the entity “Sidewalk” hence representing that a pedestrian walks on the sidewalk. On the other hand, there are some entities which are not

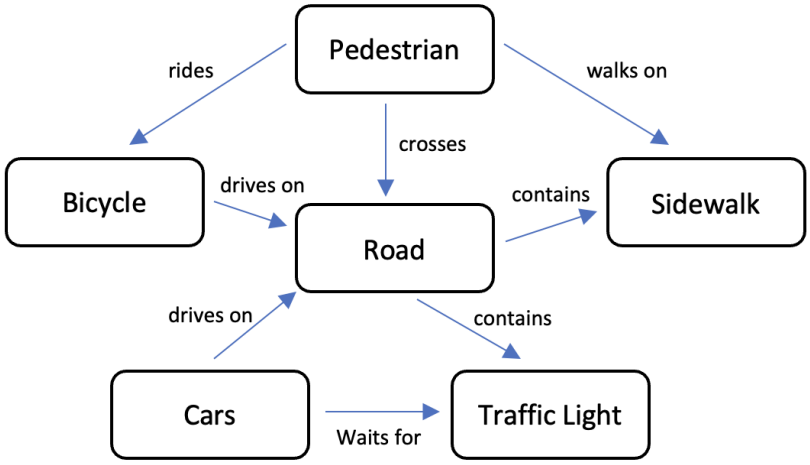


Figure 3.14: A Traffic Scene Knowledge Graph

directly connected and therefore have an indirect relationship. For instance, the entity "Pedestrian" has a "Crosses" relationship with the entity "Road". And the entity "Road" has a further relation with another entity "Traffic Light". Therefore, there exists an indirect relationship between "Pedestrian" and "Traffic Light". Such relationships are very challenging to capture if we only use the images as our only source of information. Such phenomenon is termed as "Semantic Consistency" between two concepts. Semantic consistency will be high between two given concepts if they are more likely to appear together in an image. In contrast, the semantic consistency between two concepts will be low if they are less likely to appear together in an image.

In raw images, we can identify relations between detected objects and associate entities with each other to a limited extent. Generally, the concept of Co-occurrence is given significant attention for defining a relationship between two detected objects in an image. The term "Co-occurrence" refers to the idea that if two objects or entities appear in a set of images then there seems to be a correlation between two entities and consequently have a relation among them. Defining a prior from a set of co-occurring objects is not enough to achieve generalization as it will only be able to relate two entities if there were any seen examples in the training set. Therefore, the use of co-occurrence-based knowledge incorporation does not prove to be suitable for our purpose in this project.

Figure 3.15 shows the overview of our planned pipeline. The overall pipeline for this task is as follows: Firstly, an object detection model is trained on a dataset that contains objects relevant to a traffic scene i.e., pedestrians, traffic lights, other vehicles, etc. Given an input image, the object detection model provides a set of bounding boxes containing an object. Where each bounding

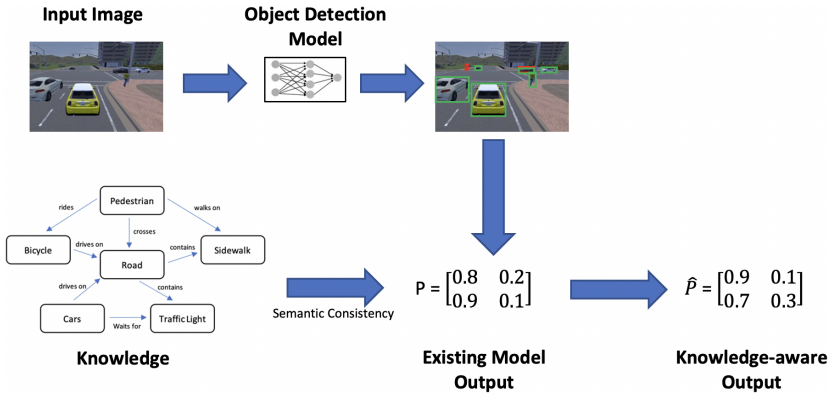


Figure 3.15: An overview of the pipeline

box is assigned an object label depending on the highest probability of that bounding box belonging to a certain class. Instead of picking a label with the highest probability, we use the probability distribution of each box for all possible object labels. We intend to optimize this probability distribution by integrating external knowledge such as a traffic scene knowledge graph.

To quantify the Semantic Consistency between two object labels we will estimate a correlation matrix. Where the co-relation between two object labels is determined by the technique called "Random Walk with Restart". In this technique, given a graph, we start from a node v_0 and after taking a certain number of steps t towards random neighboring nodes we reach a certain node v_t. Meanwhile, there is a distinct probability α that the next move will teleport us back to the starting point v_0. This technique covers all possible paths from one node to another within a specified number of steps and quantifies them in a single probability score which represents how likely it is to start from node A and reach node B after a distinct number of steps. We estimate this score from each possible combination of nodes in our knowledge graph. The resultant co-relation matrix is termed as semantic consistency between any two object labels. The biggest advantage of using semantic consistency is that it not only encompasses the direct connection among objects but also the indirect connections between two objects. Additionally, it considers multiple possible paths between two objects, therefore, making this approach more robust. Finally, both the semantic consistency and probability distribution provided by the object detection approach are then used to optimize the probability distribution to potentially change the labels of some detected objects based on the semantic consistency scores obtained by incorporating traffic scene knowledge from the knowledge graph.

3.4.6 Demonstration Concept

For knowledge integration on visual primitives, we plan to deploy our selected architecture in a Docker container. The Docker container serves as a dedicated environment with all dependencies correctly configured. Dependencies are stated clearly in the requirements file of the Docker container. Our deployment will expect a series of video frames as input to the system. The output of our system will consist of the information regarding detected objects like bounding box coordinates and their respective labels. Our module will enable the demonstrator system to detect pedestrians and some other objects in a given traffic scene by using the provided images and integrating the external traffic scene knowledge.

3.5 DLR - Hybrid-Observable Machine Learning Architecture for Automotive Scene Understanding

3.5.1 Motivation

Driving a vehicle is a complex task, which requires understanding of the situation for reliable decision making. Addressing this complex task, the automated driving systems must maintain high safety performance requiring environment perception to assess the current driving situation, followed by robust driving scene understanding and vehicle control decision making. All these sub-tasks are highly interconnected, e.g. bad perception will usually cause a bad situation understanding and therefore bad decision making. Bad decision making can negatively influence the driving situation and then overload the perception. Implementation of such highly interconnected task mechanisms requires a well-designed technical architecture.

In the first cycle of the project, we are focusing our work on developing a concept on an architecture, specifically on two parts: a) hybrid-deep learning (H-DL) models for scene understanding and decision making to predict lane-change behaviors, and b) gaining the human interpretability of the training procedure. The H-DL provides necessary mechanisms to integrate world and traffic norms with conventional deep learning for better decisions making and efficient learning on down stream task, and the human interpretability of the training procedure provides, observable information to alter the training process. In the next sections the architecture concept as well as the focused parts of it are presented.

3.5.2 Architecture Concept

Connectionist system paradigms, such as deep learning, have shown capacities to asymptotically learn required function using a large number of labeled data samples. In recent times, these systems have out shadowed humans in solving

many complex tasks. However, these systems still suffer from a lack of interpretability due to their black-box nature. Furthermore, they relay on large labeled data samples from the domain on which they are trained and tested for. Deep learning models learn features hierarchically without human intervention, and these learned features remain in sub-symbolic and is non-comprehensive for humans.

On the other side of the spectrum in the area of artificial intelligence there is Symbolic-AI, which is derived based on models designed with prior knowledge of the domain to be addressed. In general, these models are based on rules, logic and relational modeling, and they require very few data samples and certain level of expertise to model the system. Symbolic-AI systems hold declarative semantics and represent domain knowledge in a human-interpretable fashion, e.g. using high-level concepts. To this end, within our scope of the work, novel architectures for "Hybrid-AI" that uses the best of both worlds (connectionist and symbolic systems) shall be researched towards scene understanding for autonomous driving scenarios. Explicit automotive domain knowledge represented within symbolic paradigms, such as knowledge graphs, proportional logic, ontologies etc. are exploited for integration with connectionist models. In figure 3.16, our proposition of a high level concept architecture for developing Hybrid-Observable Machine Learning Architecture for autonomous scene and situation understanding is illustrated.

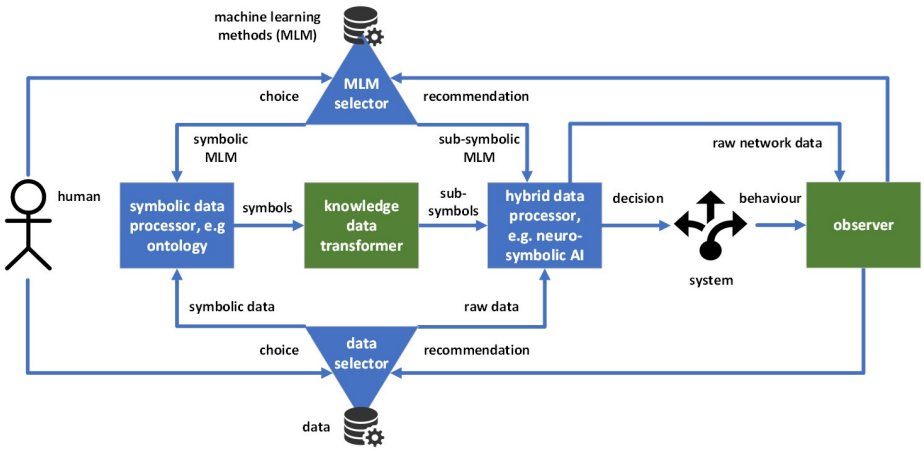


Figure 3.16: Concept architectures for learning Hybrid-AI models.

The human developer on the left side of the model usually chooses data and methods for designing knowledge models for the symbolic-data-processor and for training the neural network within the hybrid-data-processor. The symbolic-data-processor contains formalism, which use high-level concepts or symbols, e.g. ontology, knowledge graphs etc.

The knowledge-data-transformer module derives either loss constraints, ex-

tends training data by a-prior knowledge or captures implicit knowledge (extracted from the training samples) through the previous modules. In that manner, the symbolically represented knowledge from the previous modules can be integrated with connectionist models within the hybrid-data-processor.

The training process within the hybrid-data-processor as well as the later generated behavior after the training can be observed within the observer module. Being interconnected with the human developer by a frontend, this module can give feedback to the developer about the efficiency of the training progress. Therefore, it can influence developer's decisions for choosing the amount of data and training time and therefore optimize the overall system performance.

3.5.3 Data Transformation and Integration

The symbolically represented knowledge from the symbolic-data-processor module can be transformed to the sub-symbolic level and be integrated either for vertical or horizontal integration as shown in figures 3.18 and 3.17.

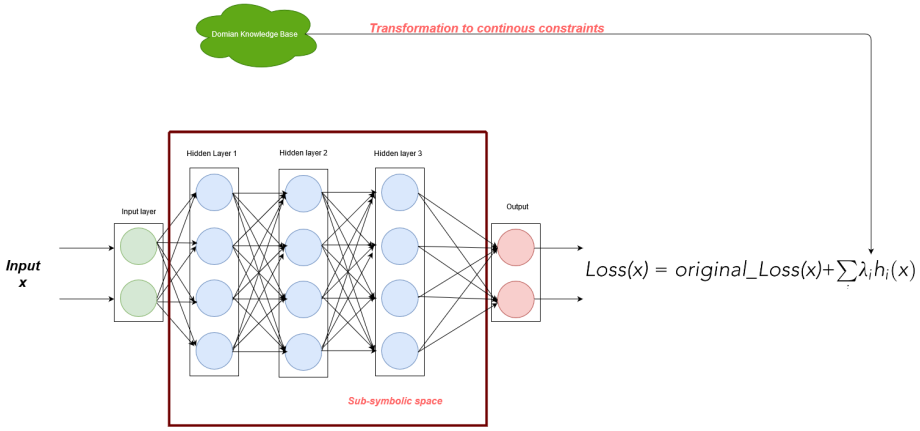


Figure 3.17: Horizontal integration of symbolic knowledge. On an abstract level, a-priori information is used to derive additional loss terms for the neural network to optimize for the required down-stream task.

Most state-of-art DL based approaches [162] [163] [8], achieve significant accuracies when trained on large scale labeled data-sets like ImageNet [164], COCO [165]. Whereas, when they are trained on labeled data which are relatively small compared to large scale data-sets, the performance of the models drops substantially. Especially, when the domain categories (e.g: Traffic signs) share similar appearances as in figure 3.5. The long-tail problem is very common for traffic signs due to class ambiguities, tiny-size objects, and unbalanced object instances in the training data-sets. However, humans can easily identify traffic sign categories, that exhibit strong overlap with similarities and class ambiguities because of the remarkable reasoning ability to scene level commonsense knowledge.

