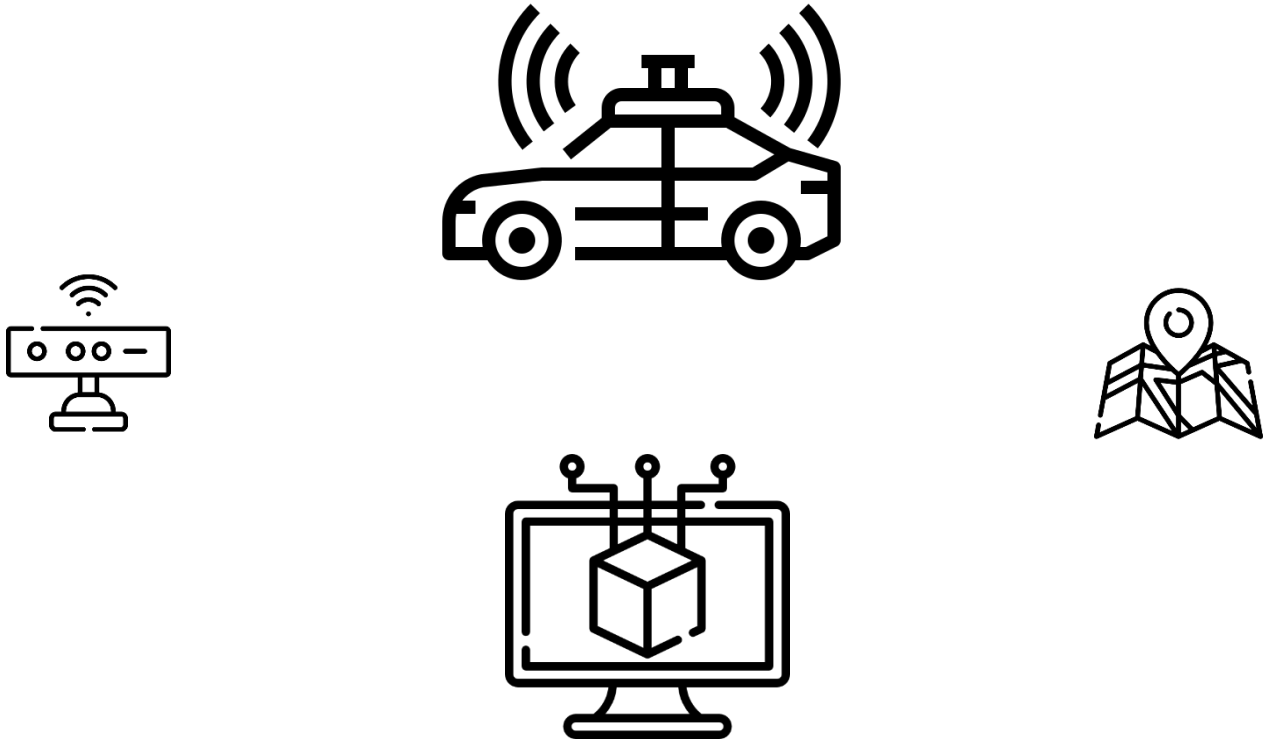
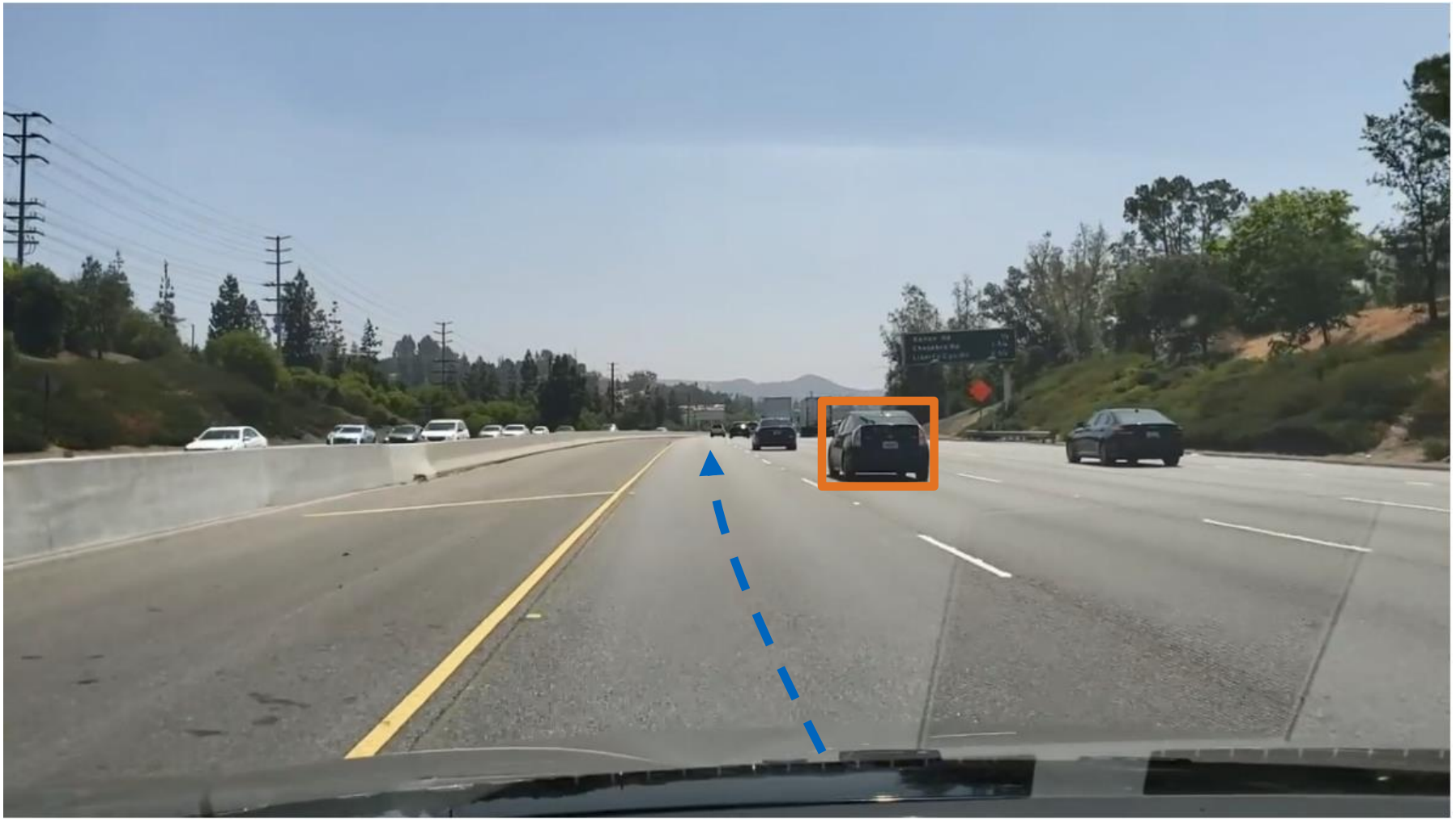
Institute of Automotive Technology
Faculty for Mechanical Engineering
Technical University of Munich

Autonomous Driving Software Engineering

Prof. Dr.-Ing. Markus Lienkamp

Phillip Karle M. Sc.





https://www.youtube.com/watch?v=01_SVshk-MY

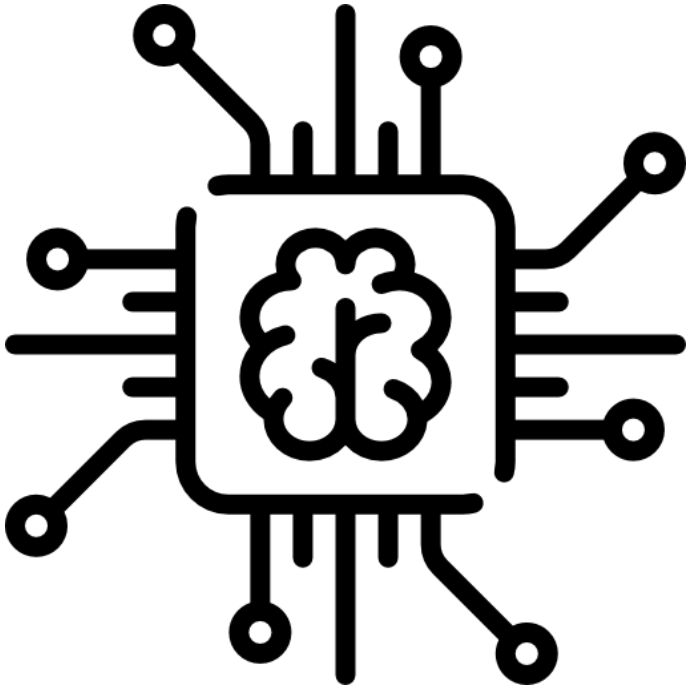
https://www.youtube.com/watch?v=SBpUppJACxA


4m


15m

How to anticipate other traffic participants?

Anticipatory Driving through Riding Experience



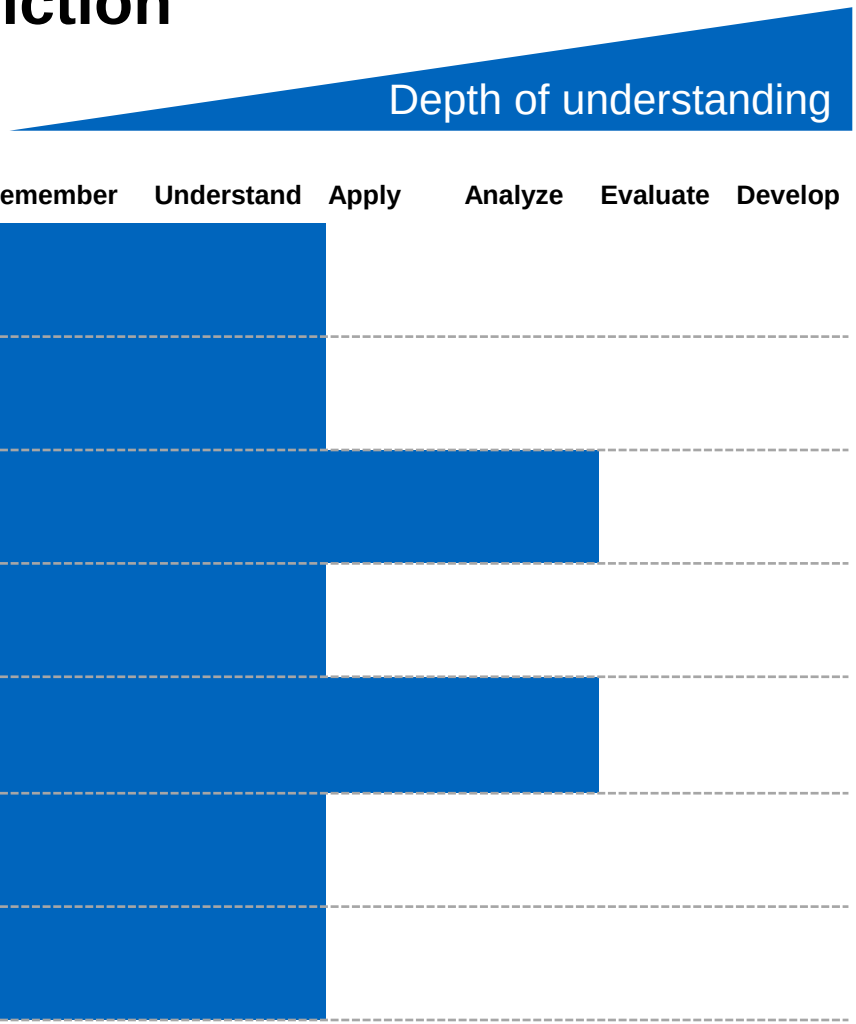
Motion Prediction through Scenario Understanding

Lecture Overview

Lecture – 90min	Practice – 45min
1 Introduction: Autonomous Driving Karle	**1 Practice** Karle
2 Perception I: Mapping Sauerbeck	**2 Practice** Sauerbeck
3 Perception II: Localization Sauerbeck	**3 Practice** Sauerbeck
4 Perception III: Detection Huch	**4 Practice** Huch
5 Prediction Karle	**5 Practice** Karle
6 Planning I: Global Planning Trauth	**6 Practice** Trauth
7 Planning II: Local Planning Ögretmen	**7 Practice** Ögretmen
8 Control 15.06.2021 – Wischnewski	**8 Practice** Wischnewski
9 Safety Assessment Stahl	**9 Practice** Stahl
10 Teleoperated Driving Feiler	**10 Practice** Feiler
11 End-to-End Betz	**11 Practice** Betz
12 From Driver to Passenger Fank	**12 Practice** Karle

Objectives for Lecture 5: Prediction

After the lecture you are able to…



… describe the necessity of motion prediction for dynamic motion planning and according interfaces

… compare the three classes of motion prediction and relevant properties (pros / cons) of each class

… implement a physics-based prediction model and check the prediction performance

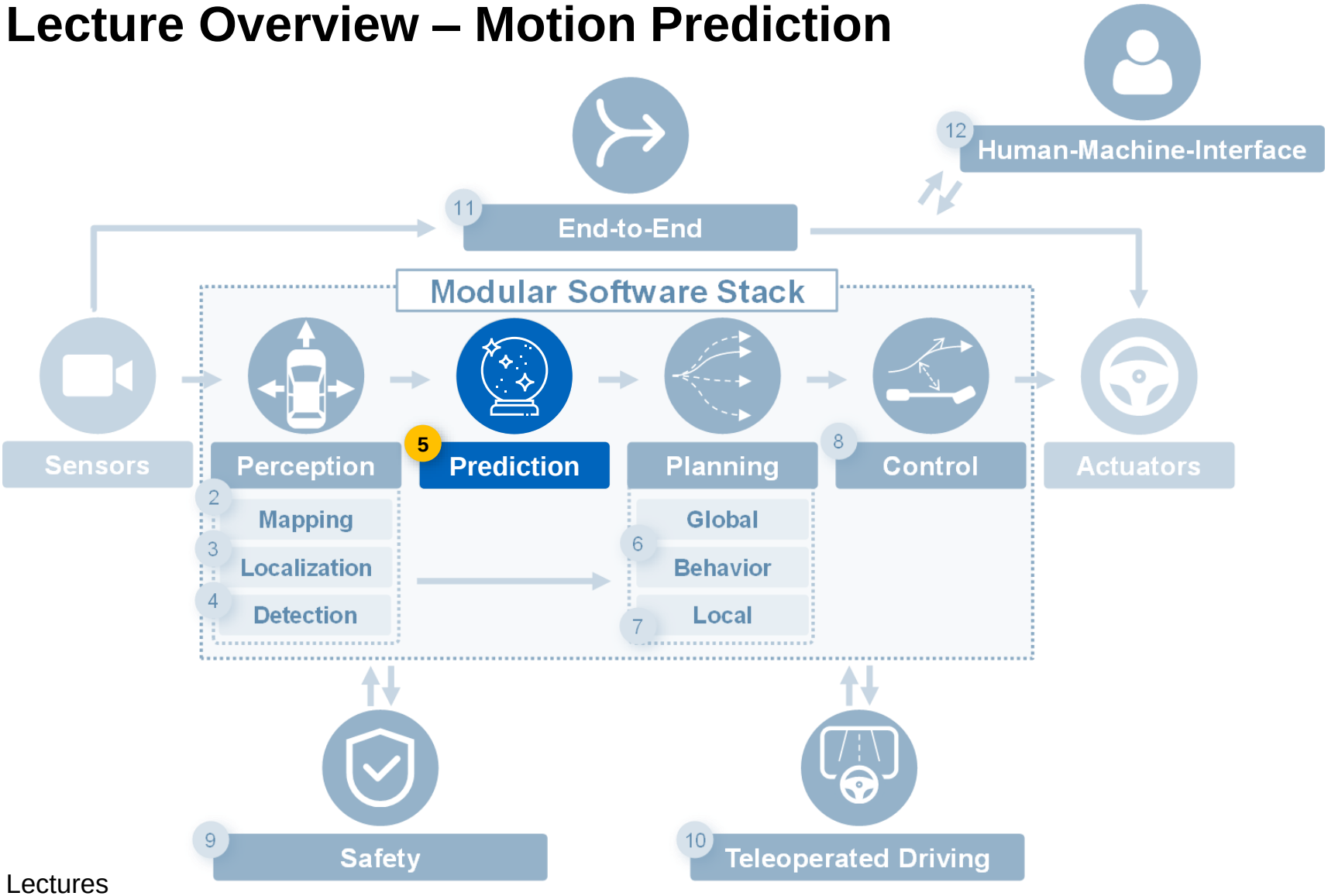
… depict Encoder-Decoder architectures

… execute a RNN-autoencoder for motion prediction including past trajectories and semantic features and interpret its prediction behavior

… explain Markov processes and the Markov assumption

… describe the dual problem of prediction and planning

Lecture Overview – Motion Prediction



X = Lectures

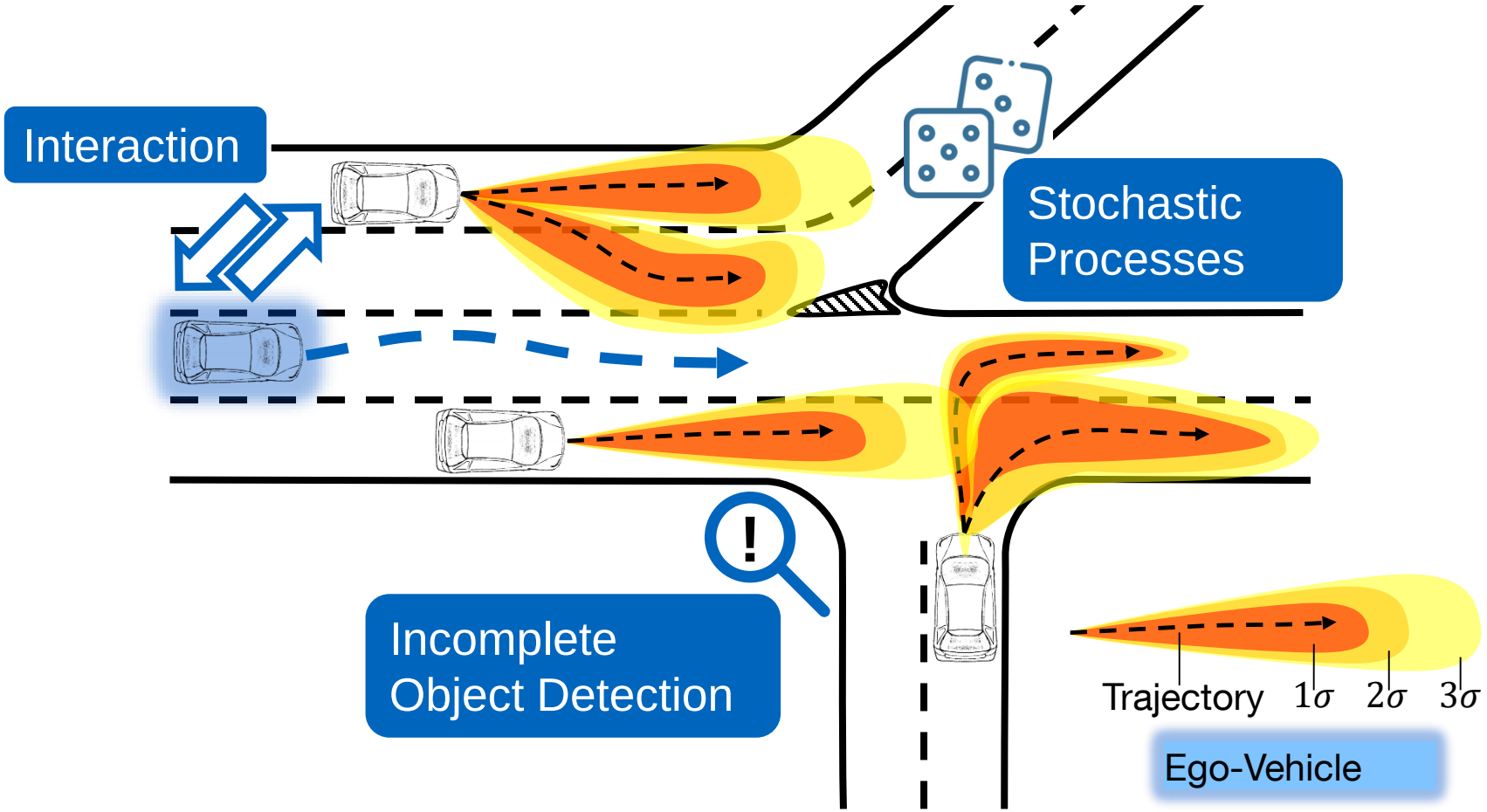
Prediction
Prof. Dr. Markus Lienkamp

Phillip Karle, M. Sc.

Agenda

1. Foundations
2. Physics-Based Prediction
3. Pattern-Based Prediction
4. Planning-Based Prediction
5. Summary and Outlook

Foundations – Complexity of Road Traffic



Foundations – Complexity of Road Traffic

Incomplete Object Detection

- Faulty or inaccurate detection of relevant features,
  e.g. turn indicator of car, inaccurate speed estimation
- Missing social features, e.g. eye contact with pedestrians, cyclists
- Sensor occlusion and mis-detections

Stochastics traffic processes

- Uncertainty about traffic behavior
- Irrational driving behavior in human drivers (mixed traffic)
- Multimodal future

Interaction between road users

- Each others' decision are not independent,
  but influence each other

Additional Slides

Challenges of Motion Prediction:

Reliable decision making and motion planning for autonomous vehicles in road traffic are essential for occupant safety and comfort. Sophisticated and robust scenario understanding and motion prediction of surrounding traffic participants are necessary to carry out these tasks. However, it is not possible to ascertain further environmental states in a deterministic way. In general, there are three primary reasons why traffic situations are strongly probabilistic processes:

(1) incomplete detection of relevant features of dynamic and static obstacles

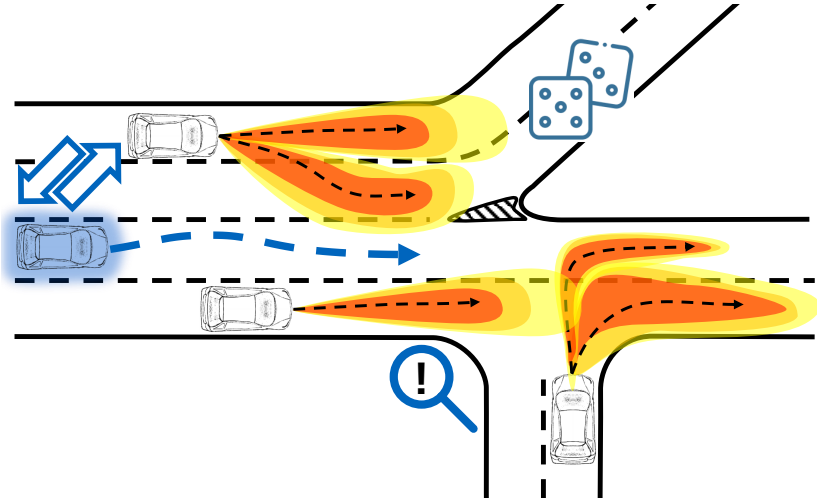
(2) stochastic and spontaneous decisions made by traffic participants

(3) high interaction between decisions and motions of neighboring objects. Commonly, models, which take interaction among objects into account, are referred to as “interaction aware”.

Foundations – Complexity of Road Traffic

Planning in dynamic and uncertain environment

- Uncertainty results in risk of planned ego decisions and trajectories
- Uncertainty and resulting collision probabilities have to be quantified to ensure safety of traffic participants
- If complexity or uncertainty in current situation is too high, no valid motion is possible
  → Freezing Robot Problem
  → Teleoperated Driving is required



Foundations – Why do we need prediction?

„Sense – Plan – Act“ falls short in complex and dynamic environment, such as road traffic

We need to:

- **Understand**
  Determine intentions of surrounding objects and reason about possible behavior
- **Predict**
  Forecast, what the surrounding objects will do and quantify the probability of each forecast

Foundations – Why do we need prediction?