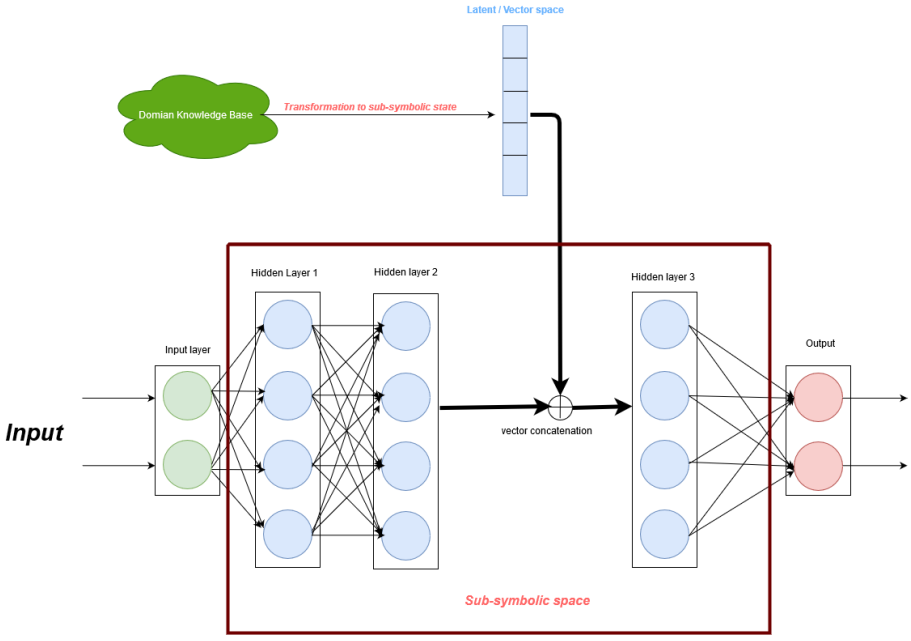


Figure 3.18: Vertical integration of symbolic knowledge. On an abstract level, a-priori knowledge is transformed into embeddings and is distilled into deep neural networks/artificial neural networks (e.g. by concatenating the information from a-priori model with certain layer feature of neural network).

This inspired us to explore the minimal possibility to integrate commonsense knowledge of class relationships in the traffic scene into existing DL based object detector. We make use of object detector paradigms that treat object bounding box regression and classification of each region proposal separately as in [162]. Within [166], the author exploited three different object commonsense knowledge a) shared attribute knowledge; b) pairwise relationship knowledge; c) spatial layout. Inspired from this work, we consider "pairwise relationship knowledge" for integrating with Faster-RCNN [162] object detector pipeline. Wherein, the specific knowledge module in the object detector pipeline learns adaptive context connections for each pairwise regions by using "pairwise relationship knowledge" graph as external supervision. The pairwise relationship knowledge is statically observed from the training data samples which is inherent and implicitly hidden. As shown in figure 3.19, the "Explicit pairwise relationship" module is responsible to learn adaptive region-to-region pairwise relationship knowledge without being explicitly summarized by the human.

In generic region based detectors such as FasterRCNN, Region-of-Interest (ROI) pooling layer extracts visual region features on input images for those regions proposed by region-proposal-network (RPN). These extracted visual

Table 3.5: Visual similarity of classes in traffic signs [161]

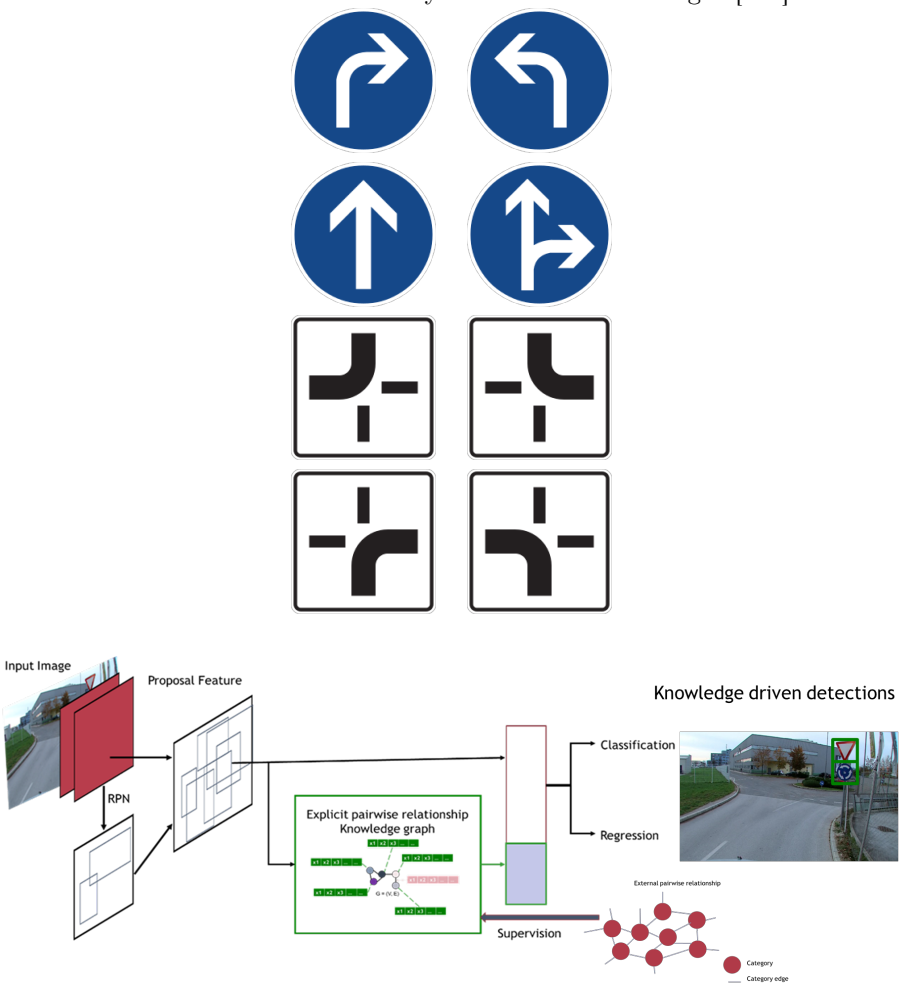


Figure 3.19: Overview of Knowledge directed traffic sign detection.

region features are further feed into detection and classification layers for category specific classification and regression. However, in our approach, we enrich the extracted visual region features before directly feeding it to the detection and classification layers of the network using "Explicit knowledge" module as illustrated in figure 3.19 . The knowledge module learns adaptive region-to-region undirected graph $\hat{G} : \hat{G} =< N, \hat{\varepsilon} >$, where N defines the region nodes and $e_{i,j} \in \varepsilon$ defines the pairwise relationship (i.e co-occurance relationship) between two nodes i, j. Enhanced features from the knowledge module are concatenated with visual region features to obtain enriched features, and later fed into the

bounding-box regression layer and classification layer to obtain detection results. Mathematical formulation of the extended knowledge module with FasterRCNN object is discussed in the following subsection.

Explicit Relationship Knowledge

Figure 3.20: "Round-about" traffic scene.

A primitive type of commonsense knowledge that could be exploited in the traffic scene to detection traffic signs is the pairwise co-occurance relationship. As shown in figure 3.20, the "yield" sign appears at the "round-about", therefore the "yield" sings co-occur with high frequency with respect to "round-about" sign . Such relationship knowledge can be extracted by statistically observing the labeled training samples, therefore giving ε for supervising the learned graph $\hat{G}$.

Adaptive region-to-region graph. We define $\hat{G} < \mathcal{N}, \hat{\mathcal{E}} >$ for all $N_r = \mid N \mid$ region proposals, similarly to the formulation given in [166] and as depicted in figure 3.21 . Where $\mathcal{N}$, are region proposal nodes and $e_{i,j} \in \hat{\mathcal{E}}$ is learnt graph edge for each pair of region nodes.The visual region features $\mathbf{f} = \{f_i\}_{i=1}^{N_r}, f_i \in \Re^D$ of D dimension are extracted from the backbone network of FasterRCNN. With the help of external relationship knowledge graph $\mathbf{Q}$ (i.e co-occurance relational graph which is derived from training data), each edge between two regions $\hat{e_{i,j}}$ is learned by a stacked Multi-layer Perceptron (MLP) which is given as:

$$\hat{e_{ij}} = MLP_Q(\alpha(f_i, f_j)) \tag{3.11}$$

where $\alpha()$, is the L1 pairwise difference between each region pair (f_i, f_j).

Furthermore, given the co-occurance knowledge graph $\mathbf{Q}$, MLP_Q is parameterized with W_Q to generate $\hat{G}$. MLP_Q is learned by directly enforcing the predicted $\hat{e_{ij}}$ to be consistent with the edge weights of a prior graph $\mathbf{Q}$. The loss function of learned edge weights $\{\hat{e_{ij}}\}$ for all N_r region proposals is defined as:

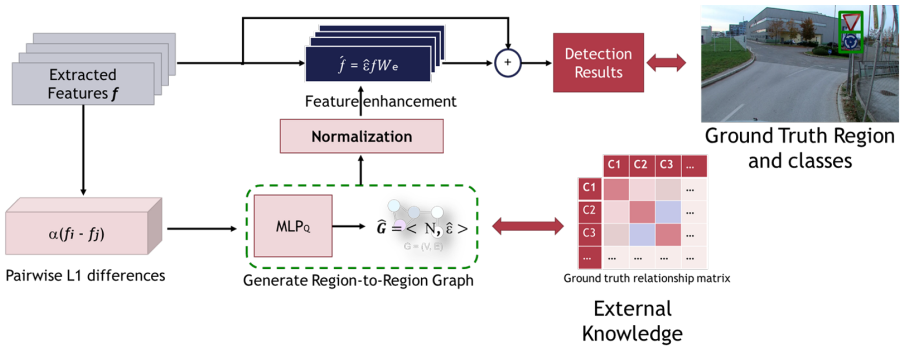


Figure 3.21: Explicit knowledge module.

$$\mathcal{L}(f, W_Q, Q) = \sum_{i=1}^{N_r} \sum_{j=1}^{N_r} 1/2(\hat{e_{ij}} - e_{ij})^2 \tag{3.12}$$

Feature enhancement: features of connected regions i.e. $\hat{\varepsilon} = \{\hat{e}_{ij}\}$ are propagated to enhance each region features $f\prime$ by the following equation:

$$f\prime = \hat{\varepsilon} f W_e \tag{3.13}$$

where, W_e is a transformation weight matrix and $f\prime$ is enhanced features resulted by multiplying learned connected edges. The enhanced region features $f\prime$ are then concatenated with original region features f to obtain the enriched features embedded with pair-wise co-occurrence knowledge. In contrast to generic FasterRCNN pipeline, the resulting enriched features $(f\prime \oplus f)$ are feed into the classification and regression layers to perform detection, rather than the plain features (f).

Therefore, the hybrid network is guided by external knowledge graph $\mathbf{Q}$, to learn context of pairwise object co-occurrence for improving the traffic sign detections.

3.5.4 Network Observer

The developer on the left side of the proposed concept for the Hybrid-Observable architecture 3.16 can be supported by the Network Observer module in taking important decisions. The developer usually chooses in advance data and methods for designing the necessary knowledge models and for training the AI-networks. These decisions are usually based on experience of the developer, which can be high or low, as well as on developer's preferences, which can be correct or wrong. These more qualitative approaches can decrease the efficiency of the training process as well as the overall system performance after the training. Therefore, it

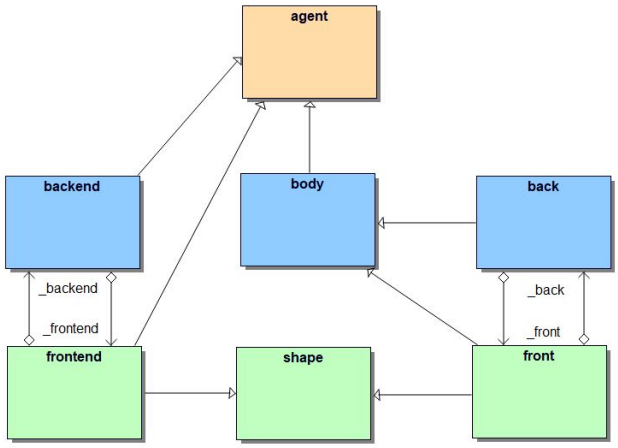


Figure 3.22: Class structure of DLR network observer module.

is highly recommendable to support this qualitative decision process with more objective approaches.

The common objective approach is to use measurements and statistics while training or usage of the trained system. As the training itself is mostly non-transparent, this is often a difficult task to decide about the needed amount of data and time for model training. The developer can observe black-box-like the statistical failure rate in the output of the trained network and the system behavior depending on data fed to it. The developer can define the acceptable failure or success rates and interrupt training or even system usage when these rates are reached or not. This approach, however, can be further optimized.

The training process as well as the later system behavior can be observed by an software module. Being interconnected with the human developer by a frontend, this observer module can give feedback to the developer about the efficiency of the training progress or the system usage. Therefore, it can influence the developer's decisions for choosing the amount of data and training time and therefore optimize the overall system performance.

In the figure 3.22 the basic implementation of the DLR neural network observer is shown. It consists of a backend (blue level) calculating all necessary states and other relevant observation data of the neural network in an training environment. On the frontend (green level) the observed information of the training neural network is presented to the human developer. The backend includes a fair amount of graph search, analysis and manipulation functions, which can be used to calculate different data constellations for network observation. Both, backend and frontend objects are using an agent based implementation (orange level).

In the figure 3.23 an example of how the frontend can be presented to the

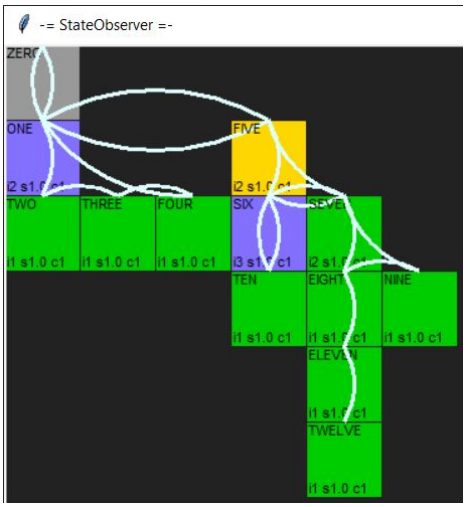


Figure 3.23: Example of a small net observed by the DLR-network observer.

developer is shown. Depicted is a simple 13-node-network with 19 edges. The positioning of nodes (squares) and edges (arcs) shows the hierarchical structure of the network. The positions of these objects can be changed to reach a better observable object distribution. The depicted network has a certain dynamic behavior, which can be shown by color changes for nodes and edges. The labels of nodes as well as another observation parameters are also presented by letters, signs and numbers.

3.5.5 Demonstration Concept

As pointed out earlier, within the first cycle of the project we focused our research on primitive commonsense knowledge of objects to incorporate with standard DL based object detectors for traffic sing detection. Whereas, within the overall scope of the project we intend to demonstrate comprehensive scene understanding for predicting lane change behavior of an autonomous vehicle, through exploring other scene level relationships and rules associated to objects and its traffic environment. As depicted in figure 3.24, DLR's "hybrid-AI" module and Carla simulation framework are working within Docker. The communication between the modules in the docker is via ROS-Bridge. Input to the Hybrid-AI module is the sensor data (i.e RGB image) and environment information (HD map, Dynamic object list, scene traffic signs and light state) from the Carla simulation. The Hybrid-AI outputs way-points/trajectory for autonomous driver (AD) agent. The predicted trajectory for the AD-agent is communicated to the AVL AD stack to derive the motion control signals to control the AD-agent back in Carla simulation. ROS interfaces defined in TP 4.2 are used to communicate between DLR's Hybrid-AI module and the Carla simulation. The communication with

external AVL-AD stack from Docker container is via the AVL client wrapper agreed in TP 4.1. In figure 3.24, a tentative concept is presented, and is subject to possible changes in the project cycle 2.

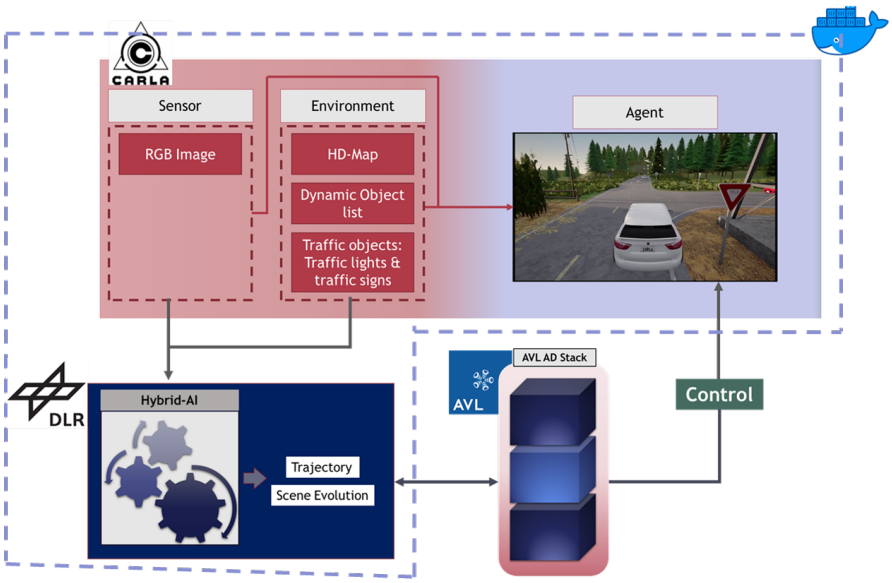


Figure 3.24: DLR Demonstration concept.

3.6 FhG Fokus and FZI - Architectures that Apply Legal Knowledge

3.6.1 Motivation

Autonomous vehicles can benefit from knowing the relevant traffic regulations that apply to the current situation or planned actions. The plain knowledge of traffic regulations is not sufficient, but the vehicle must be able to reason on the legal knowledge with respect to a traffic situation of interest (current or anticipated). The vehicle must be able to calculate an answer to the questions: "Am I currently compliant with the law?" and "Which possible actions guarantee that I will stay compliant?". Furthermore, real life traffic situations sometimes require to rank contradicting regulations and let one rule override another to make controlled rule exceptions. Moreover, this information can be employed during the training of a reinforcement autonomous driving agent or single AI modules e.g. a planning module for the next action by using the compliance check in a semantic loss/gain function.

Legal knowledge can be represented as logic rules using an adequate, machine

understandable logic formalism. Reasoning engines can then be applied to query this legal knowledge base with respect to a traffic situation. However, evaluating queries against a large knowledge base can become computationally very complex i.e. querying can become very slow or even impossible. This is certainly unsuitable for real-time planning in ADVs and possibly too costly for training purposes. Architectures that either employ DL-based reasoning for faster inference or that reduce the computational complexity to a propositional level might provide an appropriate solution.

3.6.2 State of the Art

The aforementioned issues pose a serious problem with respect to the homologation of autonomous vehicles, which are expected to ensure passenger safety and follow traffic rules at a level at least equivalent to human drivers. It therefore seems necessary to endow autonomous driving systems with an awareness of the rules of the road in a way that allows them to reason about their actions in an informed and strongly generalizing manner.

On the one hand, it is possible to train driving networks in a bayesian manner by putting a suitable prior on the model. This typically yields a regularized optimization problem [167]. Examples of this approach in the context of motion planning and prediction include [168], which adds specific loss terms for collisions, driving off-road, and not following the planned route. Similar loss terms are also added in [169], [170] for red lights and collisions. In [171], the model is trained on a non-differentiable loss function using a policy gradient method from reinforcement learning. At test time, however, rule conformity cannot be guaranteed.

A possible way to add rule conformity is given in [172]. A differentiable logic layer is proposed that corrects a predictor's decisions for compliance with temporal logic rules. Using backpropagation, it is possible to both learn rule parameters and adjust the underlying predictor's weights to better satisfy the given rules.

Additionally, methods that integrate reasoning into the prediction and planning process itself have been proposed, which are briefly surveyed in the following.

Automated Reasoning

Classical Reasoners. Classical reasoners use an algorithm that encodes a reasoning strategy as opposed to other approaches that learn the reasoning strategy and steps using ML techniques. Prominent methods are DPLL which is part of almost any SAT-solver (solves propositional problems) in some variant, refutation-based and tableaux proofing methods especially for first-order logics.

Graph Neural Networks (GNNs). In [173] multiple approaches were presented to integrate symbolic systems in Graph Neural Networks (GNNs). GNNs allow for two major advantages in solving reasoning tasks. They apply an inductive bias directly through their architecture and offer permutation

invariance because of their update and aggregation functions. Permutation invariance simplifies the representation of literals and clauses. Therefore the order of logical symbols does not impact the learning and understanding of such clauses. For example the GNN handles the logical expression $(x_1 \vee \neg x_2 \vee x_3)$ semantically the same as the expression $(x_1 \vee x_3 \vee \neg x_2)$. GNNs enable visual scene understanding and reasoning superior to Convolutional Neural Networks as shown in [174].

Logic Tensor Networks (LTNs). Tensorisation of first-order logic is another approach for solving reasoning tasks utilizing Deep Learning in combination with neural-symbolic integration. Logic Tensor Networks (LTNs) as presented in [175] are able to use full first-order logic with function symbols by embedding these logic symbols into real-valued tensors. They propose a neural-symbolic formalism called Real Logic in addition to the computational model that is designed for defining logical expressions suited for tensorisation in LTNs.

Real Logic is a many-valued, end-to-end differentiable first-order logic. It consists of sets of constant, functional, relational and variable symbols. Formulas built from these symbols can be partially true and therefore Real Logic includes fuzzy semantics. Constants, functions and predicates can also be of different types represented by domain symbols. The logic also includes connectives $\diamond \in \{\neg\}, \circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$ and quantifiers $Q \in \{\forall, \exists\}$. Semantically Real Logic interprets every constant, variable and term as a tensor of real values and every function and predicate as real function or tensor operation. Therefore Logic Tensor Networks are able to efficiently compute an approximate satisfiability by mapping logical expressions to real-valued tensors.

Model-Based Planning

Signal temporal logic (STL) [176] is a popular way to express traffic rules [177], [178], [179]. Cho et al. propose to incorporate STL rules as constraints into a model-predictive controller [70], [71]. The rules are hand-crafted by an expert and are associated with slackness values that quantify the amount of tolerance. The authors propose to use Gaussian Process Regression to adapt rule slacknesses to the current situation.

Esterle et al. [179] propose to express traffic rules in linear temporal logic (LTL) and to convert these rules to finite state automata, called "monitors". This makes it possible to check rule conformity along the trajectory in real-time. Monitors for basic traffic rules in highway environments have open-sourced under the BARK simulator [180]. The advantage of monitors is that their state can be added to the global system state, enabling tree search methods like AlphaZero [181].

Tree search methods can be readily combined with model learning methods like [182], which considers highway driving. The authors propose to learn a video prediction model of the environment and to use a optimization-based MPC scheme with additional costs for safety distances and lane position.

Another model learning method is proposed in MATS [183], which takes a graph-based system representation and predicts the coupled system dynamics as a mixture of affine time-varying systems.

Sampling

In [184], [185], a convolutional network is trained to predict intermediate semantic representations like time-varying semantic grid maps and drivable areas. These intermediate representations can be used to perform sampling based planning using predetermined cost functions. The parameters of these cost functions can be set manually or learned during the training phase. Similar methods are presented in [186], [187].

Probabilistic Inference

Knowledge graphs are a widely used in Natural Language Processing to add background knowledge to language models. Similarly, some motion planning methods use graphs as an environment representation [123], [188], [189]. There has also been recent work on integration with motion forecasting of general dynamical systems [190], [191] and road traffic specifically [192]. Graph-based approaches are interesting for their ability to incorporate probabilistic reasoning.

In [193], [194], a structured environment representation is used to augment prediction networks with a Markov Random Field as a probabilistic reasoning layer, allowing interactive reasoning over multiple actors and rules. The rules can be given by hand; their parameters can additionally be learned from data since a differentiable belief propagation algorithm is used for inference.

Another probabilistic approach is presented in [122] and [195], which frame prediction as joint inference over goals and trajectories. The joint probability factorizes into a goal prediction and a goal-conditioned trajectory prediction. This method is interesting for its ability to incorporate rules into the prior over goals given the current situation.

3.6.3 Description of the Architecture Concept

Idea of Architecture I and II. The goal of the architecture is employing expert knowledge to assess whether the ADV is in a desirable state or if a (planned) action of the ADV or other traffic participants are desirable. More concretely, the expert knowledge will be formalized traffic rules from AP1.4 and an ontology capable of describing the aspects of a traffic scene. Such ontology consists of two parts: A TBox that describes the abstract concepts (e.g. vehicle), attributes (e.g. velocity) and relations (e.g. is behind) of parts of the traffic scene. The ABox introduces actual objects of the traffic scene and connects it to the TBox e.g. there exists this specific object which is a vehicle, has velocity 12 km/h and is behind this other specific object. This information will be partly taken from other AI modules if available (however for the evaluation we rely on ground truth provided by the simulation environment) and partly inferred by

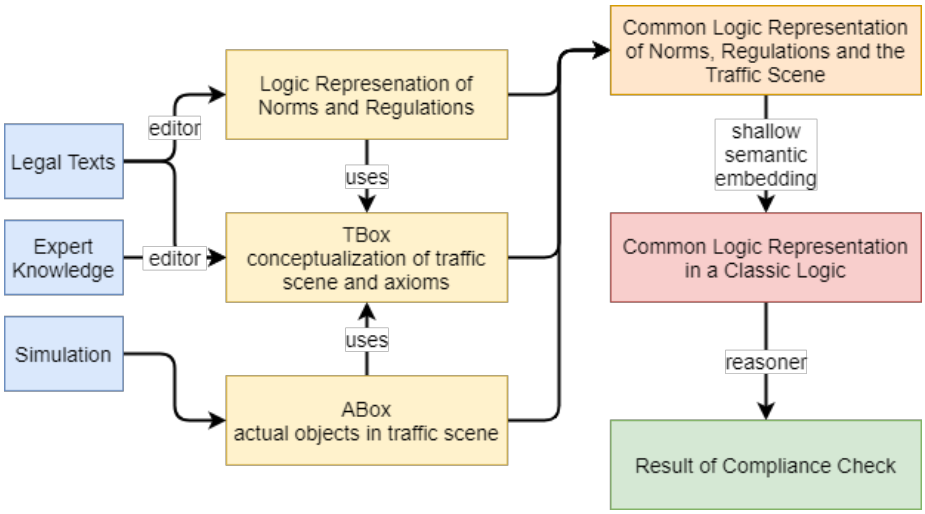


Figure 3.25: From knowledge to compliance check.

predefined rules (axioms, expert knowledge) e.g. for two vehicles on the same lane with a distance of less the 100m one is behind the other. The desirable state or (planned) action of a traffic participant (which is part of the traffic scene) will be judged according to the formalized traffic rules to determine whether this state or (planned) action is legal or illegal. This judgment will be formulated as a reasoning task on the formalized traffic rules and traffic scenery. The (planned) action will be provided by the AD-stack.