„Sense – Plan – Act“ falls short in complex and dynamic environment, such as road traffic

We need to:

- **Plan**
  Prediction is used to evaluate the collision probability of the possible further action of the ego object
- **Play**
  There is a dependency between the decisions of each object. The interaction has to be considered in the prediction and ego motion planning

Foundations – Human Driving Ability

Human	Algorithm
Driving Experience	Scenario Understanding
Anticipatory Driving	Motion Prediction

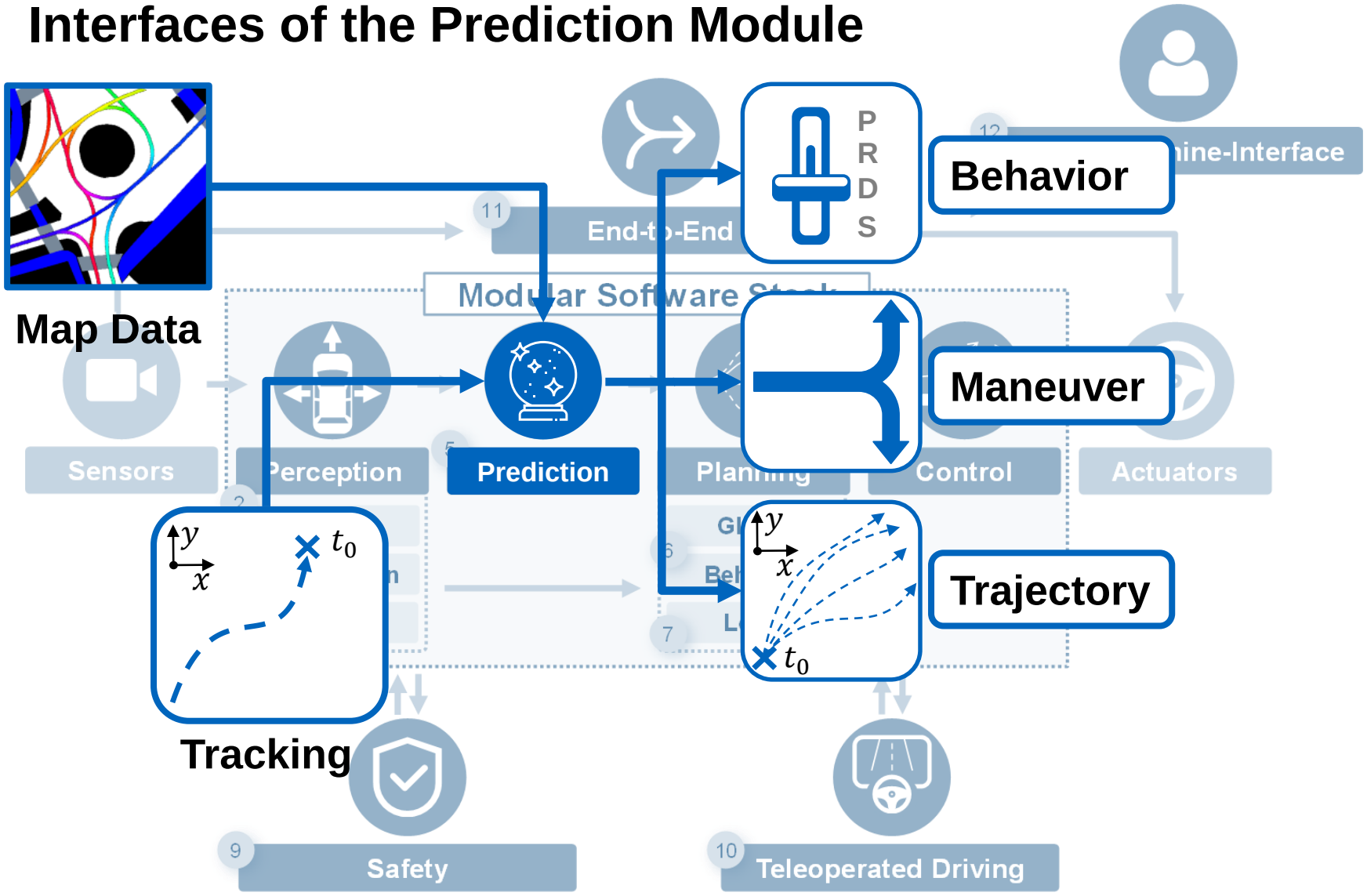
Human is skilled in complex thinking

Prediction Skills

- Based on memory of similar situations
- Pattern identification and matching
- Prediction by remembering what happened in similar situation
- Adaptivity based on prediction errors

Goal: Gain confidence in situation to make decisions

Interfaces of the Prediction Module



Additional Slides

Scenario understanding and motion prediction are essential components for completely replacing human drivers and for enabling highly and fully automated and driving of (SAE-Level 4/5) autonomous vehicles. In deeply stochastic and uncertain traffic scenarios, autonomous driving software must act beyond existing traffic rules and must predict critical situations in advance to provide safe and comfortable rides.

Scenario Understanding comprises estimating intentions, identifying and encoding relevant features in a scene and the ability to associate behavior and decisions of surrounding objects. Hence, it is an essential subtask and the basis for making comprehensive predictions.

Motion prediction implicates the forecast of an object's behavior, maneuver or trajectory, depending on the desired degree of abstraction. **Behavior** is the most general term and defines an action and the manner of execution, e.g., "following the road and maintaining a safe distance". **Maneuvers** are discrete actions, which an object can execute without a detailed specification, e.g., "turn right". **Trajectories** are the most detailed kind of prediction and describe an object's position over discrete time steps.

Additional Slides

Interfaces

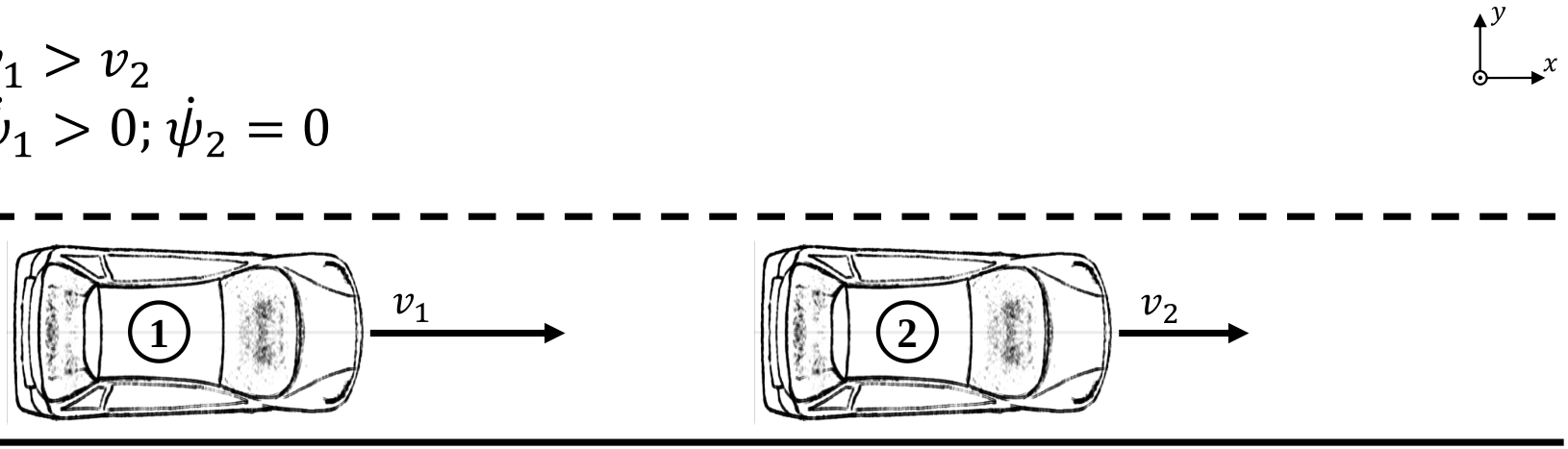
The prediction module gets the current and past state of surrounding objects such as position and velocity as input from the perception module. Additionally, object specific features, e.g., type and size, can be passed. Furthermore, map data can be used as input to improve the semantical awareness of the prediction. Information about lanes and drivable areas enable the reduction of prediction uncertainty and hence, longer prediction horizon.

The output of the prediction module are forecasts in one or more of the previously introduced levels of abstraction to the planning module. The motion planner uses this input to plan a safe and feasible ego trajectory within the specified time horizon. To realize this task, a policy must be determined based on specific metrics such as collision probability, comfort and progress toward a destination and decisions regarding the next action to be taken based on this policy. Due to the previously mentioned interaction among traffic participants, motion planning and motion prediction are not independent modules. Finally, the control module determines actor signals to follow the planned trajectory and passes them to the actors.

Where is the car going?

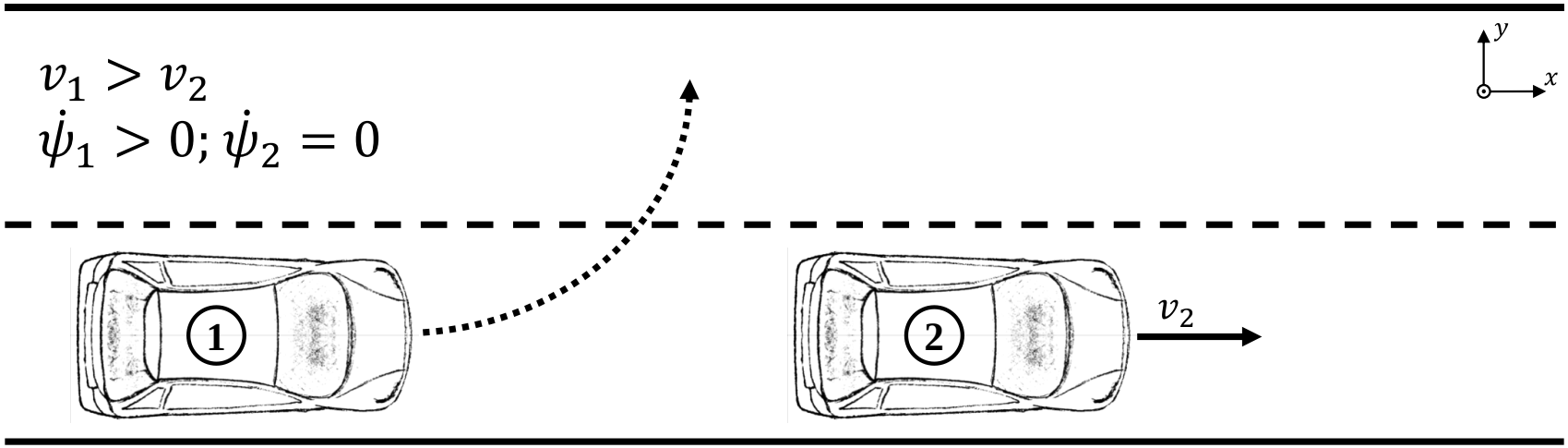
$v_1 > v_2$

$\dot{\psi}_1 > 0; \dot{\psi}_2 = 0$



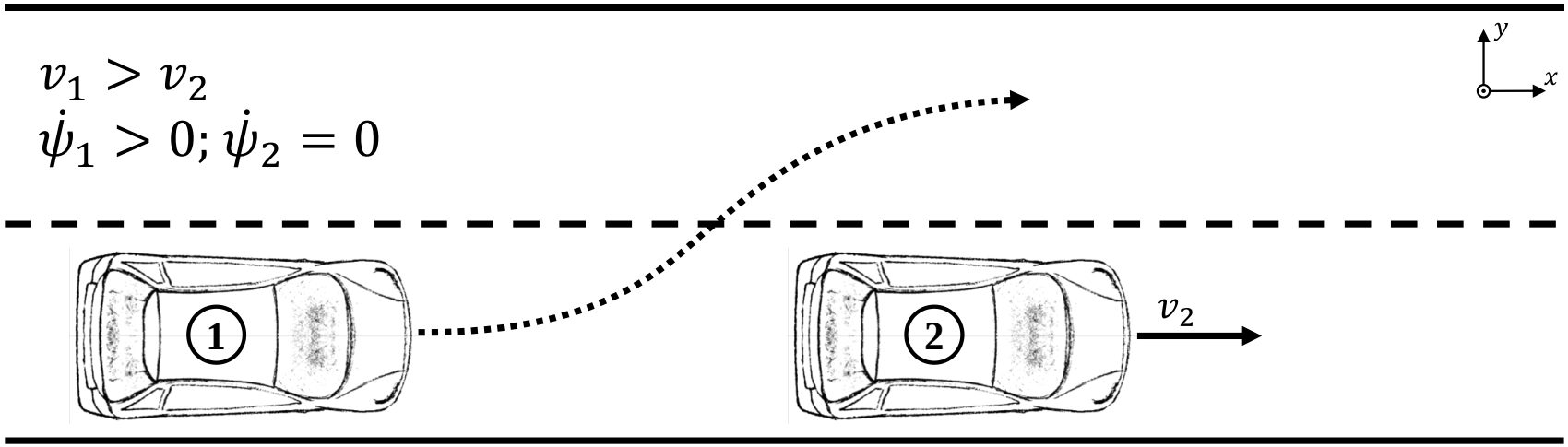
Where is the car going? – Physics-based

Assumption of constant velocity and yaw rate



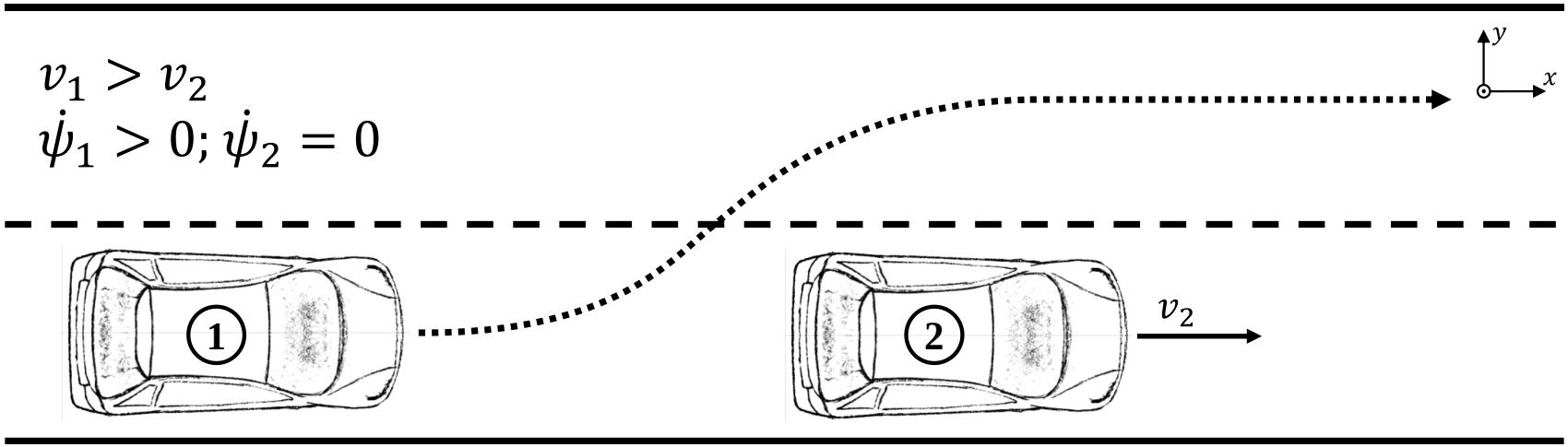
Where is the car going? – Pattern-based

Similar scenarios in the dataset resulted in overtaking



Where is the car going? – Planning-based

The behavior model assigns the lowest cost to „lane-change and acceleration“



Classes of Prediction Models

Class	Concept	Methods
Physics-based Prediction	Sense – Predict	• State Estimation • Reachable Sets
Pattern-based Prediction	Sense – Learn – Predict	• Clustering & Classification • Deep Learning Patterns
Planning-based Prediction	Sense – Reason – Predict	• Learning from Demonstration • Planning under Uncertainty

Prediction
Prof. Dr. Markus Lienkamp

Phillip Karle, M. Sc.

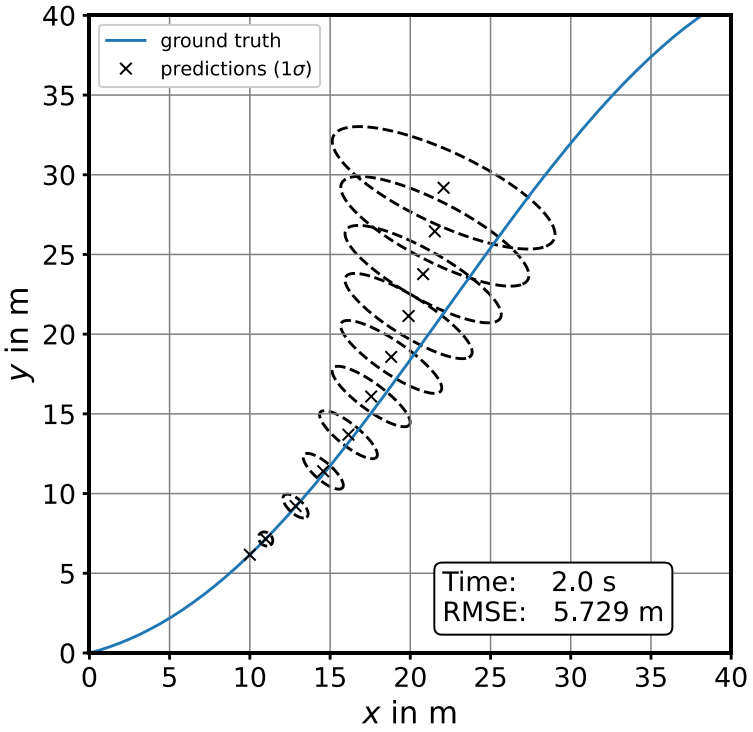
Agenda

1. Foundations
2. **Physics-Based Prediction**
   a. **State Estimation**
   b. **Reachable Sets**
3. Pattern-Based Prediction
4. Planning-Based Prediction
5. Summary and Outlook

Classes of Prediction Models

Class	Concept	Methods
Physics-based Prediction	Sense – Predict	• State Estimation • Reachable Sets
Pattern-based Prediction	Sense – Learn – Predict	• Clustering & Classification • Deep Learning Patterns
Planning-based Prediction	Sense – Reason – Predict	• Learning from Demonstration • Planning under Uncertainty

Physics-Based Prediction: Approaches



State Estimation

State Estimation

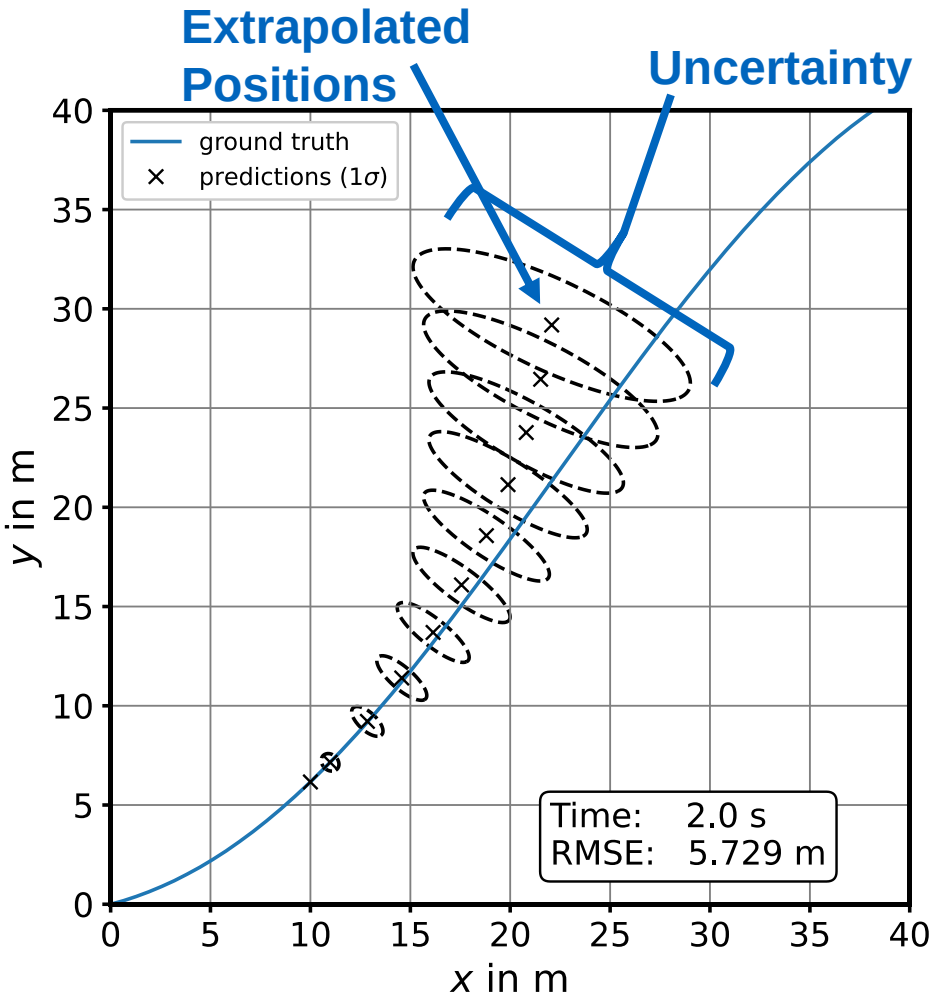
Idea

- Prediction of future positions by physics-based models
- Quantification of uncertainty by means of Bayesian filter

Output: Trajectory Prediction

Application

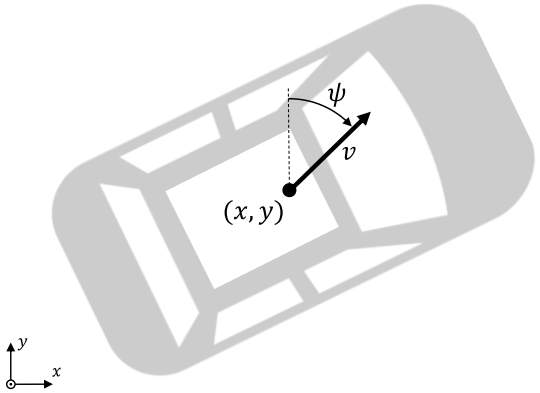
- Short-Term Prediction
- Collision Check



State Estimation: State-Space Model

Discrete State-Space Model

- Initial State $\boldsymbol{x}_{t_0} = \boldsymbol{x}_0$
- State Variables $\boldsymbol{x}$
- Input $\boldsymbol{u}$
- Linear Case: $\boldsymbol{x}_{t+1} = \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{B}\boldsymbol{u}_t$
- General Case: $\boldsymbol{x}_{t+1} = \boldsymbol{F}(\boldsymbol{x}_t, \boldsymbol{u}_t, \Delta t)$



Approach

- Laws of Mechanics with Simplifications
- Bayesian Filter for Uncertainty Estimation

State Estimation: State-Space Model

Laws of Mechanics with Simplifications

Kinematic Models

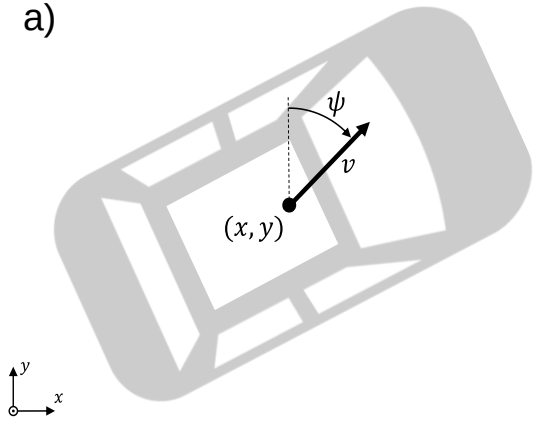
→ Point-mass model, a)

Dynamic Models

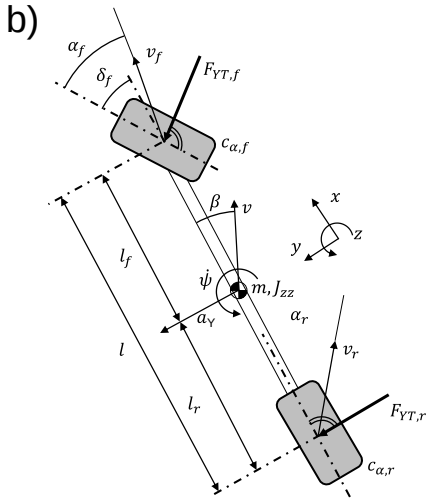
→ Bicycle Model, b)