Architecture I. Figure 3.25 illustrates the reasoning pipeline from formalized (legal) knowledge and the traffic scene to a compliance check. The input (yellow) of the architecture comprises a (normative) logic representation of legal knowledge (1a), the TBox of an ontology of which the representation of legal knowledge make use of its concepts and an ABox that describes the traffic scene employing the TBox of the ontology. The ABox and TBox of the ontology are expected to be modelled with description logics which has to be mapped onto the normative logic since the normative logic is used to model the legal knowledge. This results in a normative logic description of all knowledge (orange) that could be processed by a reasoner for the specific normative logic employed here tasked with the compliance check. Unfortunately, for most normative logics reasoning systems do not exist. However, classical logics do have a wide variety of reasoning software and they can be reused for reasoning with normative logics applying the shallow semantic embedding approach [196]. This approach remodels the description of the knowledge in a normative logic as a classical logic description (red), that can be processed by a reasoner for classic logic.

Architecture II. Architecture I can be modified at such that the the normative logic description of all knowledge is simplified by eliminating the quantifiers by

instantiating all quantified rules with their related entities that actually occur in the description (i.e. especially in the traffic scene). This process leads to a quantifier-free knowledge description thus reducing the computational complexity to a NP-complete problem. After the applying the semantic embedding approach the resulting classical logic description will be propositional hence any query could be solved by a SAT solver. The theoretical feasibility of this approach has yet to be assessed.

3.6.4 Demonstration Concept

Architecture I and II. Within architecture I different reasoners will be evaluated with respect to their performance (number of solved reasoning tasks, running time) while checking compliance of the traffic situation / (planned) action with the formalized regulations and also compared to the performance of architecture II and appropriate SAT solver(s) if this approach proves to be valid.
Motion Planning Architecture. In the course of this project, a motion planning pipeline shall be developed that operates on a structured environment representation and actively takes rules and exceptions into account when reasoning about trajectory plans. The system is expected to work off the outputs of an object detection and vehicle localization stage, taking into account environment information possibly given in the form of an HD map, and to produce interactive predictions of other participants, as well as goal-conditioned trajectory plans for the ego vehicle (to be processed by a low-level trajectory controller). The system should demonstrate cooperative planning capabilities in dense traffic and under rule conflicts and shall be evaluated on use cases in highway and urban driving.

3.7 FZI TKS - Application of a Rule Engine

3.7.1 Introduction

In recent years several different strategies on how to tackle the control problem of self-driving vehicles have been developed. The most prominent are classical planning using iterative planning algorithms and imitation learning. While classical planning is limited by the complexity of designing driving strategies by hand, imitation learning can potentially learn driving based on the experience of human drivers. However, a problem with imitation learning is that the performance of the driving policy is limited by the quality of the collected data. Especially rare situations will be underrepresented. A third approach that can mitigate these problems is reinforcement learning. An agent is trained to drive such that it maximizes that mean cumulative reward given by a reward function. An advantage in comparison to classical planning and imitation learning is that the agent can discover unseen situations while exploring the environment and can learn to act optimally after some time. While in theory such an approach could lead to an optimal driving policy a well-designed reward function is crucial to enforce legal driving maneuvers and efficient policy learning. In this work,

we focus on the latter by extending an existing model-based agent with human interpretable environment understanding and prediction. This allows to integrate a rule engine that is developed in AP1.1. The main idea is that the rule engine can either be used to directly find optimal driving trajectories or to distill the behavior specified in the according rule book into a neural driving policy. Furthermore, we plan to integrate knowledge about the scene structure directly into the model by enforcing a graph-structured latent space for forward prediction instead of a vector space. We hope that a structured latent scene representation will emerge from this constraint.

3.7.2 Related Work

The usage of a rule engine is inspired by [197]. They show that a rule graph can be used to effectively choose the correct behaviour in a given situation. Especially when multiple rules compete the approach allows to prioritize. The rule engine is implemented in AP1.1, however a necessary requirement, namely an interpretable description of the current scene, is implemented in this AP. Our model-based agent is based on Dreamer [38, 198], a recent approach to continuous control problems with high-dimensional observations. In contrast to [199], we train our model with prediction targets, such as the road topology, instead of reconstruction as reconstruction makes poor use of model capacity. Additionally, we have strong prior knowledge on what is important for autonomous driving so that we can be more specific than reconstruction. In recent research on end-to-end autonomous driving [200, 201], auxiliary targets are commonly used for this purpose. They provide an additional learning signal for which we are sure that the encoded information is relevant for the control problem and that aids convergence. Another prior information that we plan to integrate into our architecture is that a typical scene in autonomous driving consists of multiple actors that interact with each other. Recent works have used graph neural networks [202] or transformers [121] to model the relationship between traffic actors and include it into the learning process [124, 123]. However, these approaches are not end-to-end in that they require a preceding perception pipeline. We instead aim to use a graph-structure latent space in our model that is learned directly from sensor observations. Our approach is related to [191], but instead of using contrastive learning, we train our model by prediction, which has the benefit of being human and rule-engine interpretable.

3.7.3 Architecture

We mainly focus on designing an efficient model that is able to predict the future development of the environment directly from sensor data. We assume that the world can be modeled as a partially observable Markov decision process (POMDP). Thus, we use a stochastic sequential latent variable model that takes as input the sensor observations at each time-step and predicts the current and future developments of the scene with respect to the agents own behavior.

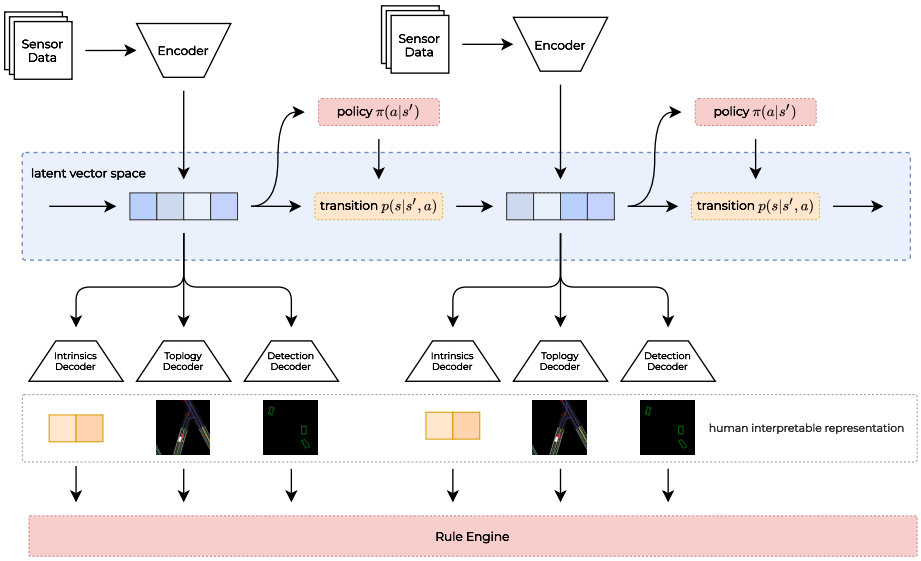


Figure 3.26: A simplified representation of our proposed architecture to generate a scene representation that can be interpreted by a rule-engine. The sensor observations are encoded by a CNN at each time-step. The resulting feature vector and the history of past latent states is used to infer the current latent state. From the latent state relevant information that is commonly used as input for a driving policy can be decoded. In our case, these information is fed into a rule engine that rates the driving behavior. The latent transition model allows to rollout future trajectories which can be used to plan future trajectories in accordance with a given rule-book.

The overall architecture (see figure 3.26) can be divided into three parts: encoder, stochastic latent transition model and decoder heads. The encoder takes as input the sensor data and outputs a vector that contains relevant information of the input data. In our case we use a CNN (e.g. ResNet-18) to analyze rgb front camera and birds-eye-view (BEV) lidar data. In practice a suitable neural network for each modality or a combination of sensors can be chosen, but we use a small CNN for ease of implementation and reduced training time. The transition function is based on recurrent stochastic state space models and trained via an amortized variational inference scheme. At training time a posterior and prior distribution over latent states are learned. The prior is then used to train a policy purely in simulated rollouts of the future. At inference time the posterior is used to infer the current state based on the past and current observation based on which the policy then outputs the next action. The decoder provides the training signal by learning to predict values of interest which is at minimum the reward that is returned by entering a state. Often sensor observation reconstruction is used as an additional auxiliary loss

to provide a stronger learning signal. However, in autonomous driving sensor observations can be of massive size and additionally contain mostly information that is not necessary for the task of driving. For this reason we use targets from which we know that they contain information that at least intuitively can help humans to drive such as road topology or semantic maps.

3.7.4 Human interpretable outputs to aid rule specification

A rule engine such as developed in AP 1.1 works based on a rule book that mostly contains human-designed rules. These rules are based on high-level attributes of the scene. For example one of the most important rules should be to avoid crashes into other traffic actors. Such a rule can be specified as keeping the distance to other traffic actors greater than zero. Additionally, to aid learning one could specify the no crash rule to penalize getting into close distance to other actors. It is obvious that we need a representation of environment that can be generated by a learning algorithm, but is also human interpretable. In our case, we aim to specify rules based on a graph-structure that includes traffic actors and road topology.

However, in our base architecture the latent space is vector-shaped. This means that for each time-step we have to decode the latent state into the graph structure that we need to run the rule engine. We suggest to add decoder heads that are similar to the heads of recent object detection approaches. In particular, we plan to implement a CenterNet-like head that outputs a heat-map over the center points of objects in the scene and plan to later extend this to also predict road topology in graph space. In addition to providing an human- and rule-engine-interpretable state representation, we also expect a better latent state representation since we argue that human-interpretable high-level descriptions of the scene should contain most of the relevant information for driving successfully.

3.7.5 Implicit integration of scene structure into forward prediction

The main assumption driving our approach is that human-interpretable features often contain the information necessary for correct driving and furthermore that it is possible to plan mostly based on these features. The question arises whether it is possible to integrate the knowledge about the existence of such high-level features into the architecture of our stochastic sequential latent variable model. As a first step in this direction we suggest to structure the latent space as a graph. Many current state-of-the-art models already use graph-based input representations to predict for example the future trajectory of traffic actors. In contrast, our model directly takes sensor observations as an input modality. Thus, we first have to generate a set of nodes from sensor data before we can make predictions about the future using these nodes. A simple way to generate such a set would be to take the output of an object detection head and simply generate a node for each object containing key features such as the location.

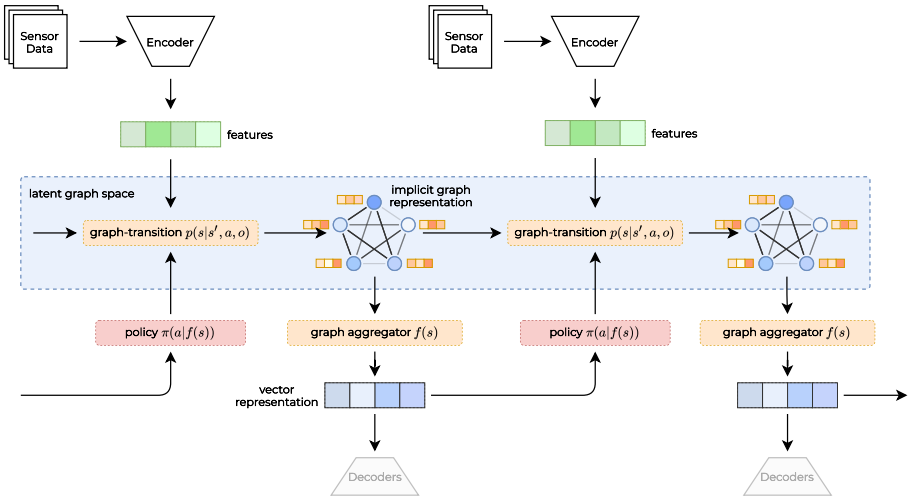


Figure 3.27: A simplified overview on how we plan to implicitly make use of a latent graph structure. The sensor observations are encoded into a compact feature representation. The graph-transition function takes the previous latent graph state and the feature representation as an input and updates each node in the graph to the current state. Each node in the graph is represented by a feature vector that is concatenated with a learnable identification vector, so that similar nodes can be tracked along time-steps. A graph-aggregation module is used to summarize the graph into a vector which is then fed into the decoders and the driving policy.