→ Four-Wheel Model, c)

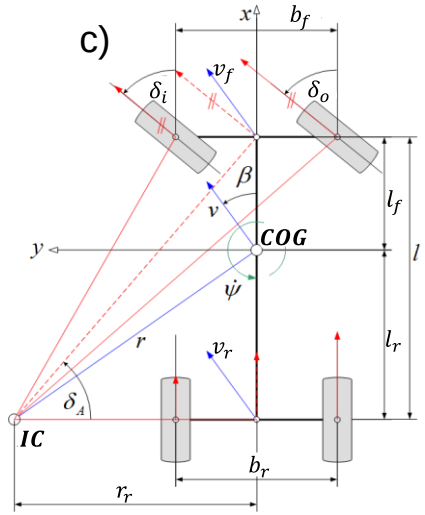
a)



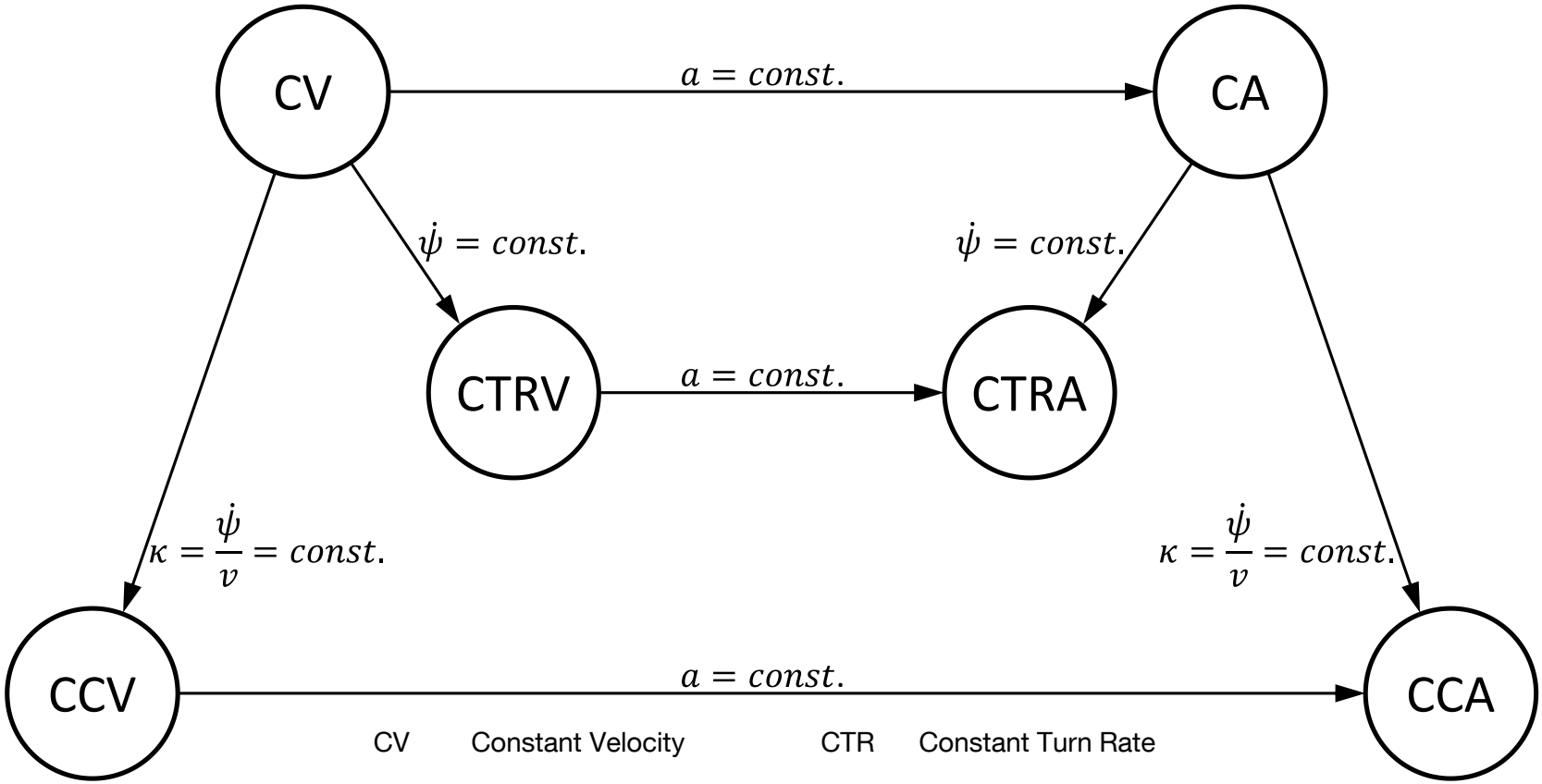
b)



c)



State Estimation: Kinematic Models – Overview

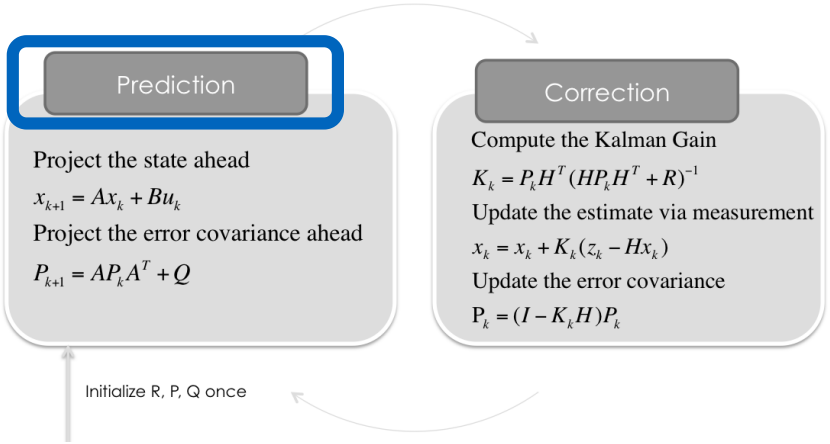


CV	Constant Velocity	CTR	Constant Turn Rate
CC	Constant Curvature	CA	Constant Acceleration
$\dot{\psi}$	Turn Rate	κ	Curvature
a	Acceleration	v	Velocity

State Estimation

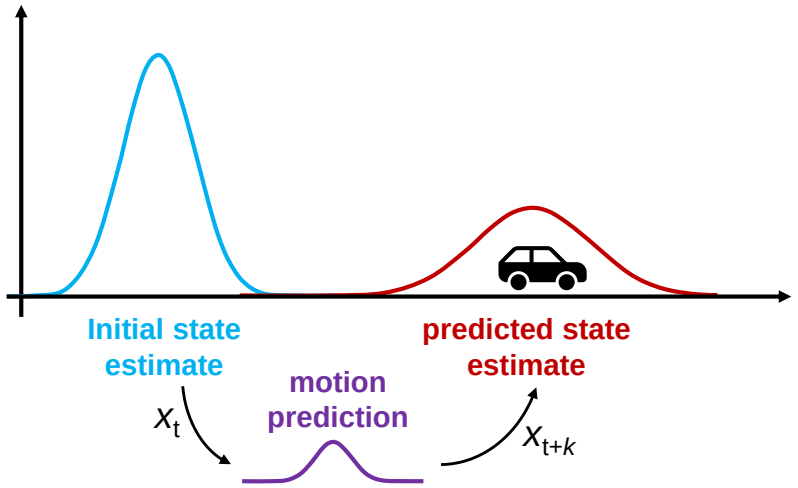
Bayesian Filter for Uncertainty Estimation

- Kalman Filter
- Extended Kalman Filter
- Unscented Kalman Filter
- Particle Filter



For Motion Prediction

→ Apply Prediction step without measurement update



Additional Slides

State Estimation

State estimation-models apply motion equations from classical mechanics. The description of object's motion can be by either dynamics or kinematics. In the event of a dynamic model, the motion is a result of the lateral and longitudinal tire forces. The vehicle is described using a four wheel [1, 2] or a bicycle model [3–8]. Pepy et al. [9] compare the proposed variants of dynamic models for the navigation task. In general, dynamic models are used for motion control rather than prediction. An overview of uncertainty and prediction in motion control is provided by Carvahlo et al. [6].

Kinematic models are preferred when predicting motion in objects because (1) sensors such as camera or LIDAR cannot measure the required input data for dynamic models and (2) there is no need to obtain such detailed vehicle dynamics information as prediction output. Instead, it is valid to make assumptions to simplify the state estimation of dynamic objects. For kinematic models, the vehicle is reduced to a point mass.

Image References:

Lecture Notes "Vehicle Dynamics of Passenger Cars", Institute of Automotive Technology, TUM

R. Schubert, E. Richter, and G. Wanielik, "Comparison and evaluation of advanced motion models for vehicle tracking," 2008 11th International Conference on Information Fusion, pp. 1–6, 2008.

https://github.com/balzer82/Kalman

State Estimation: Filter Types

Kalman Filter

- Optimal Solution for linear models and gaussian process and measurement noise

Extended Kalman Filter

- 1st order approximation of nonlinear models with Taylor expansion
- Assumption of gaussian noise

\+ Easy to implement
\+ Intuitive tuning
\+ Works well on weak non-linearization
\- Expensive Jacobi-Matrix calculation

Prediction Step

$$\boldsymbol{x}_{t+1} = \boldsymbol{F}_A(\boldsymbol{x}_t, \boldsymbol{u}_t)$$
$$\boldsymbol{y}_{t+1} = \boldsymbol{F}_C(x_{t+1})$$
$$\boldsymbol{P}_{t+1} = \boldsymbol{J}_A \boldsymbol{P}_t \boldsymbol{J}_A^T + \boldsymbol{Q}_t$$

Update Step

$$\boldsymbol{K} = \boldsymbol{P}_t \boldsymbol{J}_C^{\mathrm{T}} \left(\boldsymbol{J}_C P \boldsymbol{J}_C^{\mathrm{T}} + \boldsymbol{R}\right)^{-1}$$
$$\boldsymbol{x} = \boldsymbol{x} + \boldsymbol{K}(\boldsymbol{y} - \boldsymbol{F}_C(\boldsymbol{x}))$$
$$\boldsymbol{P} = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{J}_C)\boldsymbol{P}$$

Jacobian Matrix

$$\boldsymbol{J}_k = \frac{\partial y_i}{\partial x_j}$$

State Estimation: Filter Types

Unscented Kalman Filter

- Julier, Uhlmann, 1997 [14]
- True nonlinear modelling with gaussian noise
- 3rd order-accuracy for any nonlinearity with gaussian noise

\+ True nonlinear models

\+ Same magnitude of computational effort like EKF

\- Hyperparameters are less intuitive to tune

Parameters

$$L, \lambda, \alpha, \kappa, \beta$$

Weights

$$W_{i,m}, = f(\lambda, , L);\ W_{i,c} = f(\lambda, L, \beta)$$

Prediction Step

$$\mathbf{X}_{i,t} = \left[\boldsymbol{x}_t \quad \boldsymbol{x}_t \pm \left(\sqrt{(L+\lambda)\boldsymbol{P}_t}\right)_i\right]$$

$$\mathbf{X}_{i,t+1} = \boldsymbol{F}_A(\mathbf{X}_i)$$

$$\boldsymbol{x}_{t+1} = \sum_i^{2L} W_{i,m}\mathbf{X}_{i,t+1}$$

$$\boldsymbol{P}_{t+1} = \sum_i^{2L} W_{i,C}(\mathbf{X}_{i,t+1} - \boldsymbol{x_{t+1}})(\mathbf{X}_{i,t+1} - \boldsymbol{x_{t+1}})^T$$

Additional Slides

Unscented Kalman Filter

EKF uses one point (the mean) for prediction. In contrast, the UKF uses not only the mean, but a symmetric sample of points which have the same mean and distribution as the prior $\boldsymbol{x}_t, \boldsymbol{P}_t$. This sample of points are called sigma points. These points are processed through the nonlinear model. The prediction (posterior) x_{t+1} is calculated by the weighted sum of the processed sigma points. The prediction of the covariance is derived from the weighted sum over the difference between the processed sigma points and the posterior. This approach is called Unscented Transformation (UT). The numerical derivation in the EKF-equations is replaced by the sampling of the sigma points and their weighted summation.

Hyperparameters of the Unscented Transformation:

L: Dimension of the state variables, formula: $L = \dim(x)$

λ: Scaling Parameter, formula: $\lambda = \alpha^2(L + \kappa) - L$

α: Spread parameter of the sigma points around the mean, typically small value, e.g. $a = 0.001$

κ: Secondary scaling parameter, usually set to zero

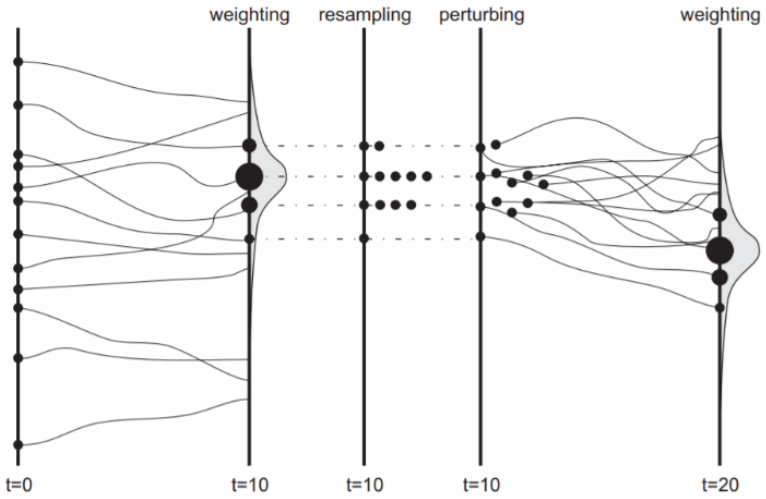
β: Prior distribution parameter, for Gaussian distribution, $\beta = 2$

References: [14, 15]

State Estimation: Particle Filter

Monte Carlo Method

- Set of particle (weights + object states)
- Process to new state (motion model)
- Weight update: likelihood to match measurement
- Get new distribution: Weight times states
- Sample form new distribution



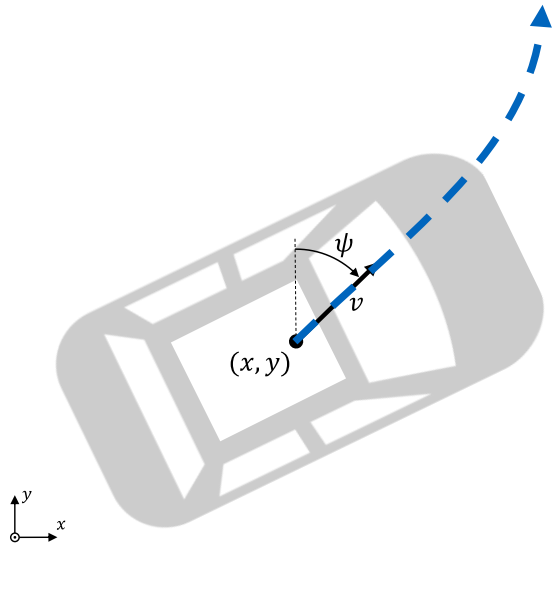
\+ Can handle any type of system and noise

– Might become computationally expensive / inefficient

State Estimation – Example

State-Space Model: CTRA

$$\begin{pmatrix} x \\ y \\ \Psi \\ v \\ \dot{\Psi} \\ a \end{pmatrix}_{t+k} = \begin{pmatrix} x + \frac{v}{\dot{\Psi}}\left(\cos\left(\dot{\Psi}\Delta t + \Psi\right) - \cos(\Psi)\right) \\ y + \frac{v}{\dot{\Psi}}\left(\sin\left(\dot{\Psi}\Delta t + \Psi\right) - \sin(\Psi)\right) \\ \psi + \dot{\Psi}\Delta t \\ v + a\Delta t \\ \dot{\Psi} \\ a \end{pmatrix}_{t}$$



State Estimation: Extended Kalman Filter

$$\boldsymbol{x}_{t+1} = \boldsymbol{F}_A(\boldsymbol{x}_t)$$

$$\boldsymbol{P}_{t+1} = \boldsymbol{J}_A \boldsymbol{P}_t \boldsymbol{J}_A^T + \boldsymbol{Q}_t$$

Jacobian Matrix $\boldsymbol{J}_k = \frac{\partial y_i}{\partial x_j}$

Process Noise $\boldsymbol{Q}_t$

State Covariance $\boldsymbol{P}_t$

CTRA: Constant Turn Rate and Acceleration

State Estimation – Example

Prediction Horizon

- Time Steps $\{0, 0.1\text{s}, \dots, t_{\text{pred}} = 2.0\text{s}\}$
- Get $\boldsymbol{x}_{\text{pred}}$, $\boldsymbol{P}_{\text{pred}}$

$$\boldsymbol{x}_{\text{pred}} = \begin{pmatrix} x_{\text{p}} \\ y_{\text{p}} \end{pmatrix} = \begin{pmatrix} x_{t_1} \dots x_{t20} \\ y_{t_1} \dots y_{t20} \end{pmatrix}$$

$$\boldsymbol{P}_{\text{pred}} = (\boldsymbol{P}_{t_1} \quad \dots \quad \boldsymbol{P}_{t_{20}})$$

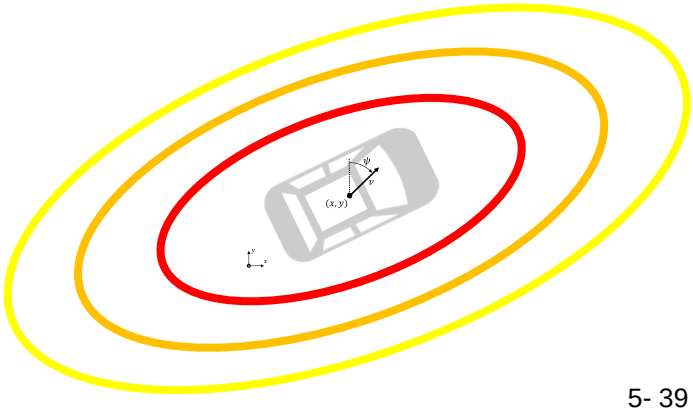
Evaluate Collision Probability

- Uncertainty Weighted Distance: Mahalanobis Distance
- χ^2-Distribution determines collision probability
- Collision probability has to stay below safety threshold D_{crit}

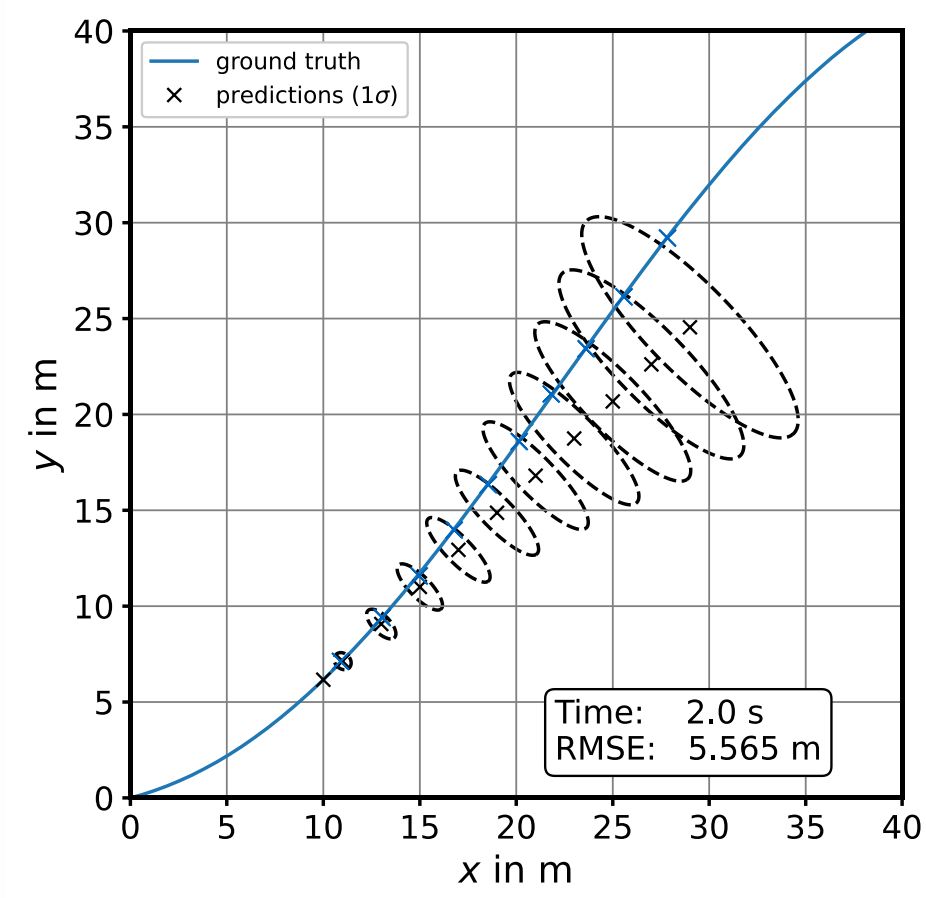
→ Ellipsoid safety region

$$D_{\text{MH}} = \sqrt{\left(\boldsymbol{x}_{t_i} - \boldsymbol{x}_{\text{ego}}\right)^{\text{T}} \boldsymbol{P}^{-1} \left(\boldsymbol{x}_{t_i} - \boldsymbol{x}_{\text{ego}}\right)}$$

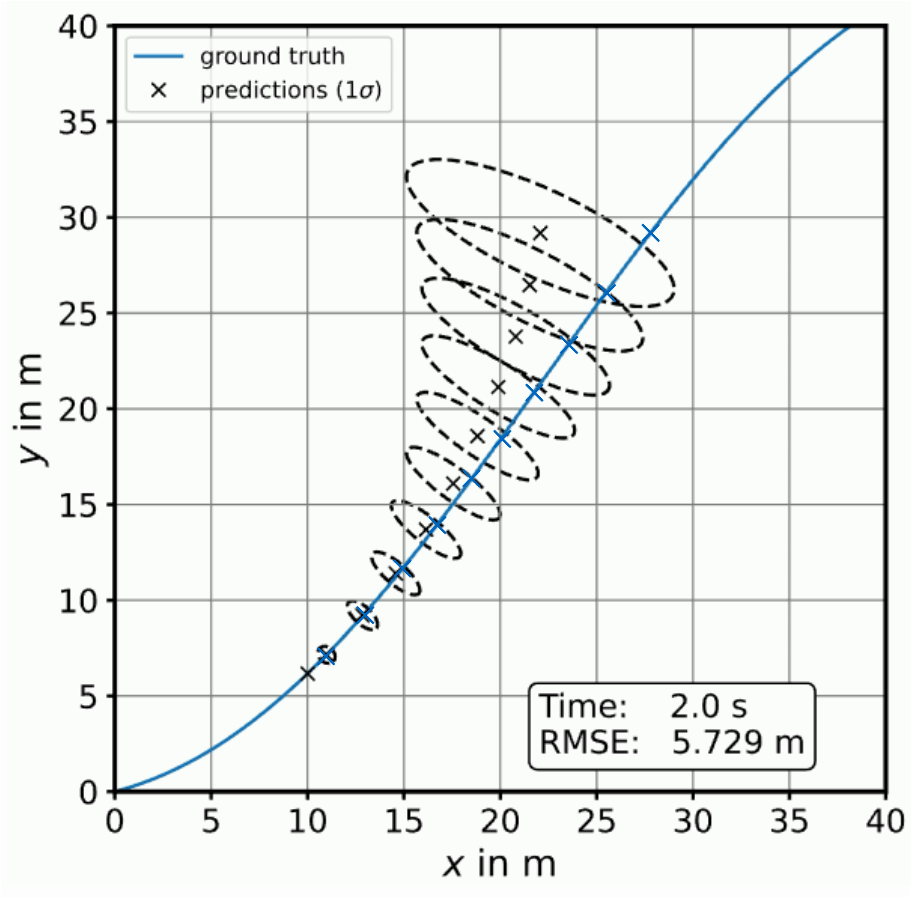
$$D < D_{\text{crit}} = \chi^2$$



State Estimation – Comparison



CV-Model



CTRA-Model

CV: Constant Velocity
CTRA: Constant Turn Rate and Acceleration
RMSE: Root Mean Square Error

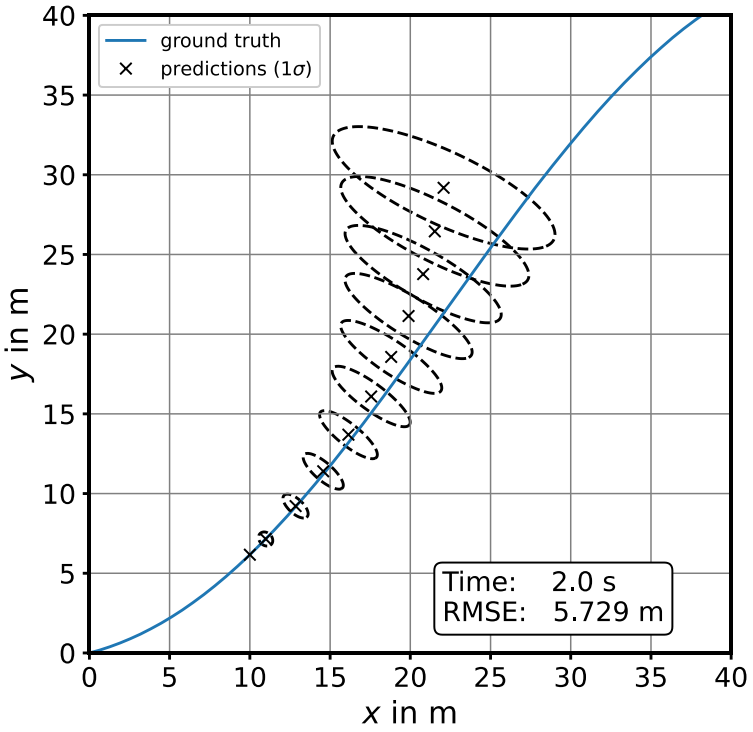
Additional Slides

Bayesian Filter

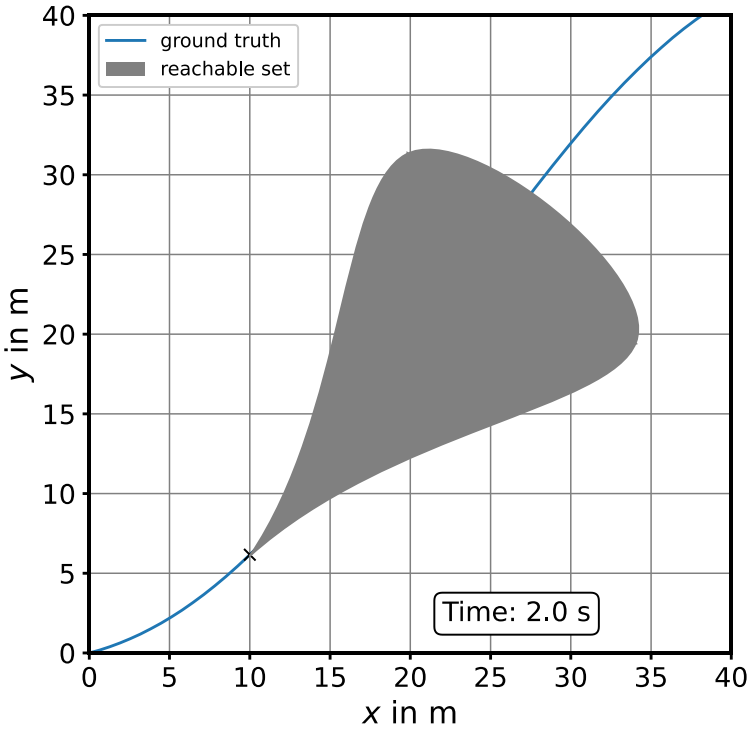
Motion prediction uncertainty can be considered by using a Bayesian filter. Using the Kalman filter (KF) [10] is valid if a normally distributed process noise and measurement noise and with a linear state-space model is assumed. Exemplary application for motion prediction are [3, 11, 12]. For non-linear kinematic models, the extended Kalman filter (EKF) [13] or the unscented Kalman filter (UKF) [14, 15] can be used, and both apply the same steps as the KF. Because the EKF approximates a nonlinear system by a first-order linearization, it offers poor accuracy when high nonlinearity occurs. In this case, the UKF is preferable since it estimates the future state distribution with deterministic sampling by using the unscented transformation (UT). Thus, the UKF is derivation-free and uses the true nonlinear state transition function $\boldsymbol{F}$. It captures the posterior mean and covariance to the third order [14]. Hence, the UKF offers superior performance at equal computational costs [15]. Exemplary applications are [5, 7] for the EKF and [16] for the UKF. YANG ET AL. [17] compare the performance of these two filters for vehicle navigation.

Particle filters are used to generalize Bayesian filtering for arbitrary models and random noise distribution, which make use of the sequential Monte Carlo method (SMC) [18]. Applications for motion prediction are given in [19–21].

Physics-Based Prediction: Approaches



State Estimation



Reachability Analysis

Reachability Analysis

Idea

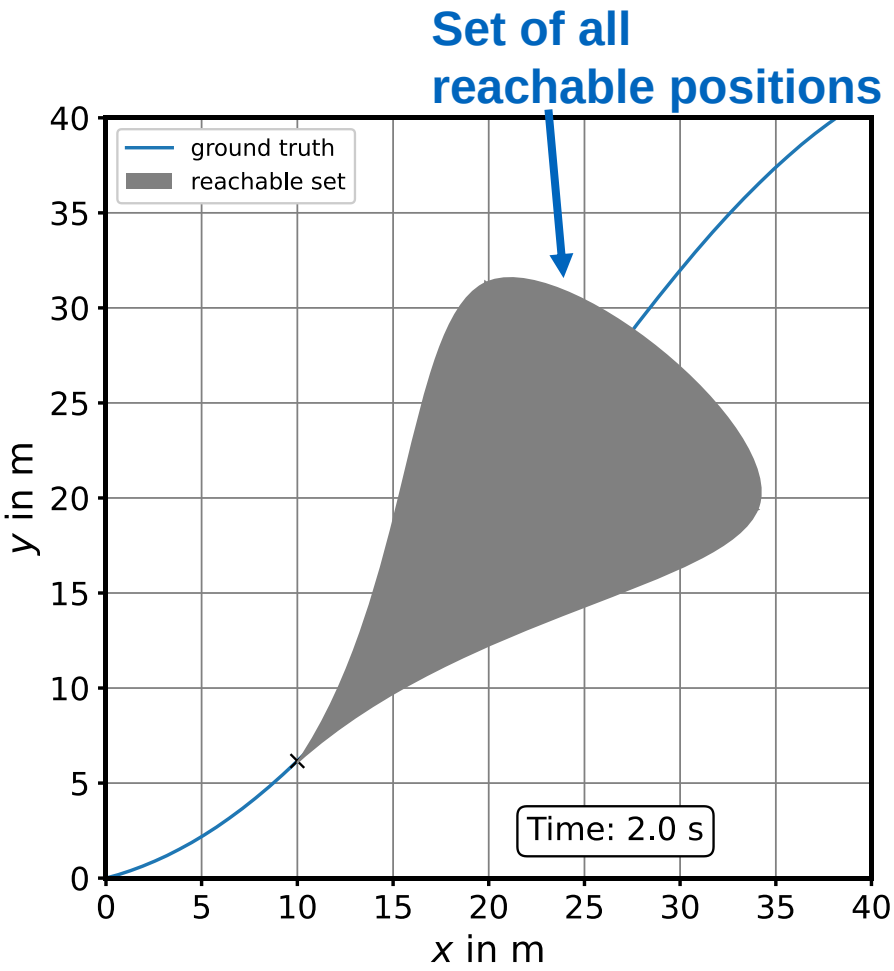
- Derivation of all possible positions within physical constraints
- Over approximation of reachable positions
- Application of Traffic Rules to shrink set size

Output

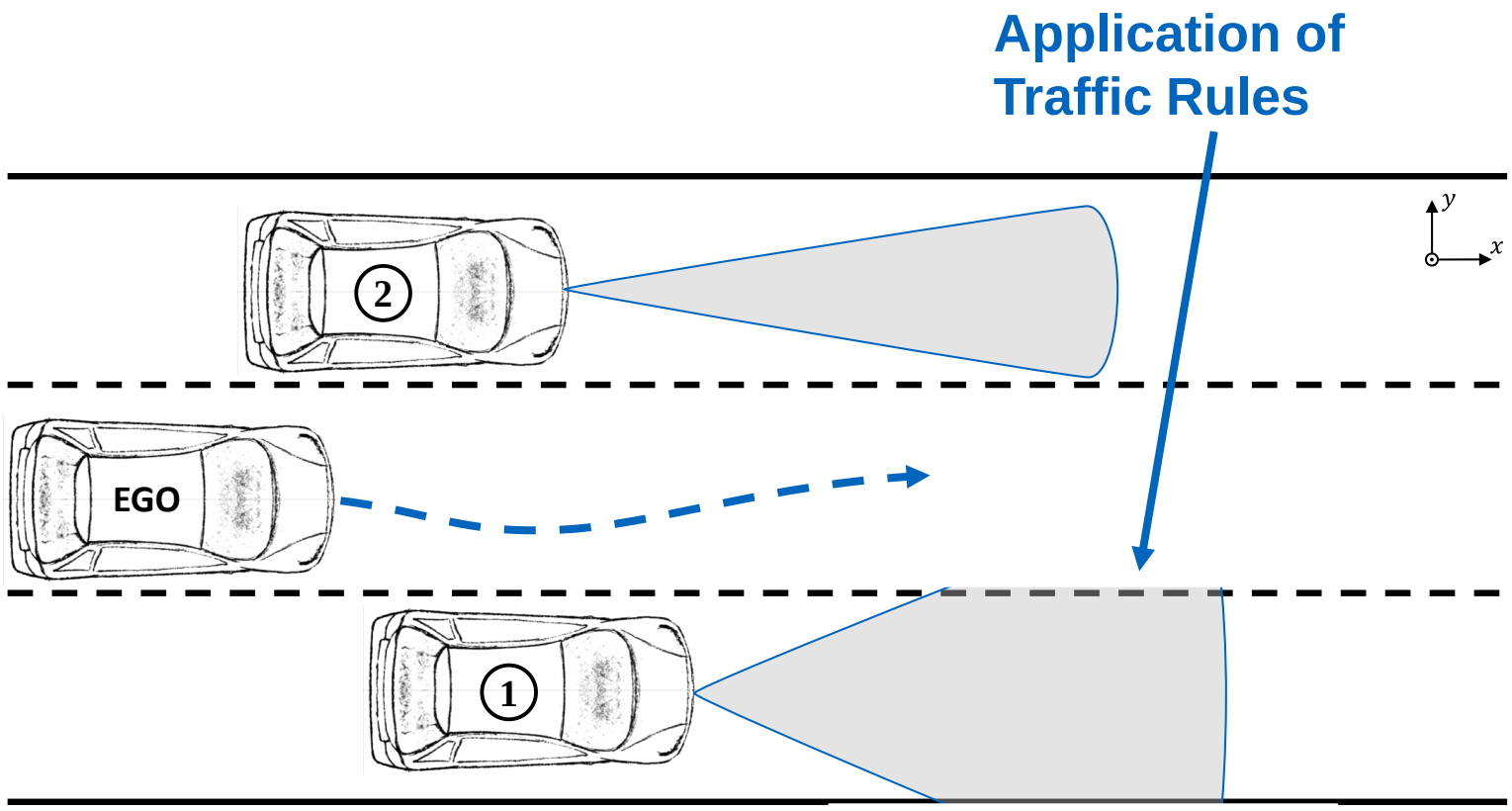
- Occupancy Map

Application

- Online Verification
- Safety Assessment



Reachable Sets



Additional Slides

Reachability Analysis:

Another approach of physics-based motion prediction is to evaluate all possible states that the object can reach within a given time horizon. This is done by defining constraints for the dynamic. The resulting solution space is called the reachable set [22–24]. To reduce the size of the set dynamic, constraints such as the kinematic simplifications [25] can be applied. Additionally, nonholonomic constraints can be taken into account so that a car cannot move arbitrarily in space [26]. The reachable sets can be over-approximated using convex shapes [22] in order to consider errors due to assumptions that are made and to simplify the shape of the set for further process steps. Since the reachable set grows strongly with the prediction horizon and thus occupies a great deal of space, the set can be described stochastically with a probability distribution [22]. The resulting probability of presence can be considered for motion planning with the corresponding risk assessment, instead of strictly taking the occupancy of the area into account. The consequence of the stochastic formulation is that the collision avoidance is not guaranteed anymore.

Prediction
Prof. Dr. Markus Lienkamp

Phillip Karle, M. Sc.

Agenda

1. Foundations
2. Physics-Based Prediction
3. **Pattern-Based Prediction**
   a. **Clustering and Classification**
   b. **Deep Learning Patterns**
4. Planning-Based Prediction
5. Summary and Outlook

Classification of Prediction Models

Class	Concept	Methods
Physics-based Prediction	Sense – Predict	• State Estimation • Reachable Sets
Pattern-based Prediction	Sense – Learn – Predict	• Clustering & Classification • Deep Learning Patterns
Planning-based Prediction	Sense – Reason – Predict	• Learning from Demonstration • Planning under Uncertainty

Additional Slides

Pattern-based Prediction

Based on the observed state of an object, the idea of pattern-based prediction models is to find the most likely motion pattern and to derive the prediction based on this pattern. The dynamic state, object specific features like type and size or environmental features like road type, drivable areas and traffic signs are commonly used to find and match patterns. Furthermore, the patterns are not limited to one specific object. It is also possible to derive patterns from a whole traffic situation, which is beneficial regarding the consideration of interaction between the different objects. In general, pattern-based prediction models strongly depend on data to learn patterns.

The output of pattern-based prediction is either explicit trajectories or less-specific high-level maneuvers. High-level maneuvers are useful for long-term prediction (>3 s) regarding occupying certain areas for ego-motion planning. By matching predicted intentions and map features, the precision can be improved. However, for exact risk estimation and collision checks, explicit trajectory prediction is necessary. Depending on the underlying method, the generated patterns can be intuitive, e.g., prototype trajectories or a black box when deep learning models are used.

Pattern Clustering

Idea

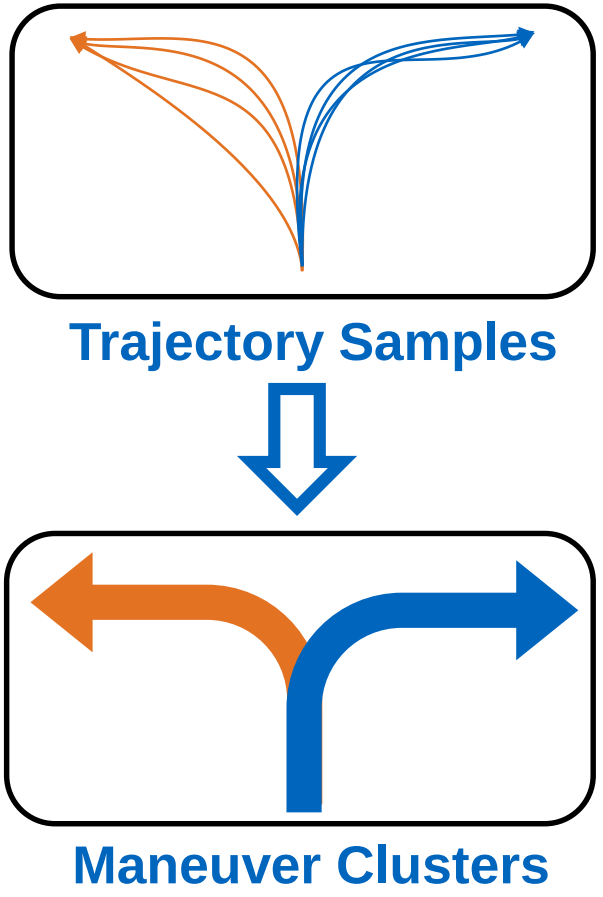
- Clustering of trajectories into few data classes
- Cluster data into maneuvers

Output

- Maneuver Cluster
- Prototype Trajectory

Application

- Input for classification-based prediction methods
- Dimension Reduction

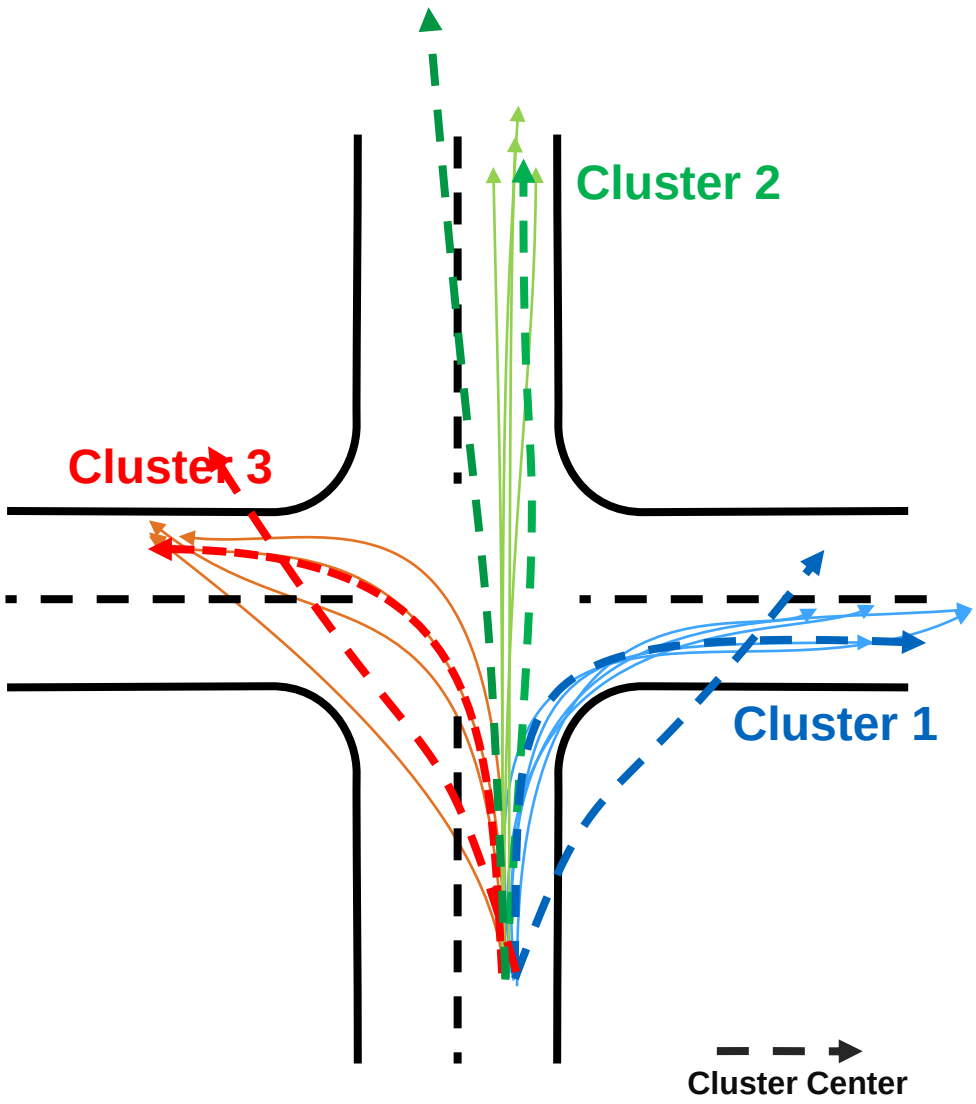


Pattern Clustering

K-means
Top-Down Clustering

Algorithm

- k initial random points
- Associate each sample to a center point
- Update new center point as mean of associated points
- Iterate until convergence

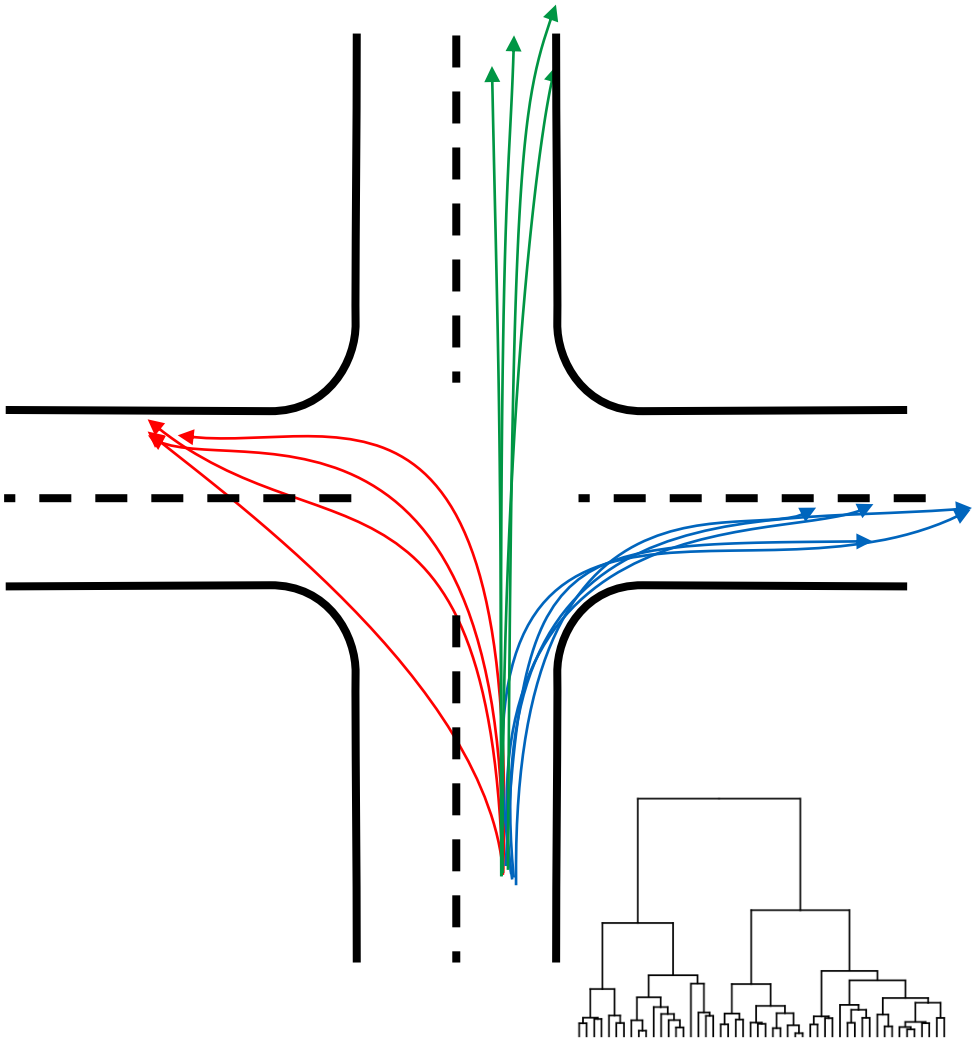


Pattern Clustering

Agglomerative Clustering: Bottom-Up

Algorithm

- Each point is one cluster
- Two nearest clusters are combined with new center point
- Distance between clusters based on different metrics (mean, max, min, etc.)
- Iterate until the distance between each cluster surpass the predefined distance



Additional Slides

Pattern Clustering

Pattern clustering aims to identify prototypical trajectories as a set of representative motion primitives of an object. These prototypes are extracted from tracked data. When applied to prediction tasks, the most probable prototypical trajectory based on observed features is outputted. Hence, in unknown scenarios, it is possible that no trajectory with high probability can be found. Two common clustering methods for this application are k-means and agglomerative hierarchical clustering. These methods are unsupervised and hence no labeled data is required [27].

Bian et al. [28] give a comprehensive survey of supervised and unsupervised algorithms for trajectory clustering, distance metrics and data pre-processing.