However, such an approach could limit the latent transition model too much and additionally make it impossible for the model to learn latent features from data that have not been specified manually. Instead, we suggest to only provide the latent graph structure and let the model learn which data to fill into each node. The main idea is that each node can decide itself which information to extract from the feature vector given by the observation encoder. Each node also has a learnable identification vector that allows the model to recognize similar nodes over multiple time-steps. Furthermore, multiple sets of weights are learned and a node can decide which weights to use for forward prediction. The intention is to implicitly encourage different kinds of nodes in the latent representation (comparable to how there are different actors in a scene, e.g. pedestrians, cyclists and car drivers). Finally, a graph aggregator module is used to summarize the graph back into a vector representation that can be used by the driving policy to infer the next action. The action is then fed back into the latent graph to predict the next graph state. As we use a fully connected graph our approach is very related to self-attention as known from transformer architectures.

In addition to the basic latent graph representation, we are going to explore

training regimes that encourage distribution of information in different nodes and similar information in the same nodes over multiple time-steps. We hope that the model will be regularized to make extensive use of the graph structure and to not collapse into a vector-based form.

3.7.6 Evaluation

Since our overall goal is not to predict key features such as object locations or lane marking, but to infer and predict a compact, but informative scene description to a driving policy, we plan to evaluate our approach in specific driving scenarios where we expect the agent to perform poorly without inclusion of prior knowledge. For example driving on a straight road is no such situation since plenty of data is available and no frequent interactions between traffic actors occur. In contrast, navigating a crowded intersection or a roundabout will need knowledge about the legal and (driving) cultural specifics. Thus, we will focus on common metrics such as route completion, number of collisions and number of infractions in several scenarios where our base agent performs poorly. We use a custom implemented OpenAI Gym environment that allows to run agents in the CARLA [203] simulator. Additionally, we implement a parallel version so a single agent can collect data in multiple environments backed by multiple CARLA servers. This is necessary as a single CARLA server instance currently only reaches up 10 frames per second when LIDAR and RGB camera sensors are added which makes experimenting prohibitively time-consuming.

3.7.7 Demonstration Concept

We plan to demonstrate the effectiveness of our approach using the CARLA simulator. Reinforcement learning is with the exception of offline reinforcement learning only possible within a closed-loop environment. Traditional datasets do not provide feedback given the actions of the agent an thus, are only usable for imitation learning. We do not expect our approach to generalize from simulation to real-world data without integration of domain adaption methods. However, this is out-of-scope for this work. We plan to investigate whether isolated parts of approach can be trained on real-world data and thus, integrated into the overall demonstrator. More specifically, the model of our model-based agent can be trained on offline-data. The resulting forward prediction model could then be used by other components at inference time, e.g. to generate object lists. Note that the success of the model for real-world data is directly related to the size of the dataset and comes with a high computational demand. We will analyze whether it is feasible to train such a model with our currently available hardware.

References

[1] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 1126–1135. URL: http://proceedings.mlr.press/v70/finn17a.html.

[2] Jake Snell, Kevin Swersky, and Richard S Zemel. "Prototypical networks for few-shot learning". In: *arXiv preprint arXiv:1703.05175* (2017).

[3] Yonggang Li et al. "Differentiable Automatic Data Augmentation". In: *European Conference on Computer Vision*. Springer. 2020, pp. 580–595.

[4] Oriol Vinyals et al. "Matching Networks for One Shot Learning". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee et al. Vol. 29. Curran Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper/2016/file/90e1357833654983612fb05e3ec9148c-Paper.pdf.

[5] Timothy Hospedales et al. *Meta-Learning in Neural Networks: A Survey*. 2020. arXiv: 2004.05439 [cs.LG].

[6] Flood Sung et al. "Learning to compare: Relation network for few-shot learning". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 1199–1208.

[7] Spyros Gidaris et al. "Boosting few-shot visual learning with self-supervision". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 8059–8068.

[8] Joseph Redmon et al. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[9] J. Redmon and A. Farhadi. "YOLO9000: Better, Faster, Stronger". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 6517–6525. DOI: 10.1109/CVPR.2017.690.

[10] Wei Liu et al. "Ssd: Single shot multibox detector". In: *European conference on computer vision*. Springer. 2016, pp. 21–37.

[11] Zhi Tian et al. “Fcos: Fully convolutional one-stage object detection”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 9627–9636.

[12] Tsung-Yi Lin et al. “Focal loss for dense object detection”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2980–2988.

[13] Shaoqing Ren et al. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes et al. Vol. 28. Curran Associates, Inc., 2015.

[14] K. He et al. “Mask R-CNN”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2980–2988. DOI: 10.1109/ICCV.2017.322.

[15] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[16] Nicolas Carion et al. “End-to-end object detection with transformers”. In: *European Conference on Computer Vision*. Springer. 2020, pp. 213–229.

[17] Gongjie Zhang et al. *Meta-DETR: Few-Shot Object Detection via Unified Image-Level Meta-Learning*. 2021. arXiv: 2103.11731 [cs.CV].

[18] Bingyi Kang et al. “Few-Shot Object Detection via Feature Reweighting”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019), pp. 8419–8428.

[19] Xiaopeng Yan et al. “Meta R-CNN: Towards General Solver for Instance-Level Low-Shot Learning”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 9576–9585. DOI: 10.1109/ICCV.2019.00967.

[20] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. “Meta-Learning to Detect Rare Objects”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2019.

[21] Abhishek Gupta et al. “Meta-Reinforcement Learning of Structured Exploration Strategies”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper/2018/file/4de754248c196c85ee4fbdcee89179bd-Paper.pdf.

[22] Fei Ye et al. “Meta Reinforcement Learning-Based Lane Change Strategy for Autonomous Vehicles”. In: *arXiv preprint arXiv:2008.12451* (2020).

[23] Ignasi Clavera et al. “Learning to Adapt in Dynamic, Real-World Environments through Meta-Reinforcement Learning”. In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=HyztsoC5Y7.

[24] Shanshan Zhang, Rodrigo Benenson, and Bernt Schiele. "Citypersons: A diverse dataset for pedestrian detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 3213–3221.

[25] Amir Rasouli, Iuliia Kotseruba, and John K Tsotsos. "Are they going to cross? a benchmark dataset and baseline for pedestrian crosswalk behavior". In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2017, pp. 206–213.

[26] Amir Rasouli, Iuliia Kotseruba, and John K Tsotsos. "Agreeing to cross: How drivers and pedestrians communicate". In: *IEEE Intelligent Vehicles Symposium (IV)*. 2017, pp. 264–269.

[27] Shanshan Zhang, Jian Yang, and Bernt Schiele. "Occluded pedestrian detection through guided attention in cnns". In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2018, pp. 6995–7003.

[28] Yanwei Pang et al. "Mask-guided attention network for occluded pedestrian detection". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 4967–4975.

[29] Lianghua Huang, Xin Zhao, and Kaiqi Huang. "Bridging the gap between detection and tracking: A unified approach". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 3999–4009.

[30] Bo Li et al. "High performance visual tracking with siamese region proposal network". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 8971–8980.

[31] Fisher Yu et al. "Bdd100k: A diverse driving dataset for heterogeneous multitask learning". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 2636–2645.

[32] Marius Cordts et al. "The cityscapes dataset for semantic urban scene understanding". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3213–3223.

[33] Iuliia Kotseruba, Amir Rasouli, and John K Tsotsos. "Joint attention in autonomous driving (JAAD)". In: *arXiv preprint arXiv:1609.04741* (2016).

[34] Eunbyung Park and Alexander C Berg. "Meta-tracker: Fast and robust online adaptation for visual object trackers". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 569–585.

[35] Guangting Wang et al. "Tracking by instance detection: A meta-learning approach". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 6288–6297.

[36] Judea Pearl. *Causality: Models, Reasoning, and Inference*. 2. ed., repr. with corr. Cambridge: Cambridge University Press, 2013. 464 pp. ISBN: 978-0-521-77362-1 978-0-521-89560-6.

[37] Lars Buesing et al. *Woulda, Coulda, Shoulda: Counterfactually-Guided Policy Search*. Version 1. Nov. 15, 2018. arXiv: 1811.06272 [cs, stat]. URL: http://arxiv.org/abs/1811.06272 (visited on 06/09/2021).

[38] Danijar Hafner et al. *Dream to Control: Learning Behaviors by Latent Imagination*. Mar. 17, 2020. arXiv: 1912.01603 [cs]. URL: http://arxiv.org/abs/1912.01603 (visited on 10/25/2021).

[39] Julian Schrittwieser et al. "Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model". In: *Nature* 588.7839 (7839 Dec. 24, 2020), pp. 604–609. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/s41586-020-03051-4. arXiv: 1911.08265. URL: http://arxiv.org/abs/1911.08265 (visited on 06/09/2021).

[40] Danijar Hafner et al. *Mastering Atari with Discrete World Models*. May 3, 2021. arXiv: 2010.02193 [cs, stat]. URL: http://arxiv.org/abs/2010.02193 (visited on 10/25/2021).

[41] Jianyu Chen, Shengbo Eben Li, and Masayoshi Tomizuka. *Interpretable End-to-End Urban Autonomous Driving with Latent Deep Reinforcement Learning*. July 7, 2020. arXiv: 2001.08726 [cs]. URL: http://arxiv.org/abs/2001.08726 (visited on 10/25/2021).

[42] James Kirkpatrick et al. "Overcoming Catastrophic Forgetting in Neural Networks". In: *Proceedings of the National Academy of Sciences* 114.13 (Mar. 28, 2017), pp. 3521–3526. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.1611835114. pmid: 28292907. URL: https://www.pnas.org/content/114/13/3521 (visited on 10/25/2021).

[43] Arslan Chaudhry et al. *Efficient Lifelong Learning with A-GEM*. Jan. 9, 2019. arXiv: 1812.00420 [cs, stat]. URL: http://arxiv.org/abs/1812.00420 (visited on 05/12/2021).

[44] David Lopez-Paz and Marc'Aurelio Ranzato. *Gradient Episodic Memory for Continual Learning*. Nov. 4, 2017. arXiv: 1706.08840 [cs]. URL: http://arxiv.org/abs/1706.08840 (visited on 02/03/2021).

[45] Yunhui Guo et al. *Improved Schemes for Episodic Memory-Based Lifelong Learning*. Dec. 14, 2020. arXiv: 1909.11763 [cs, stat]. URL: http://arxiv.org/abs/1909.11763 (visited on 05/12/2021).

[46] Robert Krajewski et al. "The highD Dataset: A Drone Dataset of Naturalistic Vehicle Trajectories on German Highways for Validation of Highly Automated Driving Systems". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 2118–2125. DOI: 10.1109/ITSC.2018.8569552.

[47] Paul Spannaus, Peter Zechel, and Kilian Lenz. "AUTOMATUM DATA: Drone-based highway dataset for the development and validation of automated driving software for research and commercial applications". In: *2021 32st IEEE Intelligent Vehicles Symposium (IV)*. 2021.

[48] Ramalingam Shanmugam. "Elements of Causal Inference: Foundations and Learning Algorithms". In: *Journal of Statistical Computation and Simulation* 88.16 (16 Nov. 2, 2018), pp. 3248–3248. ISSN: 0094-9655, 1563-5163. DOI: 10/gd6dxt. URL: https://www.tandfonline.com/doi/full/10.1080/00949655.2018.1505197 (visited on 07/15/2020).

[49] Judea Pearl. "The seven tools of causal inference, with reflections on machine learning". en. In: *Communications of the ACM* 62.3 (Feb. 2019). Number: 3, pp. 54–60. ISSN: 00010782. DOI: 10.1145/3241036. URL: http://dl.acm.org/citation.cfm?doid=3314328.3241036 (visited on 09/05/2019).

[50] Alex H. Lang et al. "PointPillars: Fast Encoders for Object Detection from Point Clouds". In: *CoRR* abs/1812.05784 (2018). arXiv: 1812.05784. URL: http://arxiv.org/abs/1812.05784.

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

[52] Holger Caesar et al. "nuScenes: A multimodal dataset for autonomous driving". In: *arXiv preprint arXiv:1903.11027* (2019).

[53] Angel X. Chang et al. *ShapeNet: An Information-Rich 3D Model Repository*. Tech. rep. arXiv:1512.03012 [cs.GR]. Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[54] Laura von Rueden et al. "Informed Machine Learning–A Taxonomy and Survey of Integrating Knowledge into Learning Systems". In: *arXiv preprint arXiv:1903.12394* (2019).