Figure References:

Junfeng Zhang, Wei Chen, Mingyi Gao, and Gangxiang Shen, "K-means-clustering-based fiber nonlinearity equalization techniques for 64-QAM coherent optical communication system," Opt. Express 25, 27570-27580 (2017)

https://www.datanovia.com/en/lessons/agglomerative-hierarchical-clustering/

Pattern Classification

Idea

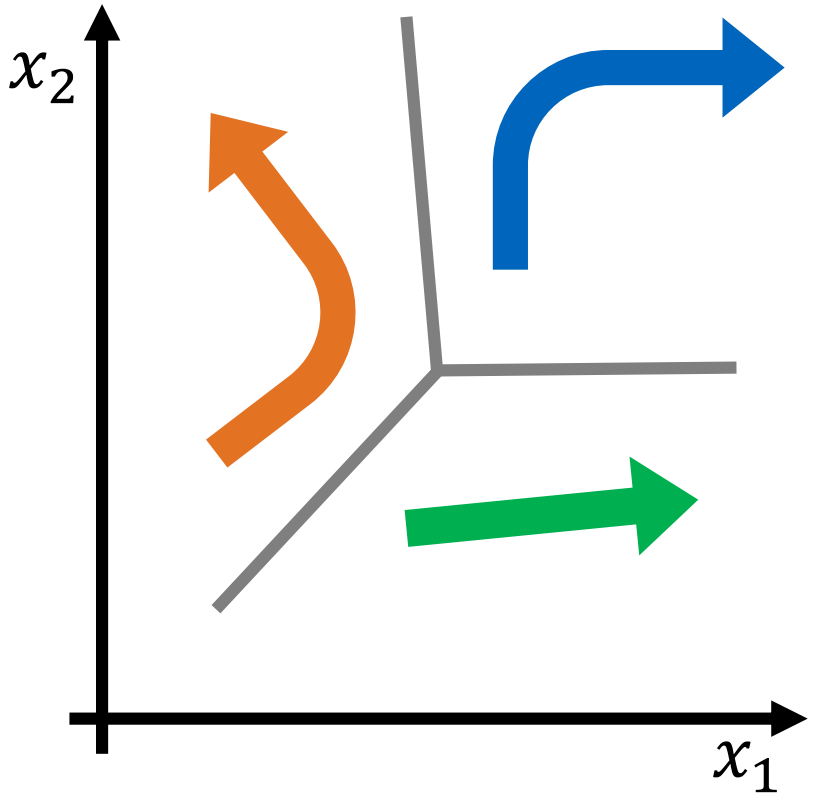
- Classification of new received trajectory into pre-defined classes
- A priori knowledge of outcome: Labeled data necessary for training

Output

- Discrete maneuver classification

Application

- Maneuver Prediction
  - Lane Change
  - Intention at intersections
- Interactive Planning (Game Theory)



x_i: features

Additional Slides

Patten Classification

Instead of clustering unlabeled data and matching observations to learned motion primitives, pattern classification methods tend to categorize observations into predefined maneuver groups by means of specific features. Hence, there is a strong dependency between the robustness of the algorithm and the definition of relevant features and maneuver classes. Support Vector Machines (SVM) [29] and Hidden Markov Models (HMM) are two classification algorithms that are used in the context of maneuver estimation. The SVM as discriminative algorithm is primarily used for lane change detection [30–32] or maneuver prediction at intersections [33]. Selected features are lateral position or distance to the center line, steering angle and respective derivations [30, 32]. Since object motions are sequential, the feature vector must also consider the temporal dimension, which is realized by sliding windows [30, 32]. Ward et al. [34] compare the SVM and further methods, namely, the k-nearest neighbor (k-NN) and random forest (RF), to classify driver maneuvers at intersections. A similar comparison is drawn by Aoude et al. [35] between an SVM and an HMM which estimate driver behavior at intersections by classifying between compliant and violating traffic lights.

Classification by means of an HMM is beneficial in terms of directly modeling the temporal evolution. An HMM is a discrete Markov models, which have in common to describe a scenario with discrete states and according transition probabilities. Furthermore, all Markov models build upon the Markov property, which defines that only the current state is relevant for prediction [36].

Clustering vs. Classification

In general, in classification a set of predefined classes is given and the question is to which class a new object belongs to. Clustering tries to group a set of objects and find whether there is some relationship between the objects.

In the context of machine learning, classification is supervised learning and clustering is unsupervised learning.

Markov Models

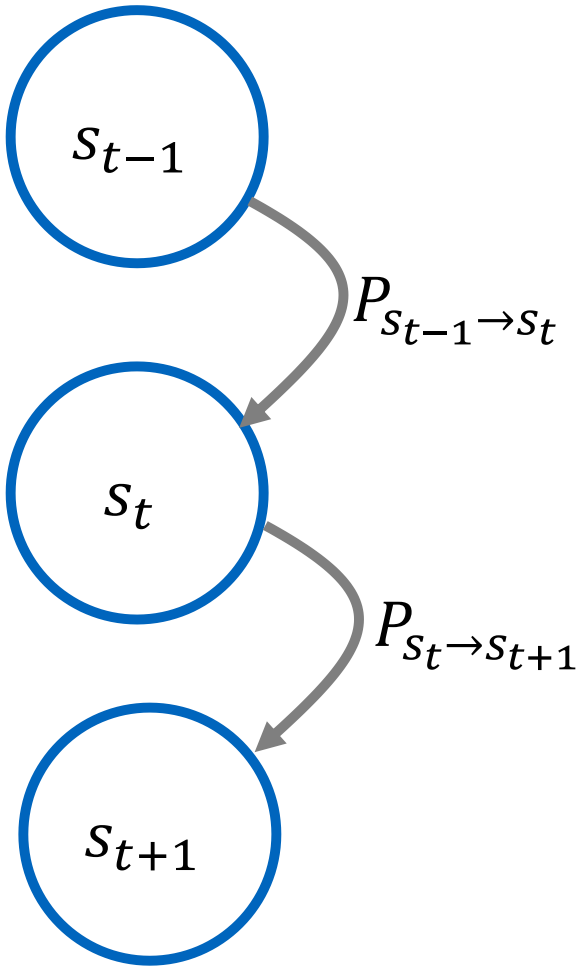
Idea

- Stochastic model of discrete processes
- Markov Assumption: memoryless property of a stochastic process

Definition

A stochastic process is a Markov process if the conditional probability distribution of future states of the process depends only upon the present state, not on the sequence of events that preceded it.

$$P(s_{t+1}|s_{t,t-1:1}) = P(s_{t+1}|s_t);\ \forall t$$



Transition Probability *P*
State *s*

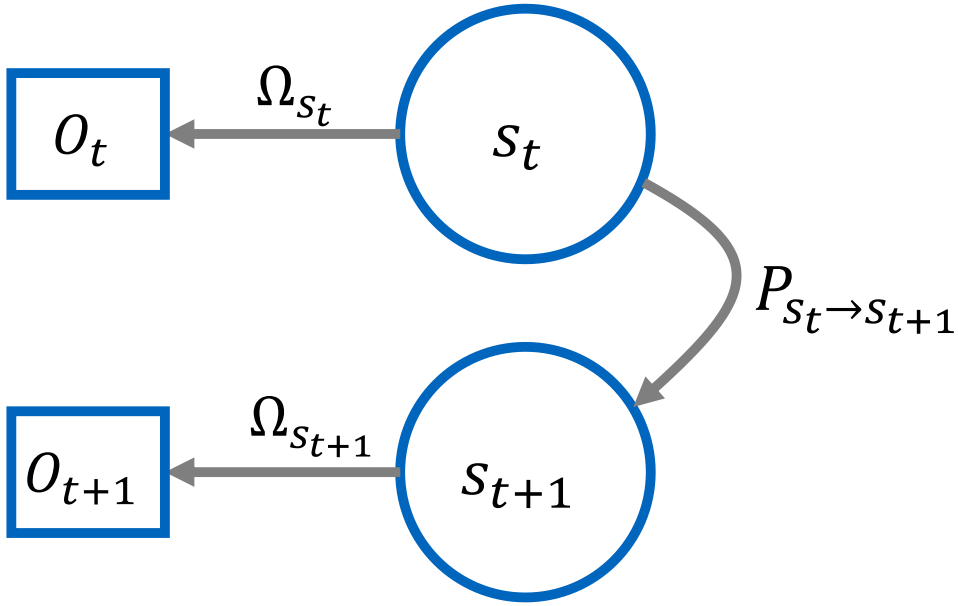
Markov Models

	The current state is known	The current state is estimated
Actions are not controllable by the agent in every state	Markov Process (MP)	Hidden Markov Model (HMM)
Actions are controllable by the agent in every state	Markov Decision Process (MDP)	Partially observable Markov Decision Process (POMDP)

Hidden Markov Model

Definition “Hidden”

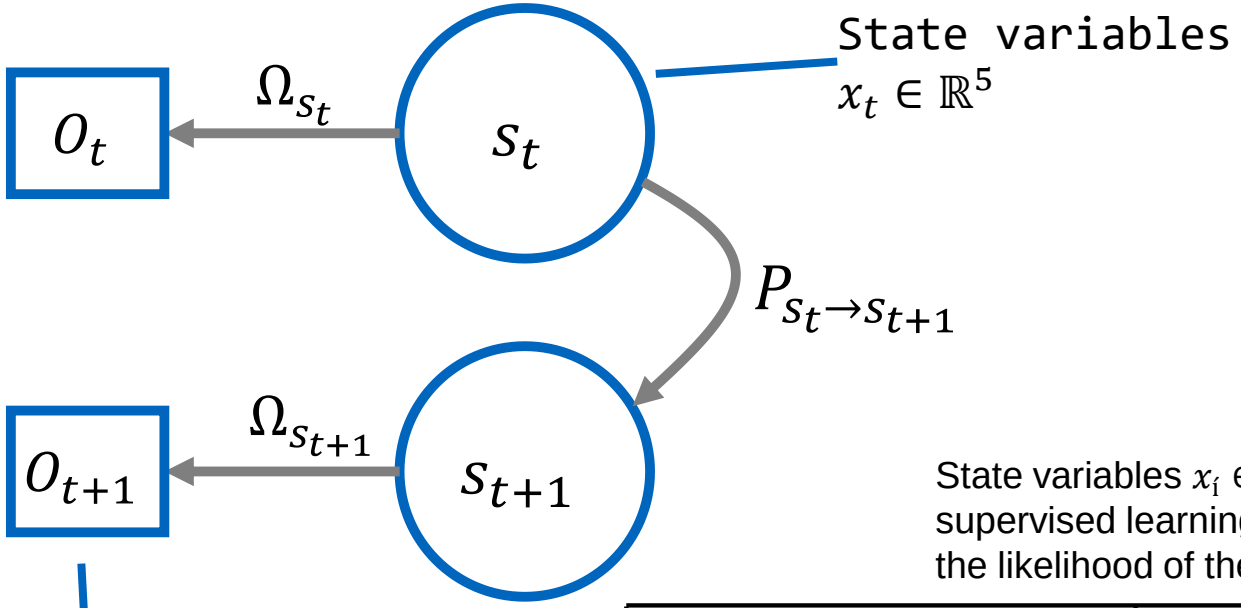
- States of the Markov process are unknown
- Only observations available



Model Parameters

- Tuple: (S, P, Ω, O)
- States $S = \{s_0, \ldots, s_n\}$
- Transition probability $P(s'|s)$
- Observation $O(t)$
- Observation model $\Omega(s)$

HMM Example – Driver‘s intention at intersection



State variables $x_i \in \mathbb{R}^5$ determined by supervised learning based on maximization the likelihood of the maneuver prediction

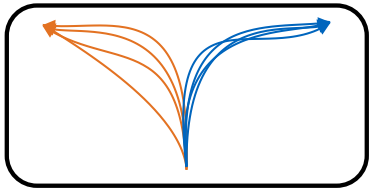
	Predicted direction		
Direction driven	left	right	straight
left	71	1	1
right	11	197	3
straight	1	11	537
mean prediction time	8.3s	4.7s	8.3s

T. Streubel and K. H. Hoffmann, “Prediction of driver intended path at intersections,” in *IEEE Intelligent Vehicles Symposium Proceedings*, 2014, pp. 134–139.

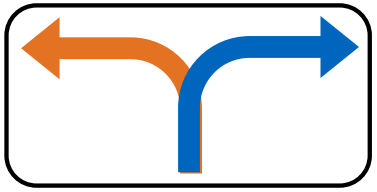
Clustering and Classification

+ Discrete, low-dimensional solution space
+ Applicable for structured environments, e.g. highway
+ Applicable for interactive planning (game theory)

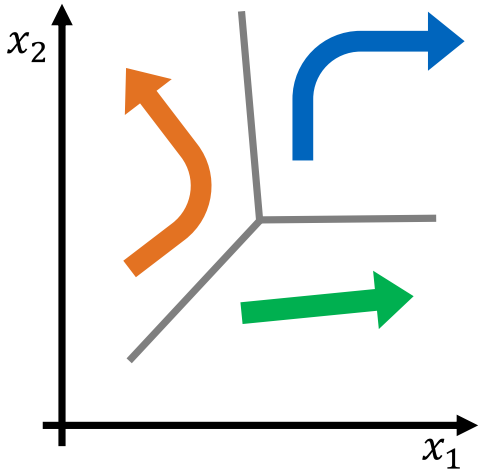
- Clustering: Number and shape of classes unknown
- Classification: high dependency on a priori class definition
- No trajectory prediction possible
- Complexity of road traffic is hard to cover by few classes



Trajectory Samples



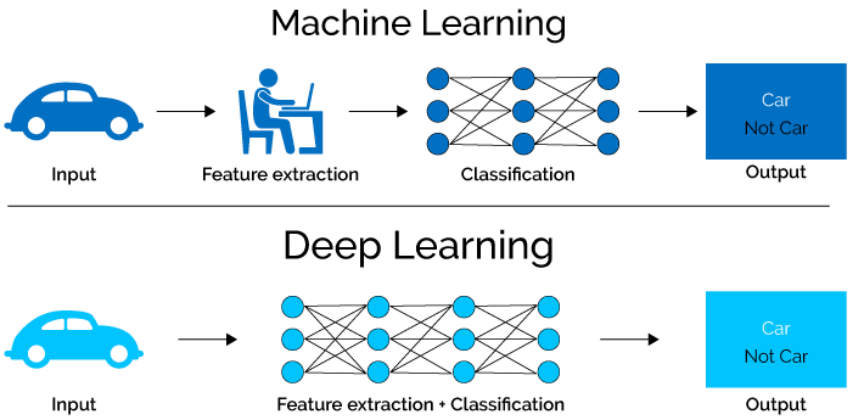
Maneuver Clusters



Deep Learning Patterns – Motivation

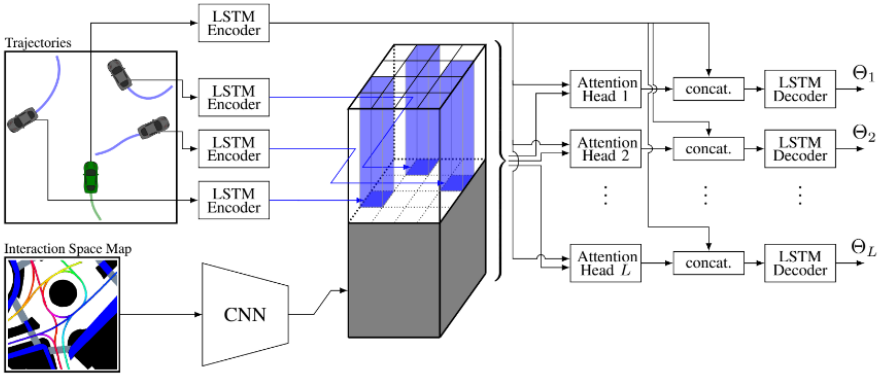
Why Deep Learning?

- Maximum utilization of data
- Creation of new features
  → But: Reason about valid input



Motion Prediction

Latest prediction challenges dominated by Deep Learning Methods with Encoder-Decoder-Architectures



Messaoud – Trajectory Prediction for Autonomous Driving based on Multi-Head Attention with Joint Agent-Map Representation

Deep Learning Patterns – Motivation

ICRA 2020 Prediction Challenge

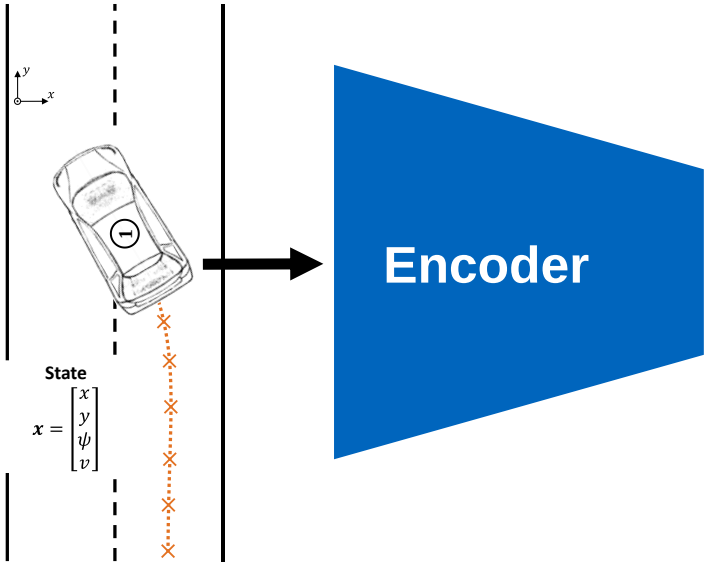
Three student-led teams took the winning prizes for the prediction challenge. All three used encoder-decoder architectures with novel input representations beyond rasterizing the scene context.
For the full leaderboard, see https://www.nuscenes.org/prediction.

minADE 5	Prize	Method	Authors	Affiliation
1.63	1st	cxx	Chenxu Luo	The Johns Hopkins University
1.813	2nd	MHA-JAM	Kaouther Messaoud, Nachiket Deo, Mohan Trivedi, Fawzi Nashashibi	INRIA Paris - RITS Team, UCSD - LISA Lab
1.877	3rd	Trajectron++	Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, Marco Pavone	Autonomous Systems Lab, Stanford University and Ford Greenfield Labs

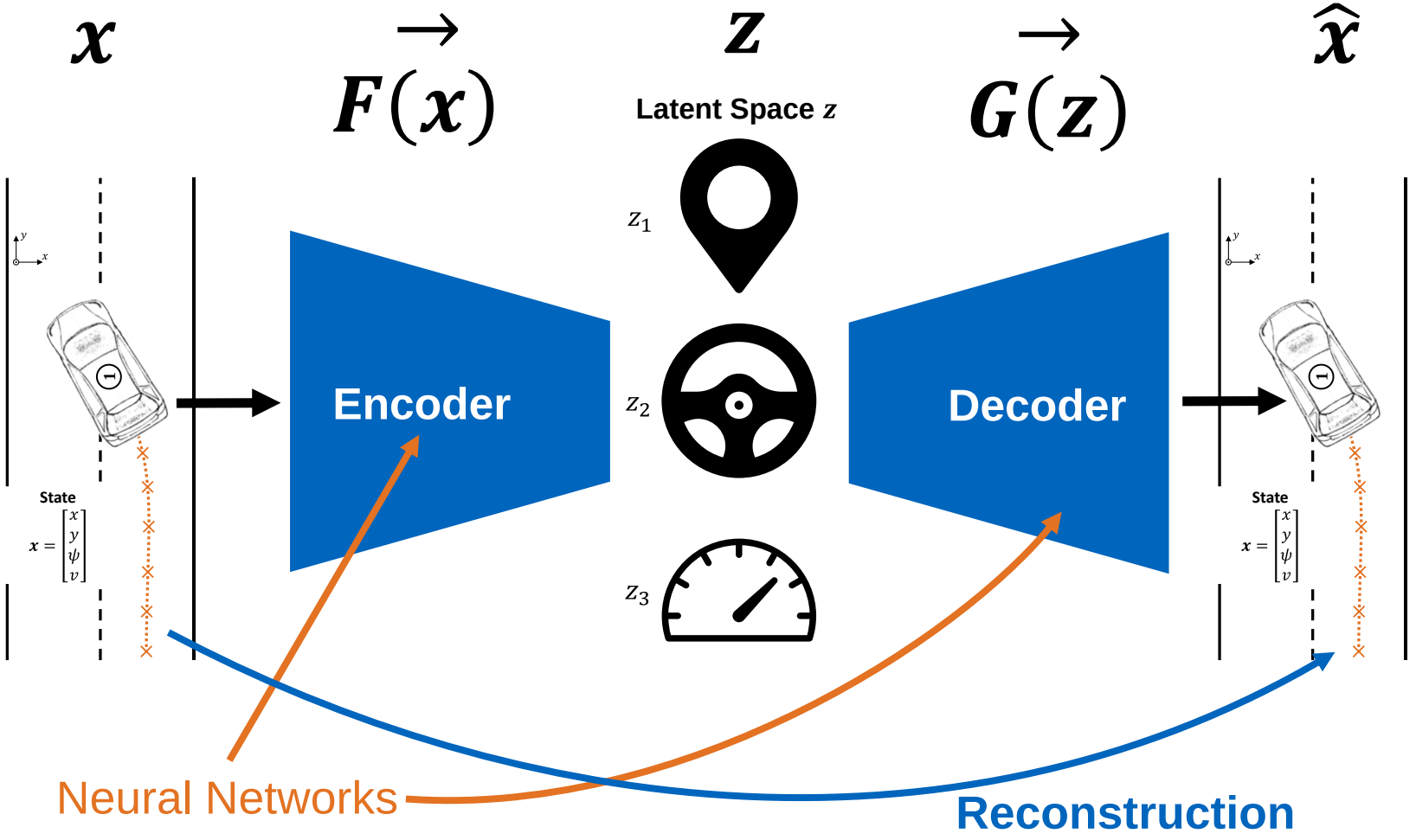
minADE: minimal Average Displacement Error

Deep Learning Patterns – Encoder-Decoder

$\boldsymbol{x}$ $\rightarrow$



Deep Learning Patterns – Encoder-Decoder



Deep Learning Patterns – Encoder-Decoder

Idea

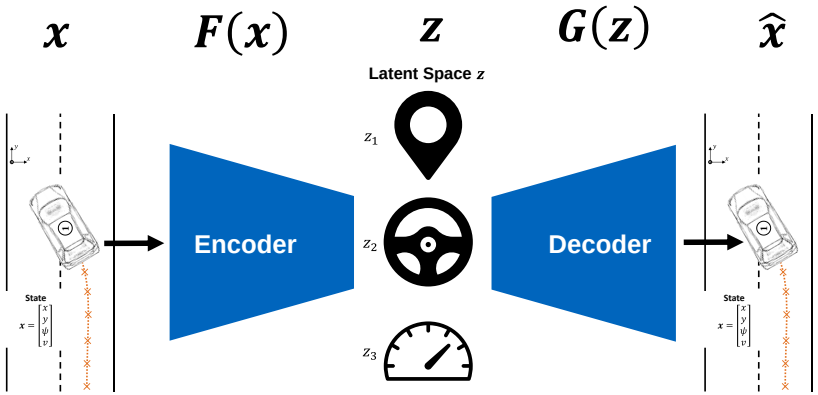
- Method to efficiently compress and encode data and how to decode and reconstruct it
- Reduction of data dimensions by learning how to ignore the noise in the data.

Latent Space

The “bottleneck” between in- and output and compressed representation of input data

Unsupervised Method

The compressed representation is unknown a priori



x:	Input
$\hat{x}$:	Reconstruction
$F(x)$:	Encode-Function
z:	Latent Space
$G(z)$:	Decode-Function
$\mathcal{L}(x, \hat{x})$:	Loss-Function

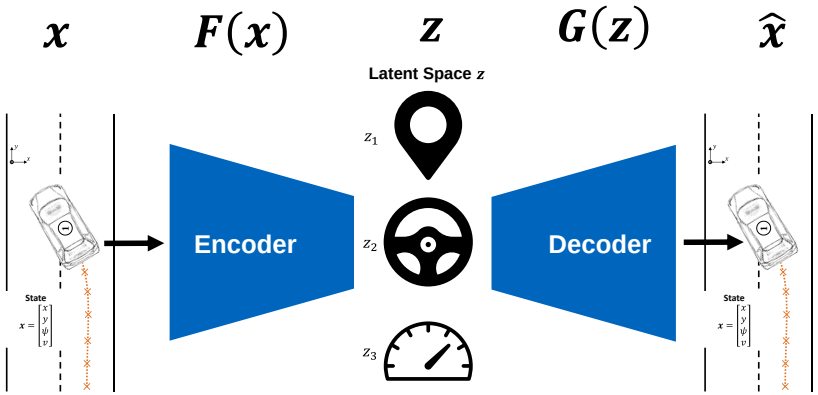
Deep Learning Patterns – Encoder-Decoder

Reconstruction Loss

$$\mathcal{L}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \mathcal{L}(\boldsymbol{x}, \boldsymbol{G}(\boldsymbol{F}(\boldsymbol{x})))$$

$\|.\|_2^2$-Norm:

$$\mathcal{L}_2(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \|\boldsymbol{x} - \boldsymbol{G}(\boldsymbol{F}(\boldsymbol{x}))\|_2^2$$



Regularized Autoencoder

Additional Loss Term to prevent the autoencoder to learn the identity function and to improve generalization.

$$\mathcal{L}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \|\boldsymbol{x} - \boldsymbol{G}(\boldsymbol{F}(\boldsymbol{x}))\|_2^2 + \text{Regularizer}$$

$\boldsymbol{x}$:	Input
$\hat{\boldsymbol{x}}$:	Reconstruction
$\boldsymbol{F}(\boldsymbol{x})$:	Encode-Function
$\boldsymbol{z}$:	Latent Space
$\boldsymbol{G}(\boldsymbol{z})$:	Decode-Function
$\mathcal{L}(\boldsymbol{x}, \hat{\boldsymbol{x}})$:	Loss-Function

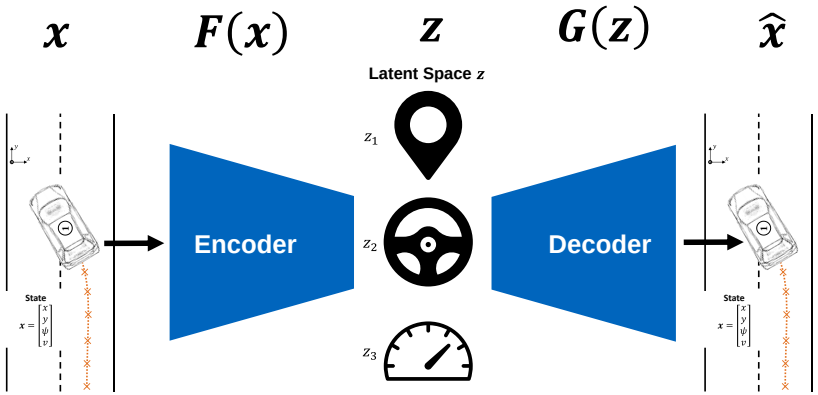
Deep Learning Patterns – Encoder-Decoder

Application

- Dimensional Reduction
- Features Extraction
- Image Denoising

- Sequence-To-Sequence-Prediction

  → **Generative Model**



$\boldsymbol{x}$: Input

$\hat{\boldsymbol{x}}$: Reconstruction

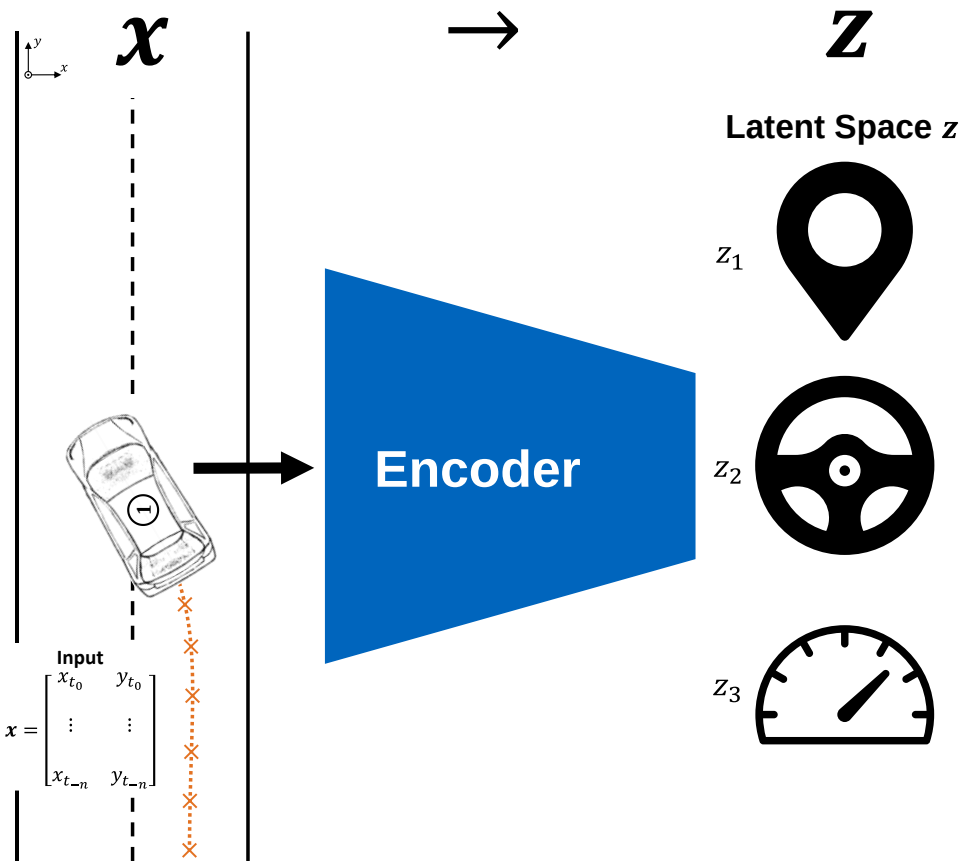
$\boldsymbol{F(x)}$: Encode-Function

$\boldsymbol{z}$: Latent Space

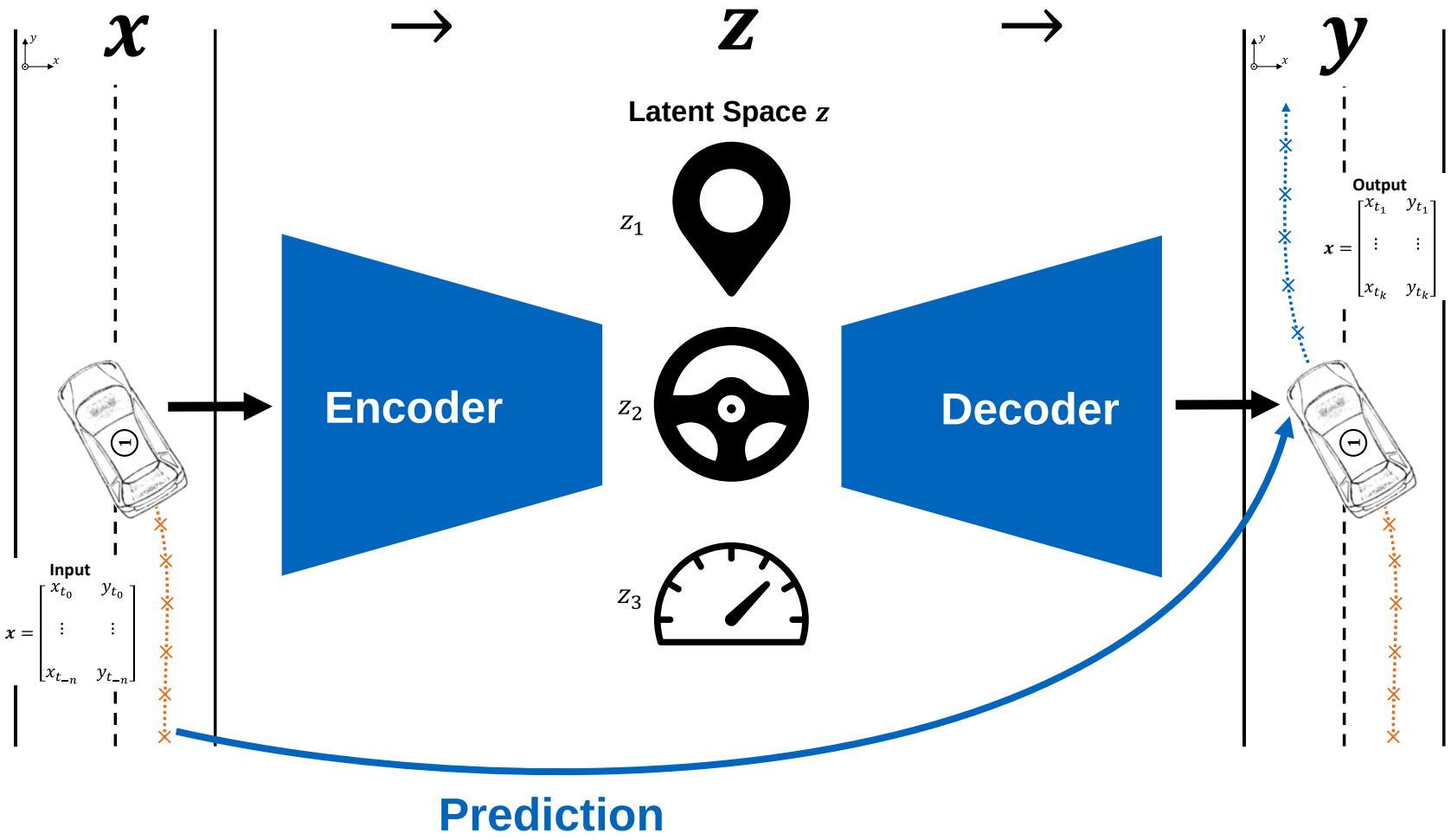
$\boldsymbol{G(z)}$: Decode-Function

$\mathcal{L}(\boldsymbol{x}, \hat{\boldsymbol{x}})$: Loss-Function

Deep Learning Patterns – Encoder-Decoder



Deep Learning Patterns – Encoder-Decoder



Deep Learning Patterns – Encoder-Decoder

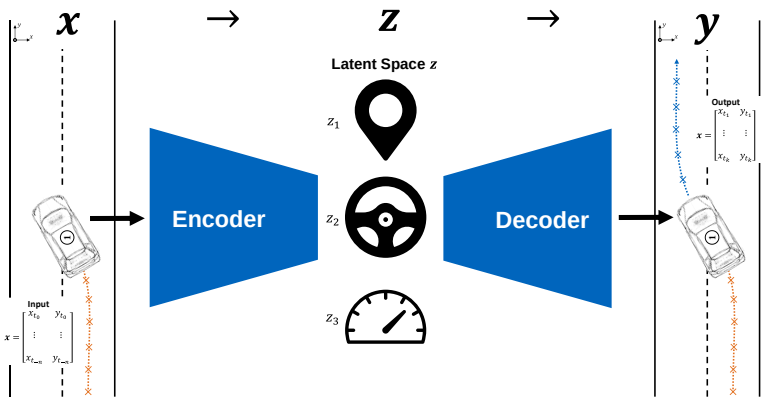
- Loss-Function (e. g. $\|.\|_2^2$-Norm):

$$\mathcal{L}_2(\boldsymbol{y_P}, \boldsymbol{y_{GT}}) = \|\boldsymbol{y_P} - \boldsymbol{y_{GT}}\|_2^2$$

with

$$\boldsymbol{y_P} = \begin{bmatrix} x_{t_1} & y_{t_1} \\ \vdots & \vdots \\ x_{t_k} & y_{t_k} \end{bmatrix}_{\mathrm{P}}; \quad \boldsymbol{y_{GT}} = \begin{bmatrix} x_{t_1} & y_{t_1} \\ \vdots & \vdots \\ x_{t_k} & y_{t_k} \end{bmatrix}_{\mathrm{GT}}$$

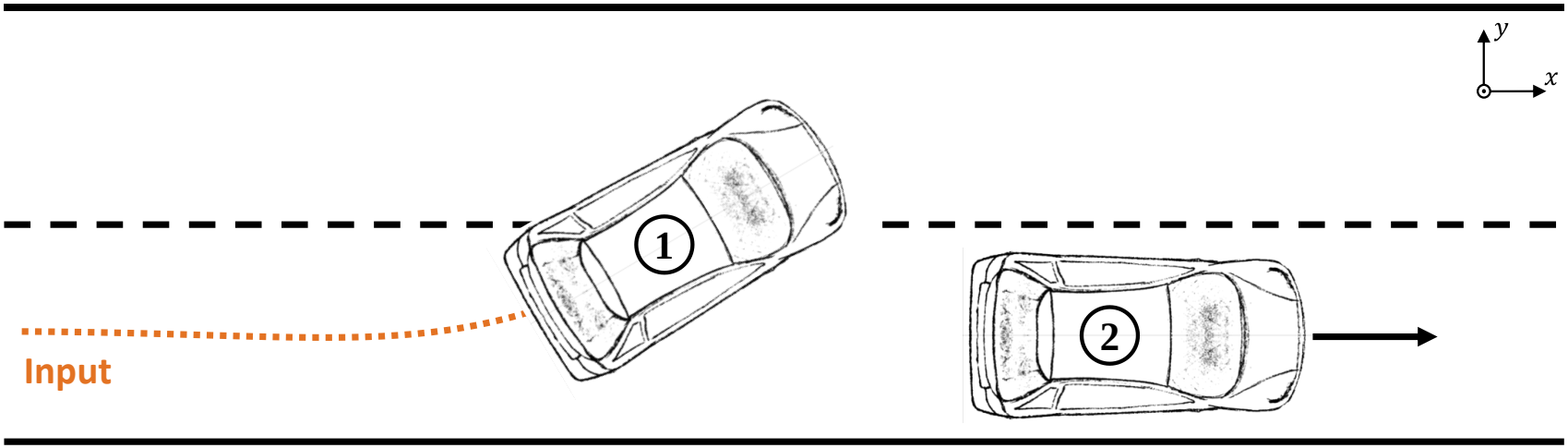
$\cdot_p$: prediction
$\cdot_{\mathrm{GT}}$: ground truth



$\boldsymbol{x}$: Input
$\boldsymbol{y}$: Prediction

$\boldsymbol{F(x)}$: Encode-Function
$\boldsymbol{z}$: Latent Space
$\boldsymbol{G(z)}$: Decode-Function
$\mathcal{L}(\ )$: Loss-Function

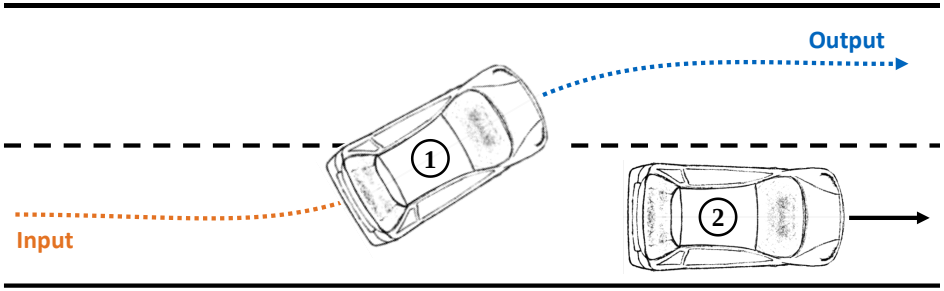
Deep Learning Patterns – Self-Supervision



Deep Learning Patterns – Self-Supervision

Definition

A form of unsupervised learning where the data provides the supervision



Idea

Withhold some part of the data, and task the network with predicting it

For Motion Prediction

- Track trajectories of vehicles
- Withhold a part of it, which the model has to learn

"Self-supervised learning is the key to human-level intelligence."

- Yann LeCun
Founding father of convolutional neural networks, 2018 Turing Award Winner

Additional Slides

Deep Learning Patterns

In the latest prediction challenges on public data sets [37, 38] in the field of motion prediction for autonomous driving, the best results were achieved with autoencoder architectures consisting of deep neural networks [39–42]. In contrast to encrypted data processing in which the decoder just rebuilds the original uncompressed input, for motion prediction the decoder outputs future trajectories while the encoder gets recent tracks of the objects. Since input and output have the same format, no labeled data is needed. Instead, a tracked trajectory is split into an input with specified observation length and the ground truth as the succeeding part over the prediction horizon. Thus, using autoencoders can be interpreted as a self-supervised learning method.

The type of network within an encoder-decoder architecture depends on the proposed field of application. In case of motion prediction, mainly two aspects are relevant. On the one hand, the temporal evolution of an object’s motion is essential information to encode sequential data Recurrent Neural Networks (RNN) are the dedicated network architecture. The Long-Short Term Memory (LSTM) is the primarily used type. Alternatively the Gated Recurrent Unit (GRU) [43] can be used as cell within a RNN.

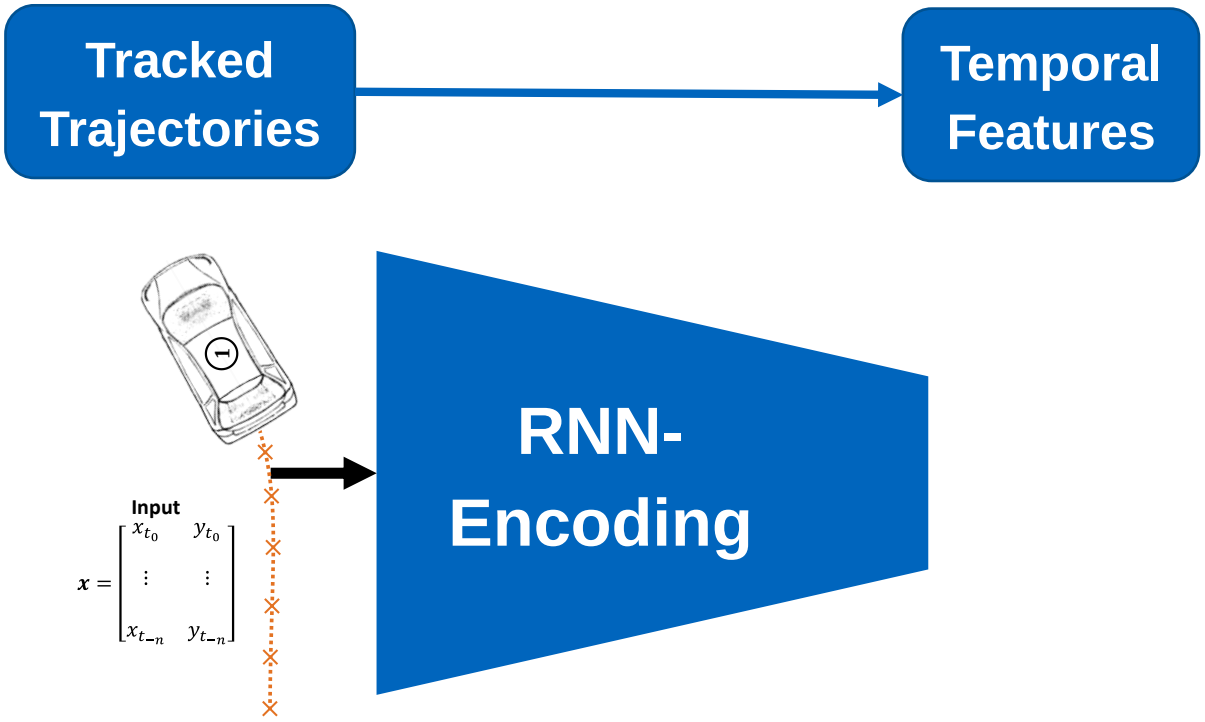
Additional Slides

State of the Art encoder-decoder models for motion prediction from Literature:

		Encoder-Decoder Architectures ***Latent Space Representation***	
		Autoencoder *Discrete*	VAE, cVAE, WAE *Distributional*
Network ***Encoding***	RNN *Temporal*	• Grid Map [44, 45] • GAN [46–48] • Spectral clustering [49] • Self-attention layer [39]	• Kinematic constraint layer [50]
	CNN *Spatial*	• Rasterization into RGB-image [51, 52] • Graph-operational layer [53] • Endpoint determination [54]	
	RNN-CNN, ConvLSTM *Spatiotemporal*	• BEV-Rendering [55, 56] • Social Tensor [40, 57, 58] • Spatial horizon interaction [59] • Grid Map [60–63]	• cGAN [64] • Regression-based adaptation [65] • MANN [66] • Determinantal point process [67]

VAE: Variational Autoencoder
cVAE: Conditional Variational Autoenocer
WAE: Wasserstein Autoencoder

Deep Learning Patterns – Encoder-Decoder Example



RNN Recurrent Neural Network
CNN Convolutional Neural Network

Additional Slides

Recurrent Neural Networks

Recurrent Neural Network(RNN) are a type of Neural Network where the output from previous step are fed as input to the current step. In traditional neural networks, all the inputs and outputs are independent of each other, but in cases like when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words. Thus RNN came into existence, which solved this issue with the help of a Hidden Layer. The main and most important feature of RNN is Hidden state, which remembers some information about a sequence.

RNN have a “memory” which remembers all information about what has been calculated. It uses the same parameters for each input as it performs the same task on all the inputs or hidden layers to produce the output. This reduces the complexity of parameters, unlike other neural networks.

Advantages of Recurrent Neural Network

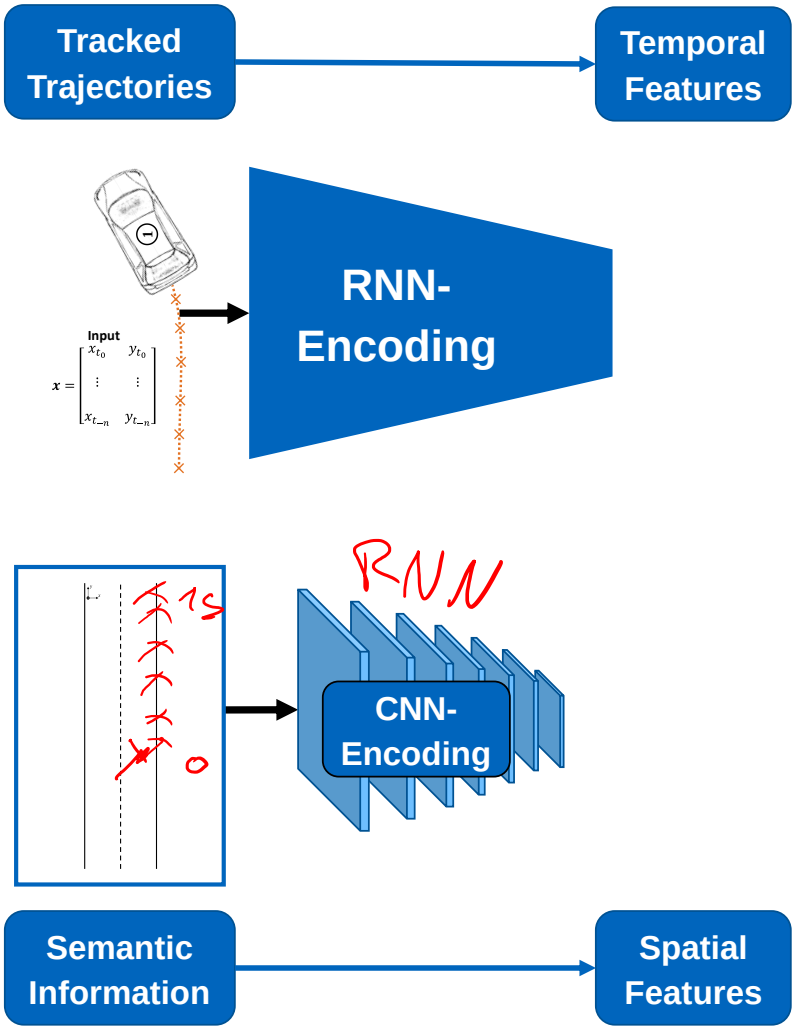
- An RNN remembers each and every information through time. It is useful in time series prediction only because of the feature to remember previous inputs as well. This is called Long Short Term Memory.
- Recurrent neural network are even used with convolutional layers to extend the effective pixel neighborhood.