[55] B Ravi Kiran et al. "Deep reinforcement learning for autonomous driving: A survey". In: *IEEE Transactions on Intelligent Transportation Systems* (2021).

[56] Chris Paxton et al. "Combining neural networks and tree search for task and motion planning in challenging environments". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 6059–6066.

[57] Jacob Andreas, Dan Klein, and Sergey Levine. "Modular multitask reinforcement learning with policy sketches". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 166–175.

[58] Martin L Puterman and Moon Chirl Shin. "Modified policy iteration algorithms for discounted Markov decision problems". In: *Management Science* 24.11 (1978), pp. 1127–1137.

[59] Steven Spielberg et al. "Toward self-driving processes: A deep reinforcement learning approach to control". In: *AIChE Journal* 65.10 (2019), e16689.

[60] Tuomas Haarnoja et al. *Soft Actor-Critic Algorithms and Applications*. 2019. arXiv: 1812.05905 [cs.LG].

[61] Thomas G Dietterich. "Hierarchical reinforcement learning with the MAXQ value function decomposition". In: *Journal of artificial intelligence research* 13 (2000), pp. 227–303.

[62] Richard S Sutton, Doina Precup, and Satinder Singh. "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning". In: *Artificial intelligence* 112.1-2 (1999), pp. 181–211.

[63] Pierre-Luc Bacon, Jean Harb, and Doina Precup. "The option-critic architecture". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 31. 1. 2017.

[64] Christian Daniel et al. "Hierarchical relative entropy policy search". In: *Journal of Machine Learning Research* 17 (2016), pp. 1–50.

[65] Arne Kesting et al. "Jam-Avoiding Adaptive Cruise Control (ACC) and its Impact on Traffic Dynamics". In: Springer Berlin Heidelberg, 2007, pp. 633–643.

[66] Matthew Niedoba et al. "Improving movement prediction of traffic actors using off-road loss and bias mitigation". In: *Workshop on'Machine Learning for Autonomous Driving'at Conference on Neural Information Processing Systems*. 2019.

[67] Ross Greer, Nachiket Deo, and Mohan Trivedi. "Trajectory Prediction in Autonomous Driving with a Lane Heading Auxiliary Loss". In: *IEEE Robotics and Automation Letters* 6.3 (2021), pp. 4907–4914.

[68] Jianyu Chen, Shengbo Eben Li, and Masayoshi Tomizuka. "Interpretable End-to-End Urban Autonomous Driving with Latent Deep Reinforcement Learning". In: July 7, 2020. arXiv: 2001.08726 [cs]. URL: http://arxiv.org/abs/2001.08726 (visited on 06/29/2021).

[69] Vikram Waradpande, Daniel Kudenko, and Megha Khosla. *Graph-Based State Representation for Deep Reinforcement Learning*. Version 1. Apr. 29, 2020. arXiv: 2004.13965 [cs, stat]. URL: http://arxiv.org/abs/2004.13965 (visited on 09/22/2021).

[70] Kyunghoon Cho and Songhwai Oh. "Learning-Based Model Predictive Control Under Signal Temporal Logic Specifications". en. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. Brisbane, QLD: IEEE, May 2018, pp. 7322–7329. ISBN: 978-1-5386-3081-5. DOI: 10.1109/ICRA.2018.8460811. URL: https://ieeexplore.ieee.org/document/8460811/ (visited on 05/12/2021).

[71] Kyunghoon Cho et al. "Deep Predictive Autonomous Driving Using Multi-Agent Joint Trajectory Prediction and Traffic Rules". In: *IEEE International Conference on Intelligent Robots and Systems* (2019). ISBN: 9781728140049 5 citations (Semantic Scholar/DOI) [2021-03-25], pp. 2076–2081. ISSN: 21530866. DOI: 10/ghk254.

[72] Andrea Censi et al. "Liability, Ethics, and Culture-Aware Behavior Specification Using Rulebooks". In: Mar. 1, 2019. arXiv: 1902.09355 [cs]. URL: http://arxiv.org/abs/1902.09355 (visited on 07/07/2021).

[73] Jesper Karlsson et al. “Encoding Human Driving Styles in Motion Planning for Autonomous Vehicles”. In: Mar. 2021.

[74] Laura von Rueden et al. “Combining machine learning and simulation to a hybrid modelling approach: Current and future directions”. In: *International symposium on intelligent data analysis*. 2020, pp. 548–560.

[75] Laura von Rueden et al. “Informed machine Learning–A taxonomy and survey of integrating prior knowledge into learning systems”. In: *IEEE Transactions on Knowledge and Data Engineering* (2021). DOI: 10/gkzc3j.

[76] Michael van Bekkum et al. *Modular Design Patterns for Hybrid Learning and Reasoning Systems: a taxonomy, patterns and use cases*. _eprint: 2102.11965. 2021.

[77] Sebastian Reich and Colin Cotter. *Probabilistic forecasting and Bayesian data assimilation*. Cambridge: Cambridge University Press, 2015. ISBN: 978-1-107-06939-8 978-1-107-66391-6.

[78] Sara Bouraya and Abdessamad Belangour. “Multi object tracking: a survey”. In: *Thirteenth International Conference on Digital Image Processing (ICDIP 2021)*. Ed. by Xudong Jiang and Hiroshi Fujita. Singapore, Singapore: SPIE, June 2021, p. 96. ISBN: 978-1-5106-4600-1 978-1-5106-4601-8. DOI: 10.1117/12.2602901. URL: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/11878/2602901/Multi-object-tracking-a-survey/10.1117/12.2602901.full (visited on 11/25/2021).

[79] Chanho Kim et al. “Multiple Hypothesis Tracking Revisited”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Dec. 2015.

[80] Angel F. Garcia-Fernandez et al. “Poisson Multi-Bernoulli Mixture Filter: Direct Derivation and Implementation”. In: *IEEE Transactions on Aerospace and Electronic Systems* 54.4 (Aug. 2018), pp. 1883–1901. ISSN: 0018-9251, 1557-9603, 2371-9877. DOI: 10.1109/TAES.2018.2805153. URL: https://ieeexplore.ieee.org/document/8289337/ (visited on 11/05/2021).

[81] *Multi Object Tracking*. https://www.youtube.com/channel/UCa2-fpj6AV8T6JK1uTRuFpw. Accessed: 2021-11-25.

[82] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. “Objects as Points”. In: *CoRR* abs/1904.07850 (2019). arXiv: 1904.07850. URL: http://arxiv.org/abs/1904.07850.

[83] Ba-Tuong Vo, Ba-Ngu Vo, and Antonio Cantoni. “Bayesian Filtering With Random Finite Set Observations”. In: *IEEE Transactions on Signal Processing* 56.4 (Apr. 2008), pp. 1313–1326. ISSN: 1053-587X. DOI: 10.1109/TSP.2007.908968. URL: http://ieeexplore.ieee.org/document/4471893/ (visited on 11/25/2021).

[84] Greg Welch, Gary Bishop, et al. “An introduction to the Kalman filter”. In: (1995). Publisher: Chapel Hill, NC, USA.

[85] Simon S. Haykin, ed. *Kalman filtering and neural networks*. Adaptive and learning systems for signal processing, communications, and control. New York: Wiley, 2001. ISBN: 978-0-471-36998-1.

[86] Jun S. Liu and Rong Chen. "Sequential Monte Carlo Methods for Dynamic Systems". en. In: *Journal of the American Statistical Association* 93.443 (Sept. 1998), pp. 1032–1044. ISSN: 0162-1459, 1537-274X. DOI: 10.1080/01621459.1998.10473765. URL: http://www.tandfonline.com/doi/full/10.1080/01621459.1998.10473765 (visited on 11/25/2021).

[87] Rafael Padilla, Sergio L. Netto, and Eduardo A. B. da Silva. "A Survey on Performance Metrics for Object-Detection Algorithms". In: *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*. Niterói, Brazil: IEEE, July 2020, pp. 237–242. ISBN: 978-1-72817-539-3. DOI: 10.1109/IWSSIP48289.2020.9145130. URL: https://ieeexplore.ieee.org/document/9145130/ (visited on 11/25/2021).

[88] Roy L. Streit. "The Poisson Point Process". en. In: *Poisson Point Processes*. Boston, MA: Springer US, 2010, pp. 11–55. ISBN: 978-1-4419-6922-4 978-1-4419-6923-1. DOI: 10.1007/978-1-4419-6923-1_2. URL: http://link.springer.com/10.1007/978-1-4419-6923-1_2 (visited on 11/25/2021).

[89] Harold W Kuhn. "The Hungarian method for the assignment problem". In: *Naval research logistics quarterly* 2.1-2 (1955). Publisher: Wiley Online Library, pp. 83–97.

[90] Katta G. Murty. "Letter to the Editor—An Algorithm for Ranking all the Assignments in Order of Increasing Cost". en. In: *Operations Research* 16.3 (June 1968), pp. 682–687. ISSN: 0030-364X, 1526-5463. DOI: 10.1287/opre.16.3.682. URL: http://pubsonline.informs.org/doi/abs/10.1287/opre.16.3.682 (visited on 11/05/2021).

[91] Michael Motro and Joydeep Ghosh. "Scaling data association for hypothesis-oriented mht". In: *2019 22th International Conference on Information Fusion (FUSION)*. IEEE, 2019, pp. 1–8.

[92] *PMBM Architecture*. https://youtu.be/x9GwFNB5f2k?t=74. Accessed: 2021-11-25.

[93] Charles C. Macadam. "Understanding and Modeling the Human Driver". In: *Vehicle System Dynamics* 40.1-3 (Jan. 2003), pp. 101–134. ISSN: 0042-3114. DOI: 10.1076/vesd.40.1.101.15875. URL: http://www.tandfonline.com/doi/abs/10.1076/vesd.40.1.101.15875 (visited on 11/25/2021).

[94] M. F. Land and D. N. Lee. "Where we look when we steer". en. In: *Nature* 369.6483 (June 1994), pp. 742–744. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/369742a0. URL: http://www.nature.com/articles/369742a0 (visited on 11/25/2021).

[95] O. Lappi. "Future path and tangent point models in the visual control of locomotion in curve driving". en. In: *Journal of Vision* 14.12 (Oct. 2014), pp. 21–21. ISSN: 1534-7362. DOI: 10.1167/14.12.21. URL: http://jov.arvojournals.org/Article.aspx?doi=10.1167/14.12.21 (visited on 11/25/2021).

[96] Dominik Petrich et al. "Map-based long term motion prediction for vehicles in traffic environments". In: *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. The Hague, Netherlands: IEEE, Oct. 2013, pp. 2166–2172. ISBN: 978-1-4799-2914-6. DOI: 10.1109/ITSC.2013.6728549. URL: http://ieeexplore.ieee.org/document/6728549/ (visited on 11/05/2021).

[97] Chun Yang, Michael Bakich, and Erik Blasch. "Nonlinear constrained tracking of targets on roads". In: *2005 7th International Conference on Information Fusion*. Vol. 1. IEEE, 2005, 8–pp.

[98] Chun Yang and Erik Blasch. *Fusion of tracks with road constraints*. Tech. rep. AIR FORCE RESEARCH LAB WRIGHT-PATTERSON AFB OH SENSORS DIRECTORATE, 2008.

[99] I. Bae et al. "Self-Driving like a Human driver instead of a Robocar: Personalized comfortable driving experience for autonomous vehicles". In: *arXiv:2001.03908 [cs, eess]* (Jan. 2020). arXiv: 2001.03908. URL: http://arxiv.org/abs/2001.03908 (visited on 11/25/2021).

[100] Gustavo Arechavaleta et al. "An Optimality Principle Governing Human Walking". In: *IEEE Transactions on Robotics* 24.1 (Feb. 2008), pp. 5–14. ISSN: 1552-3098. DOI: 10.1109/TRO.2008.915449. URL: http://ieeexplore.ieee.org/document/4456738/ (visited on 11/25/2021).

[101] A. Takahashi et al. "Local Path Planning And Motion Control For Agv In Positioning". In: *Proceedings. IEEE/RSJ International Workshop on Intelligent Robots and Systems '. (IROS '89) 'The Autonomous Mobile Robots and Its Applications*. Tsukuba, Japan: IEEE, 1989, pp. 392–397. DOI: 10.1109/IROS.1989.637936. URL: http://ieeexplore.ieee.org/document/637936/ (visited on 11/25/2021).

[102] Quan Tran and Jonas Firl. "Modelling of traffic situations at urban intersections with probabilistic non-parametric regression". In: *2013 IEEE Intelligent Vehicles Symposium (IV)*. Gold Coast City, Australia: IEEE, June 2013, pp. 334–339. ISBN: 978-1-4673-2755-8 978-1-4673-2754-1. DOI: 10.1109/IVS.2013.6629491. URL: http://ieeexplore.ieee.org/document/6629491/ (visited on 11/05/2021).

[103] Jurgen Wiest et al. "Probabilistic trajectory prediction with Gaussian mixture models". In: *2012 IEEE Intelligent Vehicles Symposium*. Alcal de Henares , Madrid, Spain: IEEE, June 2012, pp. 141–146. ISBN: 978-1-4673-2118-1 978-1-4673-2119-8 978-1-4673-2117-4. DOI: 10.1109/IVS.2012.6232277. URL: http://ieeexplore.ieee.org/document/6232277/ (visited on 11/05/2021).

[104] E. Mazor et al. “Interacting multiple model methods in target tracking: a survey”. In: *IEEE Transactions on Aerospace and Electronic Systems* 34.1 (Jan. 1998), pp. 103–123. ISSN: 00189251. DOI: 10.1109/7.640267. URL: http://ieeexplore.ieee.org/document/640267/ (visited on 11/05/2021).

[105] R.S. Sharp, D. Casanova, and P. Symonds. “A Mathematical Model for Driver Steering Control, with Design, Tuning and Performance Results”. In: *Vehicle System Dynamics* 33.5 (May 2000). Publisher: Taylor & Francis, pp. 289–326. ISSN: 0042-3114. DOI: 10.1076/0042-3114(200005)33:5;1-Q;FT289. URL: https://www.tandfonline.com/doi/abs/10.1076/0042-3114%28200005%2933%3A5%3B1-Q%3BFT289.

[106] M. Plöchl and J. Edelmann. “Driver models in automobile dynamics application”. en. In: *Vehicle System Dynamics* 45.7-8 (July 2007), pp. 699–741. ISSN: 0042-3114, 1744-5159. DOI: 10.1080/00423110701432482. URL: http://www.tandfonline.com/doi/abs/10.1080/00423110701432482 (visited on 11/26/2021).

[107] R. S. Sharp and Huei Peng. “Vehicle dynamics applications of optimal control theory”. en. In: *Vehicle System Dynamics* 49.7 (July 2011), pp. 1073–1111. ISSN: 0042-3114, 1744-5159. DOI: 10.1080/00423114.2011.586707. URL: http://www.tandfonline.com/doi/abs/10.1080/00423114.2011.586707 (visited on 11/26/2021).

[108] C. I. Chatzikomis and K. N. Spentzas. “A path-following driver model with longitudinal and lateral control of vehicle’s motion”. en. In: *Forschung im Ingenieurwesen* 73.4 (Dec. 2009), pp. 257–266. ISSN: 0015-7899, 1434-0860. DOI: 10.1007/s10010-009-0112-5. URL: http://link.springer.com/10.1007/s10010-009-0112-5 (visited on 11/25/2021).

[109] M. Yamakado and M. Abe. “An experimentally confirmed driver longitudinal acceleration control model combined with vehicle lateral motion”. en. In: *Vehicle System Dynamics* 46.sup1 (Sept. 2008), pp. 129–149. ISSN: 0042-3114, 1744-5159. DOI: 10.1080/00423110701882363. URL: http://www.tandfonline.com/doi/abs/10.1080/00423110701882363 (visited on 11/25/2021).

[110] Alexander Liniger, Alexander Domahidi, and Manfred Morari. “Optimization-based autonomous racing of 1:43 scale RC cars: OPTIMIZATION-BASED AUTONOMOUS RACING”. en. In: *Optimal Control Applications and Methods* 36.5 (Sept. 2015), pp. 628–647. ISSN: 01432087. DOI: 10.1002/oca.2123. URL: https://onlinelibrary.wiley.com/doi/10.1002/oca.2123 (visited on 11/25/2021).

[111] Christoph Rösmann et al. “Trajectory modification considering dynamic constraints of autonomous robots”. In: *ROBOTIK 2012; 7th German Conference on Robotics*. VDE, 2012, pp. 1–6.

[112] Fritz Ulbrich et al. "Stable timed elastic bands with loose ends". In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. Los Angeles, CA, USA: IEEE, June 2017, pp. 186–192. ISBN: 978-1-5090-4804-5. DOI: 10.1109/IVS.2017.7995718. URL: http://ieeexplore.ieee.org/document/7995718/ (visited on 11/25/2021).

[113] T. Barbié et al. "Gaussian mixture spline trajectory: learning from a dataset, generating trajectories without one". en. In: *Advanced Robotics* 32.10 (May 2018), pp. 547–558. ISSN: 0169-1864, 1568-5535. DOI: 10.1080/01691864.2018.1465849. URL: https://www.tandfonline.com/doi/full/10.1080/01691864.2018.1465849 (visited on 11/26/2021).

[114] Christopher M Bishop. "Mixture density networks". In: (1994). Publisher: Aston University.

[115] Osama Makansi et al. "Overcoming Limitations of Mixture Density Networks: A Sampling and Fitting Framework for Multimodal Future Prediction". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[116] Ivan Kobyzev, Simon J.D. Prince, and Marcus A. Brubaker. "Normalizing Flows: An Introduction and Review of Current Methods". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.11 (Nov. 2021), pp. 3964–3979. ISSN: 0162-8828, 2160-9292, 1939-3539. DOI: 10.1109/TPAMI.2020.2992934. URL: https://ieeexplore.ieee.org/document/9089305/ (visited on 11/26/2021).

[117] George Papamakarios et al. "Normalizing Flows for Probabilistic Modeling and Inference". In: *Journal of Machine Learning Research* 22.57 (2021), pp. 1–64. URL: http://jmlr.org/papers/v22/19-1028.html.

[118] Lynton Ardizzone et al. "Analyzing Inverse Problems with Invertible Neural Networks". In: *arXiv:1808.04730 [cs, stat]* (Feb. 2019). arXiv: 1808.04730. URL: http://arxiv.org/abs/1808.04730 (visited on 11/25/2021).

[119] Govinda Anantha Padmanabha and Nicholas Zabaras. "Solving inverse problems using conditional invertible neural networks". en. In: *Journal of Computational Physics* 433 (May 2021), p. 110194. ISSN: 00219991. DOI: 10.1016/j.jcp.2021.110194. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021999121000899 (visited on 11/25/2021).

[120] Paul Lyonel Hagemann and Sebastian Neumayer. "Stabilizing Invertible Neural Networks Using Mixture Models". In: *Inverse Problems* (Feb. 2021). Publisher: IOP Publishing. DOI: 10.1088/1361-6420/abe928. URL: https://doi.org/10.1088/1361-6420/abe928.

[121] Ashish Vaswani et al. "Attention is all you need". In: *Advances in Neural Information Processing Systems (NeurIPS)* (2017), pp. 5998–6008.

[122] Hang Zhao et al. "TNT: Target-driveN Trajectory Prediction". In: *arXiv:2008.08294 [cs]* (Aug. 2020). arXiv: 2008.08294. URL: http://arxiv.org/abs/2008.08294 (visited on 06/22/2021).

[123] J. Gao et al. "VectorNet: Encoding HD Maps and Agent Dynamics From Vectorized Representation". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, June 2020, pp. 11522–11530. DOI: 10.1109/CVPR42600.2020.01154. URL: https://doi.ieeecomputersociety.org/10.1109/CVPR42600.2020.01154.

[124] Ye Yuan et al. "AgentFormer: Agent-Aware Transformers for Socio-Temporal Multi-Agent Forecasting". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021.

[125] You Li and Javier Ibanez-Guzman. "Lidar for Autonomous Driving: The Principles, Challenges, and Trends for Automotive Lidar and Perception Systems". In: *IEEE Signal Processing Magazine* 37.4 (July 2020), pp. 50–61. ISSN: 1558-0792. DOI: 10.1109/MSP.2020.2973615. URL: https://ieeexplore.ieee.org/abstract/document/9127855.

[126] Joachim Mathes. 2021. URL: https://www.valeo.com/wp-content/uploads/2021/01/Evercore-CES-13-Jan-2021-Marc-Vrecko-%5C_-Joachim-Mathes.pdf.

[127] Yulan Guo et al. *Deep Learning for 3D Point Clouds: A Survey*. 2020. arXiv: 1912.12033. URL: https://arxiv.org/abs/1912.12033.

[128] Tiago Cortinhal, George Tzelepis, and Eren Erdal Aksoy. *SalsaNext: Fast, Uncertainty-aware Semantic Segmentation of LiDAR Point Clouds for Autonomous Driving*. 2020. arXiv: 2003.03653. URL: https://arxiv.org/abs/2003.03653.

[129] A. Milioto et al. "RangeNet++: Fast and Accurate LiDAR Semantic Segmentation". In: *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. 2019. URL: https://www.ipb.uni-bonn.de/wp-content/papercite-data/pdf/milioto2019iros.pdf.

[130] Shuang Wu et al. "Convolution with even-sized kernels and symmetric padding". In: (2019). arXiv: 1903.08385v2 [cs.CV]. URL: https://arxiv.org/abs/1903.08385.

[131] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. "Dilated Residual Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017. URL: https://openaccess.thecvf.com/content_cvpr_2017/html/Yu_Dilated_Residual_Networks_CVPR_2017_paper.html.

[132] Muhammad Mobaidul Islam, Abdullah Al Redwan Newaz, and Ali Karimoddini. "A Pedestrian Detection and Tracking Framework for Autonomous Cars: Efficient Fusion of Camera and LiDAR Data". In: (2021). arXiv: 2108.12375 [cs.CV].

[133] Petru Soviany and Radu Tudor Ionescu. "Optimizing the Trade-Off between Single-Stage and Two-Stage Deep Object Detectors using Image Difficulty Prediction". In: *20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2018, Timisoara, Romania, September 20-23, 2018*. IEEE, 2018, pp. 209–214. DOI: 10.1109/SYNASC.2018.00041. URL: https://doi.org/10.1109/SYNASC.2018.00041.

[134] Wei Liu et al. "SSD: Single Shot MultiBox Detector". In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe et al. Cham: Springer International Publishing, 2016, pp. 21–37.

[135] J. Redmon and A. Farhadi. "YOLO9000: Better, Faster, Stronger". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 6517–6525. DOI: 10.1109/CVPR.2017.690.

[136] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes et al. Vol. 28. Curran Associates, Inc., 2015.

[137] K. He et al. "Mask R-CNN". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2980–2988. DOI: 10.1109/ICCV.2017.322.

[138] Irtiza Hasan et al. "Pedestrian Detection: The Elephant In The Room". In: *CoRR* abs/2003.08799 (2020). arXiv: 2003.08799. URL: https://arxiv.org/abs/2003.08799.

[139] Wei Liu et al. "High-Level Semantic Feature Detection: A New Perspective for Pedestrian Detection". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 5182–5191. DOI: 10.1109/CVPR.2019.00533.

[140] Wenhao Wang. "Adapted Center and Scale Prediction: More Stable and More Accurate". In: *CoRR* abs/2002.09053 (2020). arXiv: 2002.09053. URL: https://arxiv.org/abs/2002.09053.

[141] Jialiang Zhang et al. "Attribute-aware Pedestrian Detection in a Crowd". In: (2019). arXiv: 1910.09188 [cs.CV].

[142] Yanwei Pang et al. "Mask-Guided Attention Network for Occluded Pedestrian Detection". In: Oct. 2019, pp. 4966–4974. DOI: 10.1109/ICCV.2019.00507.

[143] Zhaowei Cai and Nuno Vasconcelos. "Cascade R-CNN: High Quality Object Detection and Instance Segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.5 (2021), pp. 1483–1498. DOI: 10.1109/TPAMI.2019.2956516.

[144] Piotr Dollár et al. "Pedestrian Detection: A Benchmark". In: June 2009, pp. 304–311. DOI: 10.1109/CVPRW.2009.5206631.

[145] Shanshan Zhang, Rodrigo Benenson, and Bernt Schiele. "CityPersons: A Diverse Dataset for Pedestrian Detection". In: July 2017, pp. 4457–4465. DOI: 10.1109/CVPR.2017.474.

[146] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. "Objects as Points". In: *CoRR* abs/1904.07850 (2019). arXiv: 1904.07850. URL: http://arxiv.org/abs/1904.07850.

[147] Ping Luo, Jiamin Ren, and Zhanglin Peng. "Differentiable Learning-to-Normalize via Switchable Normalization". In: June 2018.

[148] Jia Deng et al. "Large-Scale Object Classification Using Label Relation Graphs". In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Cham: Springer International Publishing, 2014, pp. 48–64. ISBN: 978-3-319-10590-1.

[149] Hamed Pirsiavash, Carl Vondrick, and Antonio Torralba. "Inferring the Why in Images". In: *CoRR* abs/1406.5472 (2014). arXiv: 1406.5472. URL: http://arxiv.org/abs/1406.5472.

[150] Qi Wu et al. "Ask Me Anything: Free-form Visual Question Answering Based on Knowledge from External Sources". In: *CoRR* abs/1511.06973 (2015). arXiv: 1511.06973. URL: http://arxiv.org/abs/1511.06973.

[151] Cewu Lu et al. "Visual Relationship Detection with Language Priors". In: *CoRR* abs/1608.00187 (2016). arXiv: 1608.00187. URL: http://arxiv.org/abs/1608.00187.