Disadvantages of Recurrent Neural Network

- Gradient vanishing and exploding problems.
- Training an RNN is a very difficult task.
- It cannot process very long sequences if using tanh or relu as an activation function.

Reference: https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/

Deep Learning Patterns – Encoder-Decoder Example



RNN Recurrent Neural Network
CNN Convolutional Neural Network

Additional Slides

Convolutional Neural Networks

CNNs are a variant of feed-forward neural networks with a special architecture. The architecture of CNNs usually contains a convolution followed by a pooling operation. Every neuron in a convolutional layer is connected to some region in the input, which is called a local receptive field. Unlike other types of feed-forward neural networks, all weights are shared based on the position within a receptive field. The shared weights are also called filters. The shared weights are also called filters. The convolutions operation can be formalized as follows:

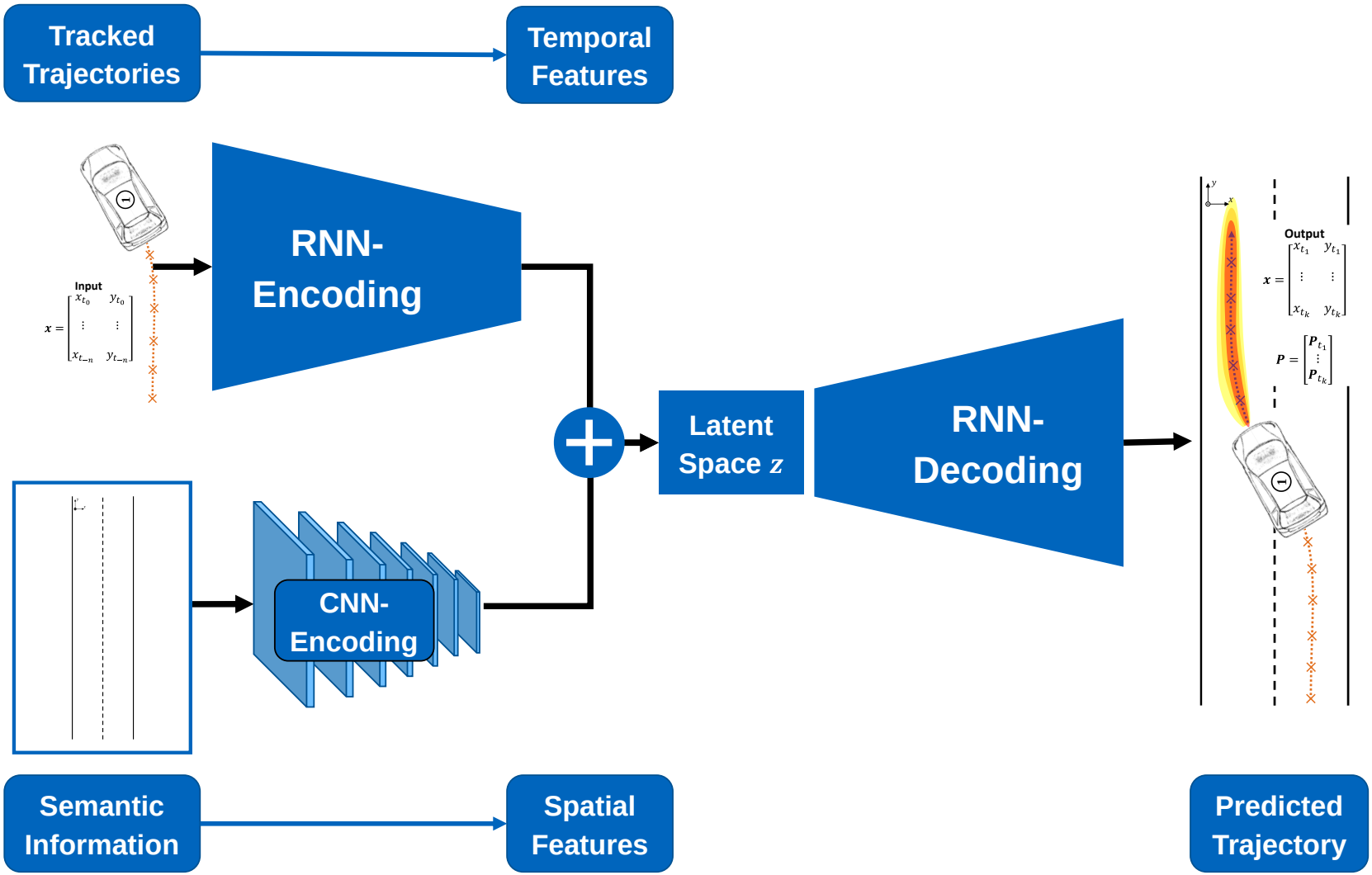
$$(f * g)(z) = \sum_x \sum_y f(x, y) \cdot g(z - x, z - y)$$

where $f(x, y)$ is the input tensor at position (x, y) and $g(z - x, z - y)$ is a trainable filter. Further, a pooling layer is usually used to generate translation invariant features by computing statistics of the convolution activations from different positions along specific windows.

Reference:

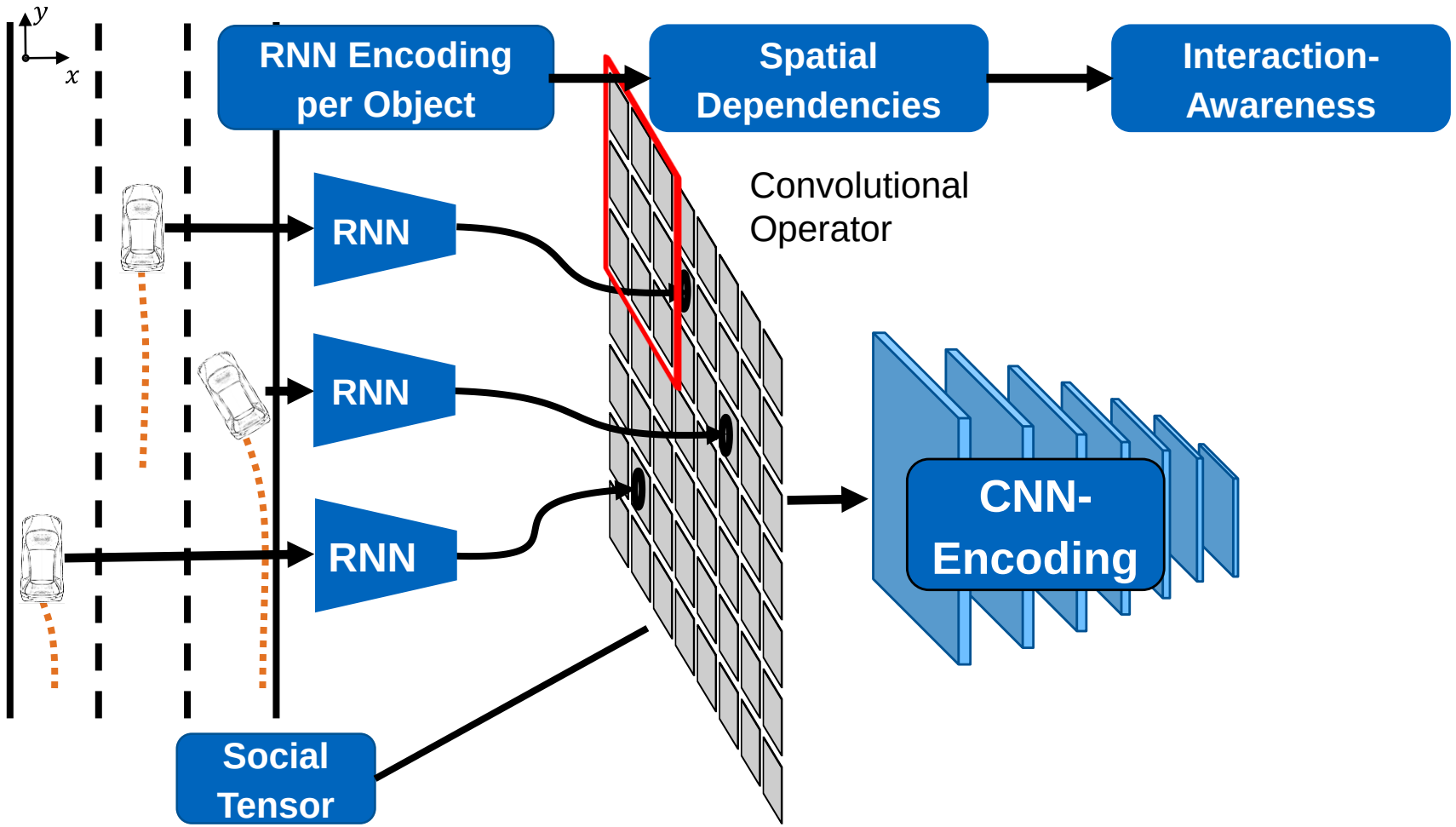
Suryani, Doetsch – On the Benefits of Convolutional Neural Network Combinations in Offline Handwriting Recognition
https://www-i6.informatik.rwth-aachen.de/publications/download/1021/SuryaniDewiDoetschPatrickNeyHermann--OntheBenefitsofConvolutionalNeuralNetworkCombinationsinOfflineHwritingRecognition--2016.pdf

Deep Learning Patterns – Encoder-Decoder Example



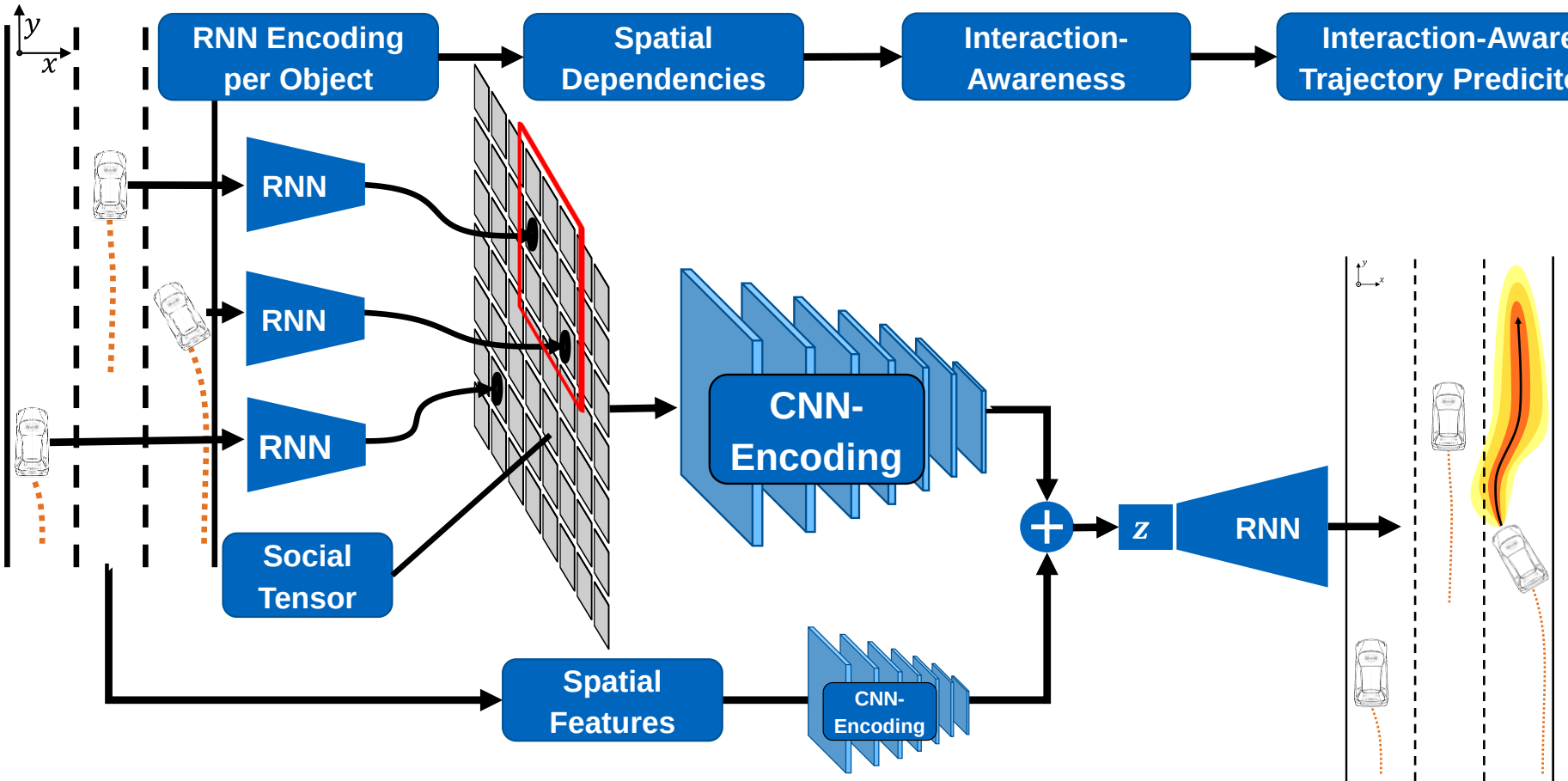
RNN Recurrent Neural Network
CNN Convolutional Neural Network

Deep Learning Patterns – Encoder-Decoder Example



RNN Recurrent Neural Network
CNN Convolutional Neural Network

Deep Learning Patterns – Encoder-Decoder Example

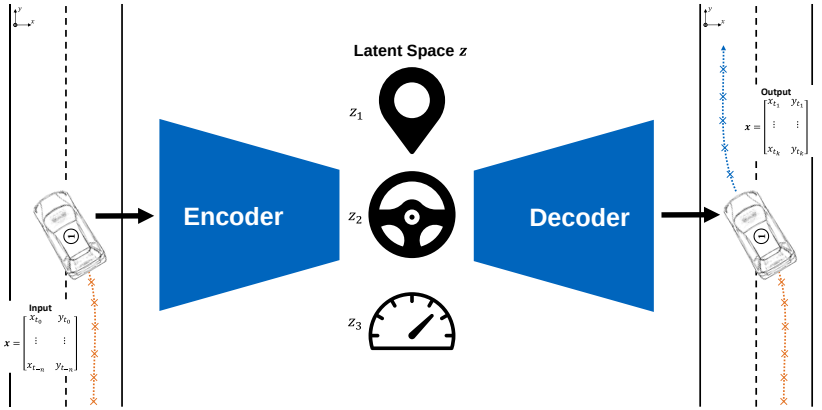


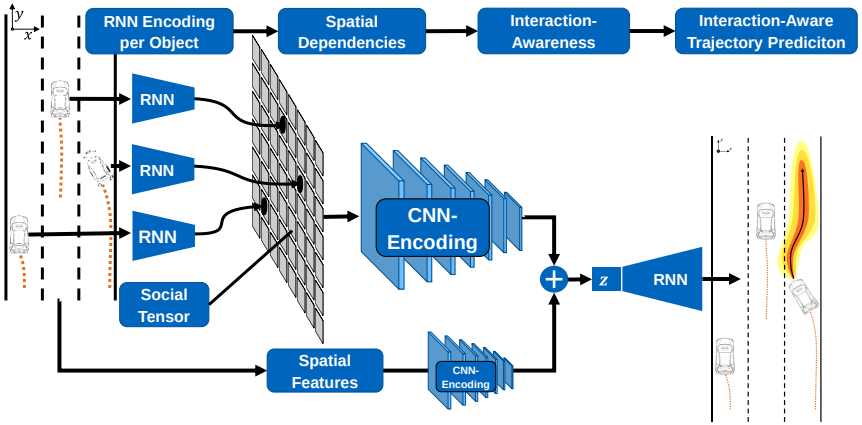
RNN Recurrent Neural Network
CNN Convolutional Neural Network

Deep Learning Patterns

+ Modular, individual network design
+ Comprehensive prediction models: consideration of spatial information and interaction possible
+ Trajectory prediction and Uncertainty quantification
+ Self-supervised Learning

- No explainability for safety verification
- Robustness to extrapolation to unknown data not defined





Additional Slides

Autoencoder Example

Temporal Features are extracted with a LSTM-Encoder Spatial Features are extracted with a CNN-Network. Input can be map data including drivable areas.

The encoded variables are concatenated and fed into an LSTM-Decoder, which maps the latent space back to the physical space. Hence, the predicted trajectory under consideration of semantical und dynamic information can be determined.

Until now, only a single car in the dynamic environment is modeled. To enable interaction-awareness, i.e. to consider surrounding objects, further features have to be added to the model. One approach is to encode each object‘s motion and to render them into a so called Social Tensor. This Tensor can be interpreted as a grid representation of the environment. Hence, the encoded objects are allocated at the corresponding position in the Tensor. To extract spatial dependencies from the tensor, CNNs can be used. The decoding step is similar to the non-interactive architecture, but the output differs in terms of spatial dependencies in the object‘s trajectories.

Alternative Approaches to consider interactions awareness are Graph Neural Networks.

Prediction
Prof. Dr. Markus Lienkamp

Phillip Karle, M. Sc.

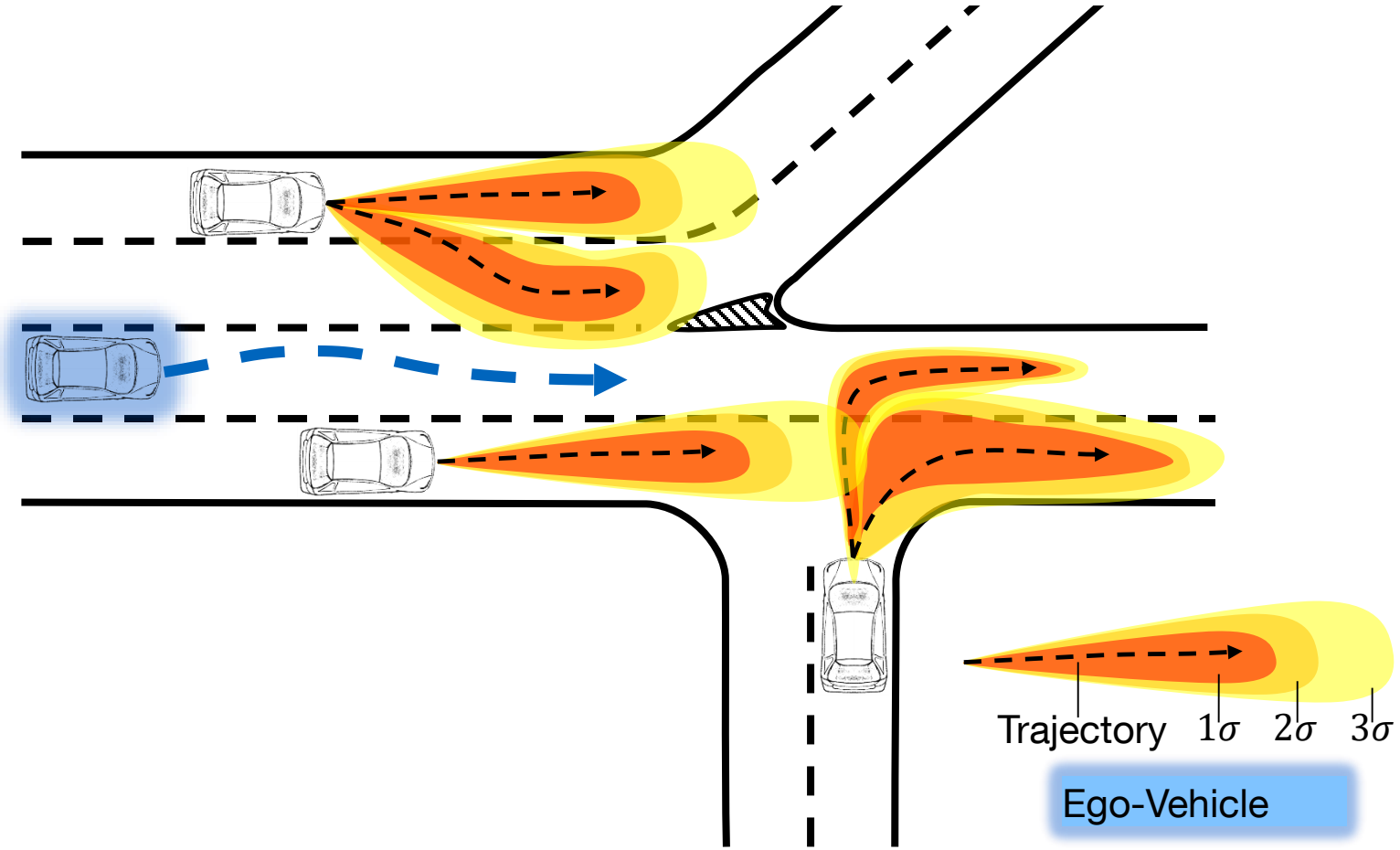
Agenda

1. Foundations
2. Physics-Based Prediction
3. Pattern-Based Prediction
4. **Planning-Based Prediction**
5. Summary and Outlook

Planning-based Prediction

Class	Concept	Methods
Physics-based Prediction	Sense – Predict	• State Estimation • Reachable Sets
Pattern-based Prediction	Sense – Learn – Predict	• Clustering & Classification • Deep Learning Patterns
Planning-based Prediction	Sense – Reason – Predict	• Learning from Demonstration • Planning under Uncertainty

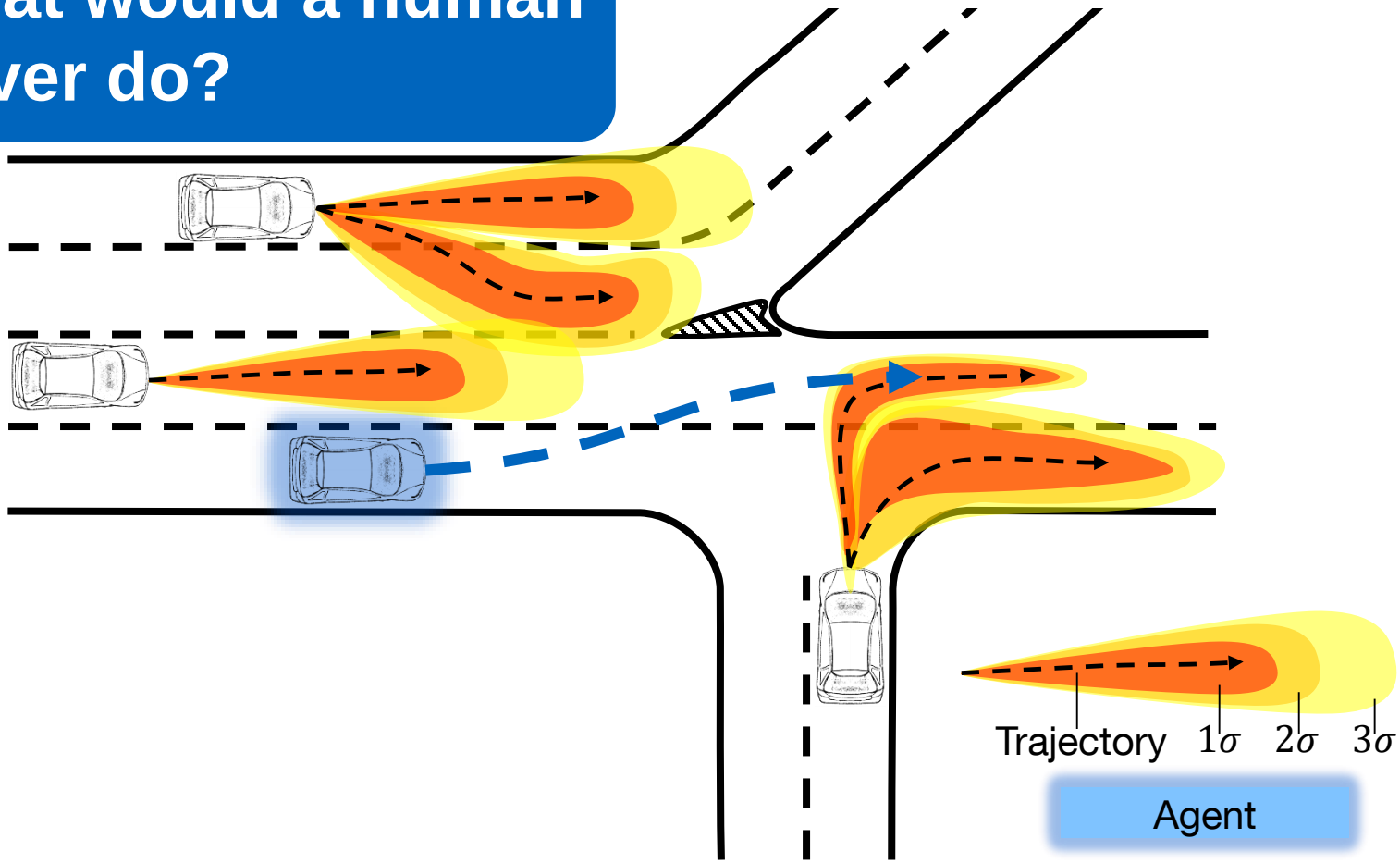
The Duality of Prediction and Ego-Planning



The Duality of Prediction and Ego-Planning

perfect

What would a ~~human~~ driver do?