[152] Jongkwang Hong et al. "Discovering overlooked objects: Context-based boosting of object detection in indoor scenes". In: *Pattern Recognition Letters* 86 (2017), pp. 56–61. ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2016.12.017. URL: https://www.sciencedirect.com/science/article/pii/S0167865516303725.

[153] Markus Braun et al. "EuroCity Persons: A Novel Benchmark for Person Detection in Traffic Scenes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019), pp. 1–1. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2019.2897684.

[154] Marius Cordts et al. "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: June 2016. DOI: 10.1109/CVPR.2016.350.

[155] *Wider Pedestrian 2019*. https://competitions.codalab.org/competitions/20132. Accessed: 2021-10-07.

[156] Shuai Shao et al. "CrowdHuman: A Benchmark for Detecting Human in a Crowd". In: Apr. 2018.

[157] Andrew Tao, Karan Sapra, and Bryan Catanzaro. "Hierarchical Multi-Scale Attention for Semantic Segmentation". In: May 2020.

[158] Gerhard Neuhold et al. "The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 5000–5009. DOI: 10.1109/ICCV.2017.534.

[159] Heiko Paulheim. "Knowledge graph refinement: A survey of approaches and evaluation methods". English. In: *Semantic Web* 8.3 (2017), pp. 489–508. DOI: 10.3233/SW-160218. URL: https://madoc.bib.uni-mannheim.de/41515/.

[160] Xin Luna Dong et al. "Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge Fusion". In: *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*. Evgeniy Gabrilovich Wilko Horn Ni Lao Kevin Murphy Thomas Strohmann Shaohua Sun Wei Zhang Geremy Heitz. 2014, pp. 601–610. URL: http://www.cs.cmu.edu/~nlao/publication/2014.kdd.pdf.

[161] *Road signs in Germany*. URL: https://en.wikipedia.org/wiki/Road_signs_in_Germany.

[162] Shaoqing Ren et al. "Faster R-CNN: towards real-time object detection with region proposal networks". In: *IEEE transactions on pattern analysis and machine intelligence* 39.6 (2016), pp. 1137–1149.

[163] D Jifeng et al. "Object Detection via Region-Based Fully Convolutional Networks". In: *People's Posts and Telecommunications Press* (2016).

[164] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[165] Tsung-Yi Lin et al. "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[166] Chenhan Jiang et al. "Hybrid knowledge routed modules for large-scale object detection". In: *arXiv preprint arXiv:1810.12681* (2018).

[167] Christopher M. Bishop. *Pattern recognition and machine learning*. en. Information science and statistics. New York: Springer, 2006. ISBN: 978-0-387-31073-2.

[168] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. "ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst". In: *arXiv:1812.03079 [cs]* (Dec. 2018). arXiv: 1812.03079. URL: http://arxiv.org/abs/1812.03079 (visited on 01/21/2021).

[169] Wenyuan Zeng et al. "End-to-end Interpretable Neural Motion Planner". In: *arXiv:2101.06679 [cs]* (Jan. 2021). arXiv: 2101.06679. URL: http://arxiv.org/abs/2101.06679 (visited on 02/24/2021).

[170] Bob Wei et al. "Perceive, Attend, and Drive: Learning Spatial Attention for Safe Self-Driving". In: *arXiv:2011.01153 [cs]* (Nov. 2020). arXiv: 2011.01153. URL: http://arxiv.org/abs/2011.01153 (visited on 02/07/2021).

[171] Sergio Casas et al. "The Importance of Prior Knowledge in Precise Multimodal Prediction". In: *arXiv:2006.02636 [cs, stat]* (June 2020). arXiv: 2006.02636. URL: http://arxiv.org/abs/2006.02636 (visited on 06/03/2021).

[172] Xiao Li et al. "Differentiable Logic Layer for Rule Guided Trajectory Prediction". en. In: *Conference on Robot Learning (CoRL)* (2021), p. 17.

[173] Luis C. Lamb et al. "Graph Neural Networks Meet Neural-Symbolic Computing: A Survey and Perspective". In: *arXiv:2003.00330 [cs]* (June 2021). arXiv: 2003.00330. URL: http://arxiv.org/abs/2003.00330 (visited on 06/30/2021).

[174] Adam Santoro et al. "A simple neural network module for relational reasoning". In: *arXiv:1706.01427 [cs]* (June 2017). arXiv: 1706.01427. URL: http://arxiv.org/abs/1706.01427 (visited on 06/30/2021).

[175] L. Serafini and A. d'Avila Garcez. "Logic Tensor Networks: Deep Learning and Logical Reasoning from Data and Knowledge". In: *arXiv:1606.04422 [cs]* (July 2016). arXiv: 1606.04422. URL: http://arxiv.org/abs/1606.04422 (visited on 12/22/2020).

[176] Alexandre Donzé and Oded Maler. "Robust Satisfaction of Temporal Logic over Real-Valued Signals". en. In: *Formal Modeling and Analysis of Timed Systems*. Ed. by David Hutchison et al. Vol. 6246. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 92–106. ISBN: 978-3-642-15296-2 978-3-642-15297-9. DOI: 10.1007/978-3-642-15297-9_9. URL: http://link.springer.com/10.1007/978-3-642-15297-9_9 (visited on 02/22/2021).

[177] N. Aréchiga. "Specifying Safety of Autonomous Vehicles in Signal Temporal Logic". In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. ISSN: 2642-7214. June 2019, pp. 58–63. DOI: 10.1109/IVS.2019.8813875.

[178] S. Maierhofer et al. "Formalization of Interstate Traffic Rules in Temporal Logic". In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. ISSN: 2642-7214. Oct. 2020, pp. 752–759. DOI: 10.1109/IV47402.2020.9304549.

[179] Klemens Esterle, Luis Gressenbuch, and Alois Knoll. "Formalizing Traffic Rules for Machine Interpretability". In: *arXiv:2007.00330 [cs]* (Sept. 2020). arXiv: 2007.00330. URL: http://arxiv.org/abs/2007.00330 (visited on 01/11/2021).

[180] Julian Bernhard et al. "BARK: Open Behavior Benchmarking in Multi-Agent Environments". In: *arXiv:2003.02604 [cs]* (Sept. 2020). arXiv: 2003.02604. DOI: 10.1109/IROS45743.2020.9341222. URL: http://arxiv.org/abs/2003.02604 (visited on 03/04/2021).

[181] David Silver et al. "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm". In: *arXiv:1712.01815 [cs]* (Dec. 2017). arXiv: 1712.01815. URL: http://arxiv.org/abs/1712.01815 (visited on 12/12/2020).

[182] Mikael Henaff, Alfredo Canziani, and Yann LeCun. "Model-Predictive Policy Learning with Uncertainty Regularization for Driving in Dense Traffic". In: *arXiv:1901.02705 [cs, stat]* (Jan. 2019). arXiv: 1901.02705. URL: http://arxiv.org/abs/1901.02705 (visited on 03/02/2021).

[183] Boris Ivanovic et al. "MATS: An Interpretable Trajectory Forecasting Representation for Planning and Control". In: *arXiv:2009.07517 [cs, eess]* (Sept. 2020). arXiv: 2009.07517. URL: http://arxiv.org/abs/2009.07517 (visited on 01/14/2021).

[184] Abbas Sadat et al. "Perceive, Predict, and Plan: Safe Motion Planning Through Interpretable Semantic Representations". en. In: *Computer Vision – ECCV 2020*. Ed. by Andrea Vedaldi et al. Vol. 12368. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 414–430. ISBN: 978-3-030-58591-4 978-3-030-58592-1. DOI: 10.1007/978-3-030-58592-1_25. URL: http://link.springer.com/10.1007/978-3-030-58592-1_25 (visited on 01/29/2021).

[185] Sergio Casas, Abbas Sadat, and Raquel Urtasun. "MP3: A Unified Model to Map, Perceive, Predict and Plan". In: *arXiv:2101.06806 [cs]* (Jan. 2021). arXiv: 2101.06806. URL: http://arxiv.org/abs/2101.06806 (visited on 02/07/2021).

[186] Alexander Cui et al. "LookOut: Diverse Multi-Future Prediction and Planning for Self-Driving". In: *arXiv:2101.06547 [cs]* (May 2021). arXiv: 2101.06547. URL: http://arxiv.org/abs/2101.06547 (visited on 10/16/2021).

[187] Abbas Sadat et al. "Jointly Learnable Behavior and Trajectory Planning for Self-Driving Vehicles". In: *arXiv:1910.04586 [cs]* (Oct. 2019). arXiv: 1910.04586. URL: http://arxiv.org/abs/1910.04586 (visited on 02/01/2021).

[188] Ming Liang et al. "Learning Lane Graph Representations for Motion Forecasting". In: *arXiv:2007.13732 [cs]* (July 2020). arXiv: 2007.13732. URL: http://arxiv.org/abs/2007.13732 (visited on 11/21/2021).

[189] Tim Salzmann et al. "Trajectron++: Dynamically-Feasible Trajectory Forecasting With Heterogeneous Data". In: *arXiv:2001.03093 [cs]* (Jan. 2021). arXiv: 2001.03093. URL: http://arxiv.org/abs/2001.03093 (visited on 01/28/2021).

[190] Thomas Kipf et al. "Neural Relational Inference for Interacting Systems". In: *arXiv:1802.04687 [cs, stat]* (June 2018). arXiv: 1802.04687. URL: http://arxiv.org/abs/1802.04687 (visited on 04/16/2021).

[191] Thomas Kipf, Elise van der Pol, and Max Welling. "Contrastive Learning of Structured World Models". In: *arXiv:1911.12247 [cs, stat]* (Jan. 2020). arXiv: 1911.12247. URL: http://arxiv.org/abs/1911.12247 (visited on 10/22/2021).

[192] Yafu Tian et al. "Road Scene Graph: A Semantic Graph-Based Scene Representation Dataset for Intelligent Vehicles". In: *arXiv:2011.13588 [cs]* (Nov. 2020). arXiv: 2011.13588. URL: http://arxiv.org/abs/2011.13588 (visited on 04/07/2021).

[193] Wenyuan Zeng et al. "DSDNet: Deep Structured self-Driving Network". In: *arXiv:2008.06041 [cs]* (Aug. 2020). arXiv: 2008.06041. URL: http://arxiv.org/abs/2008.06041 (visited on 02/01/2021).

[194] Jerry Liu et al. "Deep Structured Reactive Planning". In: *arXiv:2101.06832 [cs]* (Jan. 2021). arXiv: 2101.06832. URL: http://arxiv.org/abs/2101.06832 (visited on 02/07/2021).

[195] Nicholas Rhinehart et al. "PRECOG: PREdiction Conditioned On Goals in Visual Multi-Agent Settings". In: *arXiv:1905.01296 [cs, stat]* (Sept. 2019). arXiv: 1905.01296. URL: http://arxiv.org/abs/1905.01296 (visited on 02/01/2021).

[196] Christoph Benzmüller and Lawrence C. Paulson. *Quantified Multimodal Logics in Simple Type Theory*. Tech. rep. DFKI Bremen GmbH, Safe and Secure Cognitive Systems, Cartesium, Enrique Schmidt Str. 5, D–28359 Bremen, Germany: Saarland University, 2009. URL: http://arxiv.org/abs/0905.2435. arXiv:0905.2435.

[197] Andrea Censi et al. "Liability, Ethics, and Culture-Aware Behavior Specification using Rulebooks". In: *CoRR* abs/1902.09355 (2019). arXiv: 1902.09355. URL: http://arxiv.org/abs/1902.09355.

[198] Danijar Hafner et al. "Learning Latent Dynamics for Planning from Pixels". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, June 2019, pp. 2555–2565. URL: http://proceedings.mlr.press/v97/hafner19a.html.

[199] Jianyu Chen, Shengbo Eben Li, and Masayoshi Tomizuka. "Interpretable End-to-End Urban Autonomous Driving With Latent Deep Reinforcement Learning". In: *IEEE Transactions on Intelligent Transportation Systems* (2021), pp. 1–11. DOI: 10.1109/TITS.2020.3046646.

[200] Kashyap Chitta, Aditya Prakash, and Andreas Geiger. "NEAT: Neural Attention Fields for End-to-End Autonomous Driving". In: *International Conference on Computer Vision (ICCV)*. 2021.

[201] Marin Toromanoff, Émilie Wirbel, and Fabien Moutarde. "End-to-End Model-Free Reinforcement Learning for Urban Driving using Implicit Affordances". In: *CoRR* abs/1911.10868 (2019). arXiv: 1911.10868. URL: http://arxiv.org/abs/1911.10868.

[202] Zonghan Wu et al. "A Comprehensive Survey on Graph Neural Networks". In: *IEEE Transactions on Neural Networks and Learning Systems* 32 (2019), pp. 4–24.

[203] Alexey Dosovitskiy et al. "CARLA: An Open Urban Driving Simulator". In: *arXiv:1711.03938 [cs]* (Nov. 2017). arXiv: 1711.03938. URL: http://arxiv.org/abs/1711.03938 (visited on 06/29/2021).