Additional Slides

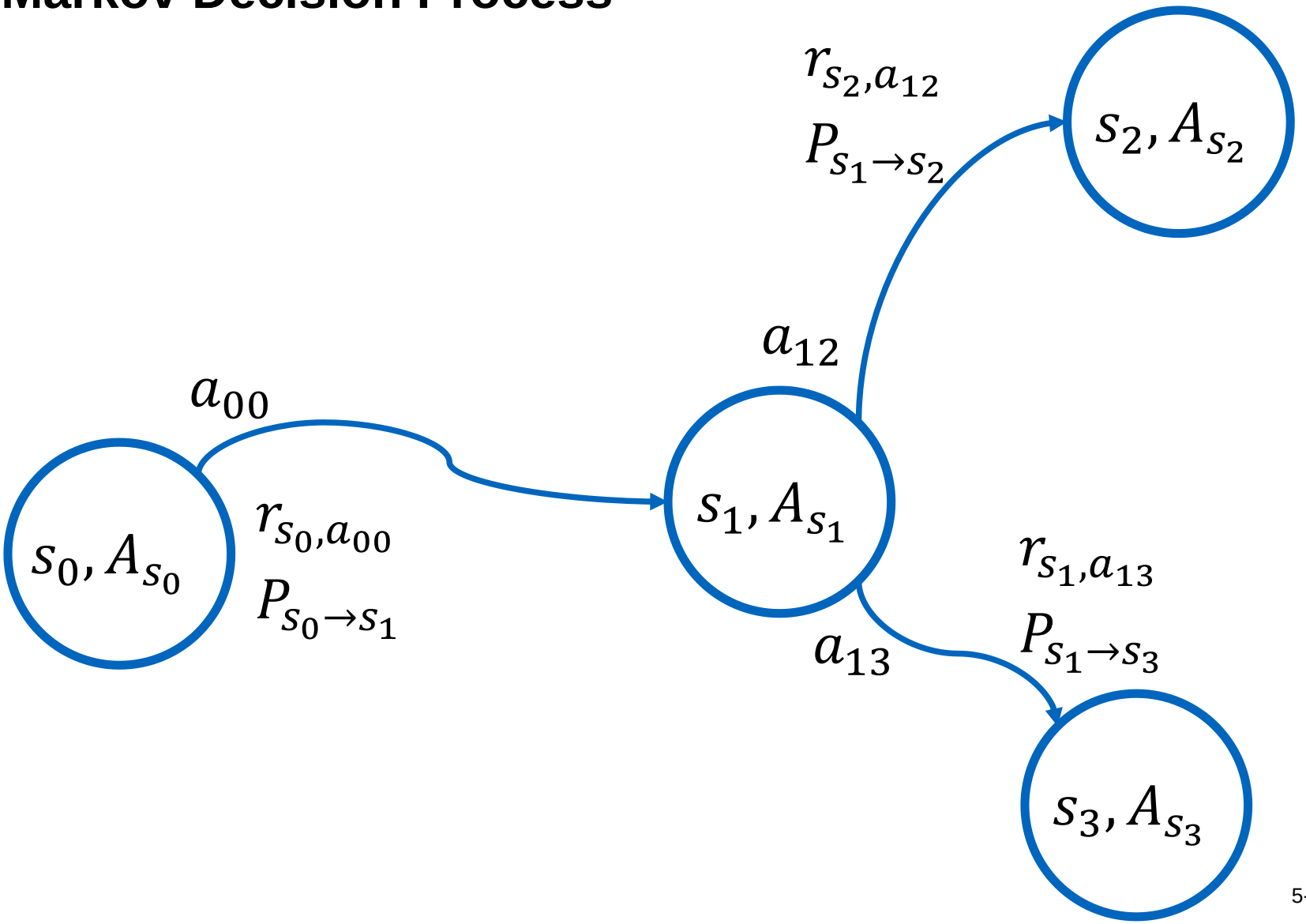
Planning-based Prediction

The idea of incorporating the prediction directly in motion planning is motivated by the point of view that prediction and planning can be considered a dual problem, meaning that predicting another vehicle with all contextual information is the same as predictively planning the ego trajectory ahead, given a specific scenario. Hence, planning under uncertainty does not predict other objects' trajectories explicitly but rather considers them as an uncertain environment implicitly in the ego-motion planning.

Markov Models

	The current state is known	The current state is estimated
Actions are not controllable by the agent in every state	Markov Process (MP)	Hidden Markov Model (HMM)
Actions are controllable by the agent in every state	Markov Decision Process (MDP)	Partially observable Markov Decision Process (POMDP)

Markov Decision Process



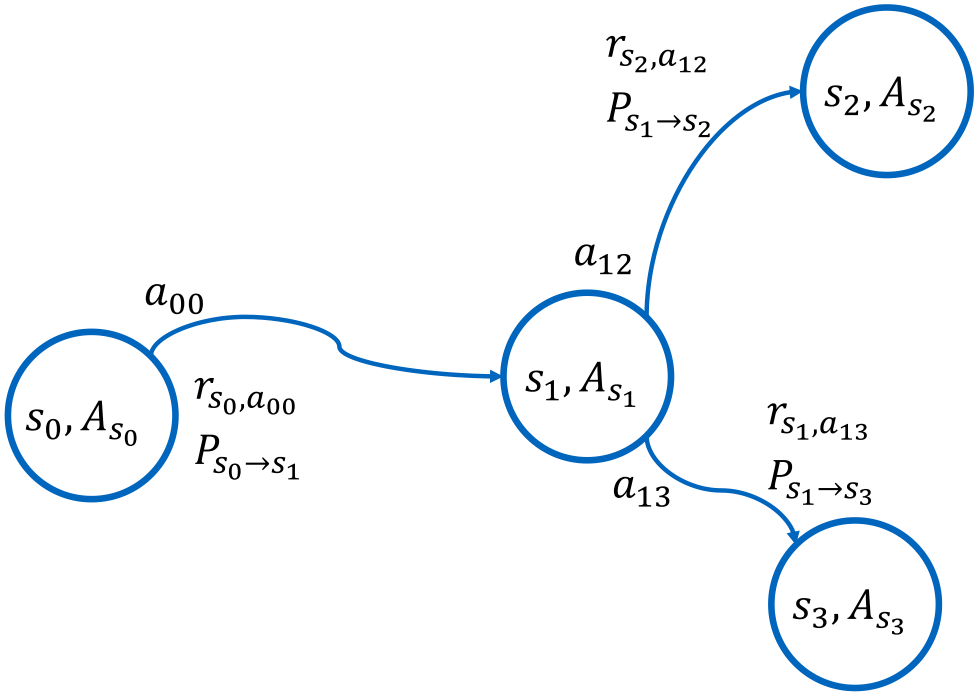
Markov Decision Process

Idea

- Markov Property
- Agent controls the action in a state

Model Parameters

- Tuple: (S, A, R, P)
- States $S = \{s_0, \dots, s_n\}$
- Set of Actions per State A with $A_{s_i} = \{a_{s,0}, \dots, a_{s,n}\}$
- Rewards $R(s, a, s')$
- Transition probability $P(s'|s, a)$



Learning from demonstration

Idea

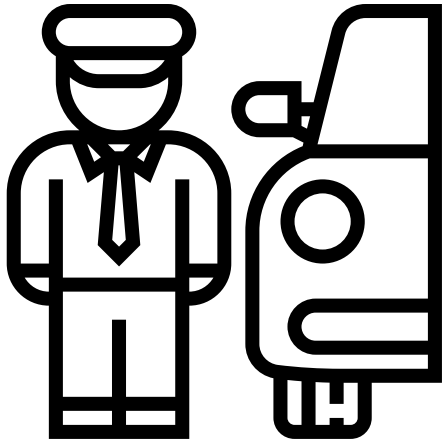
- An expert demonstrates the optimal behavior
- Model: Markov Decision Process

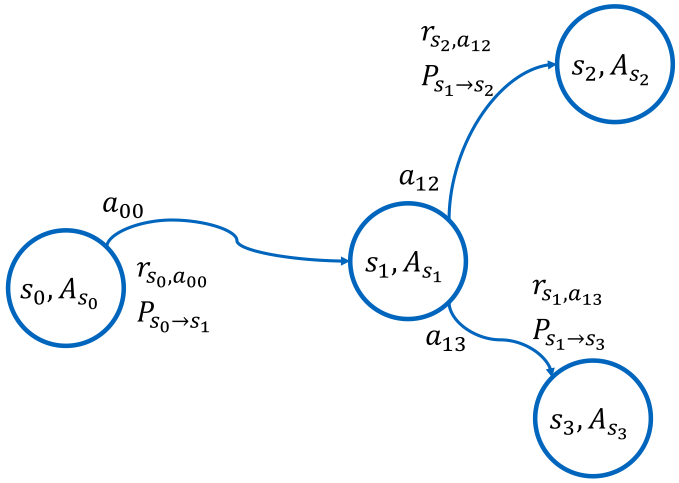
Input

- Collections of experts‘ demonstrations in various situations, which are assumed optimal, i.e. belonging to the optimal policy π^*

Definition

- A policy is defined as π: $S \mapsto A$





Learning from demonstration

Algorithm

- Given: State-Actions-Tuples

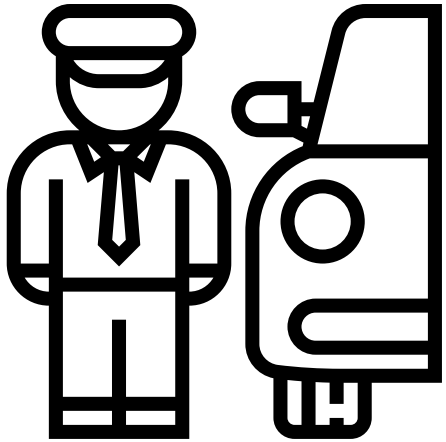
$$(s_i, a_i), a_i \sim \pi^*$$

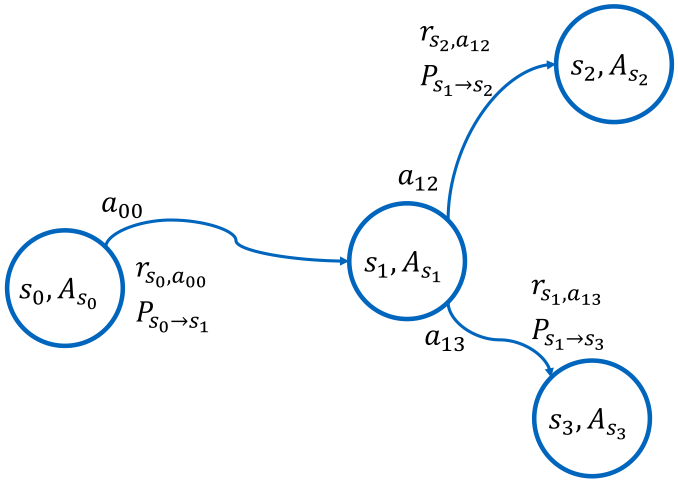
- Unknown: Reward function

$$R(s_i, a_i, s_{i+1})$$

Goal

Develop an algorithm, which is capable to imitate demonstrated driving behavior





Additional Slides

Learning from demonstration

The idea of learning from demonstration is to derive the motion prediction by estimating the underlying cost function or by directly imitating an observed behavior. In both cases, it is assumed that the observed agent always tries to reach its goal with the optimal policy regarding a specific cost function. Hand-crafted cost functions and resulting policies require comprehensive expert knowledge. The task of representing a broad range of traffic scenarios is complex. Furthermore, features, which a human would describe as essential to define the costs and explain certain behavior, might not be observable. Hence, the aim is to reproduce driver behavior from observed input data. To achieve this, it is assumed that the observed behavior originates from experts and contains enough features to imitate it. In addition, the data must comprise sufficient scenarios to learn a general and robust function. The two major approaches for Learning from Demonstration are Inverse Reinforcement Learning (IRL) and Imitational Learning (IL). The Markov Decision Process (MDP) is the framework for IRL and IL. It is the active form of a Markov Process (see Table II). For further information about the MDP, the reader is referred to [68].

Behavior Cloning

Idea

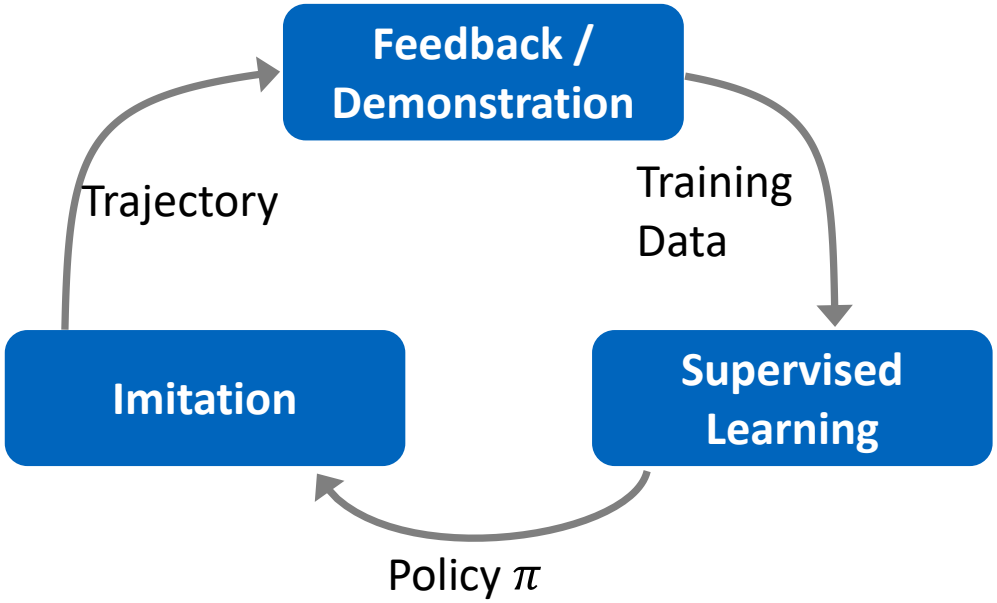
- Imitate the observed behavior
- Goal: Recover the expert‘s policy directly π^*

Output

Imitation of expert’s driving behavior

Algorithms

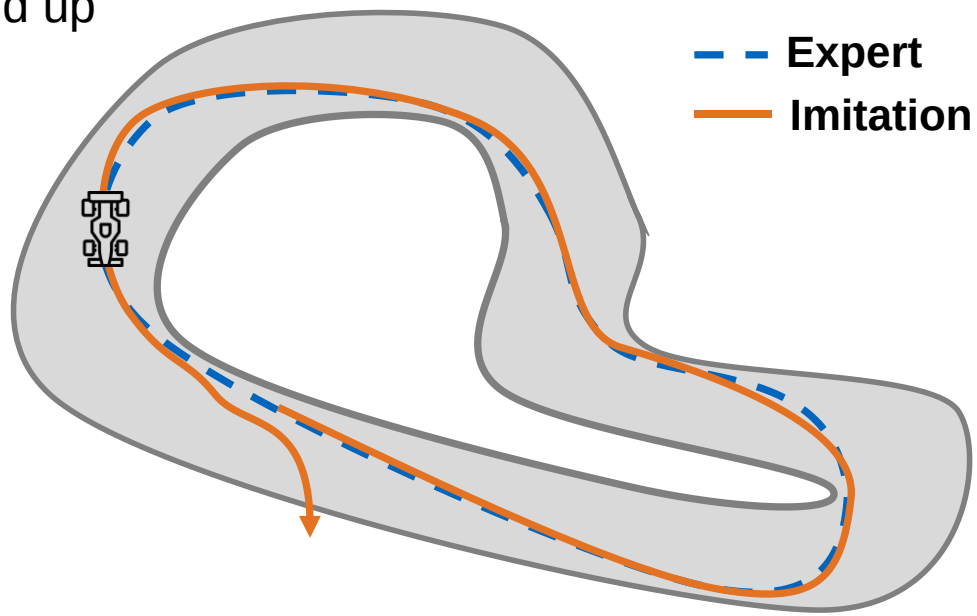
Supervised Learning, e.g. Support Vector Machine, Deep Learning



Behavior Cloning

+ Direct policy output

- No extrapolation of the imitated policy to new task / environments
- Markovian assumption may add up errors from previous states
- Curse of dimensions



Additional Slides

Why Imitational Learning fails:

- State-Actions pairs have to cover all (ideal) situations for robust supervised learning
- Memory lessness adds up errors from previous stages
- Mistake by the agent can lead him to unknown states, which the model never trained on
- Results is an undefined behavior

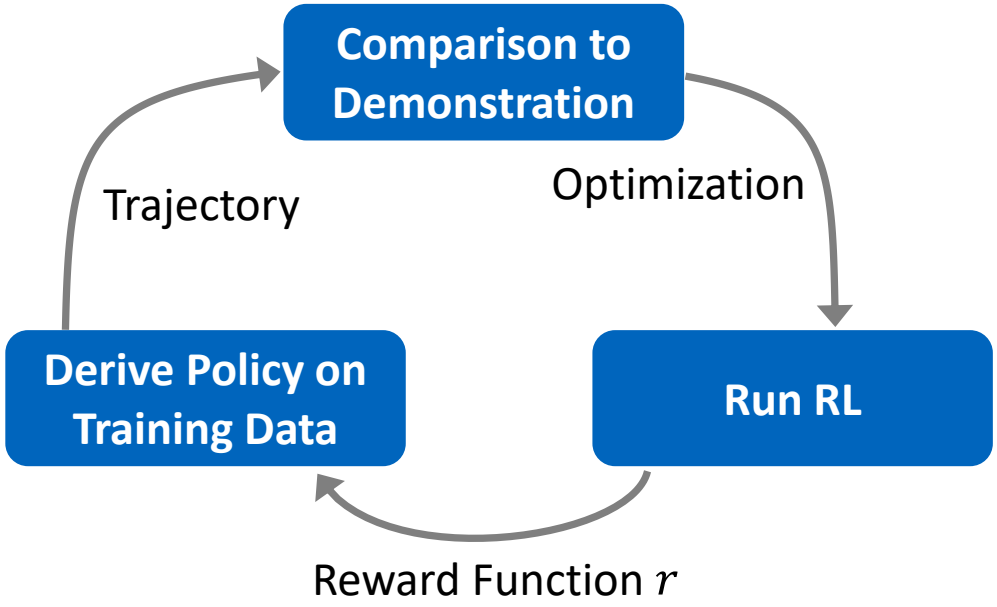
Inverse Reinforcement Learning

Idea

- Recover the expert‘s reward function $R(s, a, s')$
- Goal: Find the best function to get the maximal reward for every observed policy

Output

Reward function which depicts the expert’s driving behavior



RL: Reinforcement Learning

Example: IRL from sampled trajectories

Given

Set of trajectories $\{\xi\}$, which are assumed to be part of the optimal policy π^*

Initialization

- Reward function R from linear combination of L features ϕ_i
- Initialize the feature weights α_i randomly

Given:

$$\{\xi\}_{i=1}^{N}, \xi_i = \{(s_j, a_j)\}_{j=1}^{H}; a_j \sim \pi^*(s_j)$$

Initialization:

$$R(s, a, s') = \alpha_0\phi_0(s) + \cdots + \alpha_L\phi_L(s)$$

Example: IRL from sampled trajectories

Iteration

- Sample a set of policies $\pi_j: S \longmapsto A$ based on the initial reward function $R(\alpha_i)$
- Estimate the value (total reward) $\hat{V}^{\pi^*}(s_0)$ for the optimal policy and value for the sample $\hat{V}^{\pi_j}(s_0)$ from a initial state
- Optimize the reward function by linear programming, i.e. set α_i to maximize the difference between the optimal values and all the other value functions

Iteration – Step k

Sampling: $R_k(\alpha_{k,i}) \rightarrow \pi_{k,j}$

$\{\pi_{k,j}\}_{j=1}^{H}$ with $\pi_{k,j}: S \longmapsto A$

Value Estimation:

$$\hat{V}_i^{\pi}(s) = \sum_{a \epsilon A_s} \pi(s,a)\left(R_{s,a,s'} + \gamma P_{s',s,a} V^{\pi}(s')\right)$$

Maximize:

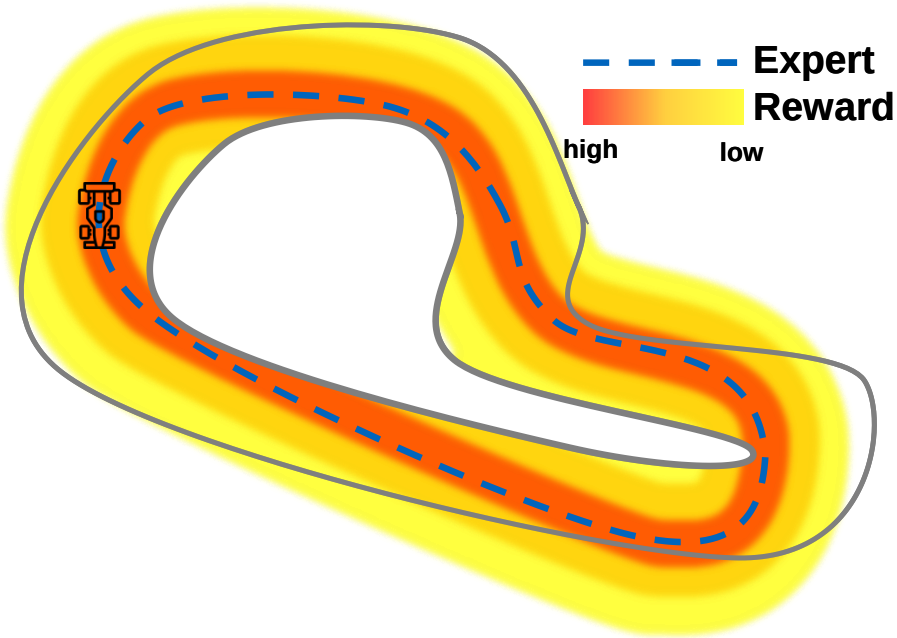
$$\sum_{j=1}^{k} P\left(\hat{V}^{\pi^*}(s_0) - \hat{V}^{\pi_j}(s_0)\right)$$

$$\text{s.t. } |\alpha_{k,i}| < 1$$

Inverse Reinforcement Learning

+ Enhanced Robustness, generalization and transferability

- No direct policy output
- Hard to train in environments with sparse rewards / no direct reward function at all
- Policies can be optimal for many reward functions



Additional Slides

What is as reward function?

“the reward function, rather than the policy, is the most succinct, robust, and transferable definition of the task”

Goal of IRL:

→ Cover as much situations as possible (generalization)

→ Create a consistent function (robustness)

→ Choose general features (transferability)

Further approaches to IRL:

- Ziebart et al., 2008 – Maximum Entropy IRL
- GAIL: Reason about intentions, but still directly learn the policy instead of just the cost function

References:

https://medium.com/@SmartLabAI/a-brief-overview-of-imitation-learning-8a8a75c44a9c

Ng, Russel, 2000 – Algorithms for Inverse Reinforcement Learning

https://ai.stanford.edu/~ang/papers/icml00-irl.pdf

https://ipvs.informatik.uni-stuttgart.de/mlr/wp-content/uploads/2017/07/09-InverseRL-Hung.pdf

Prediction
Prof. Dr. Markus Lienkamp

Phillip Karle, M. Sc.

Agenda

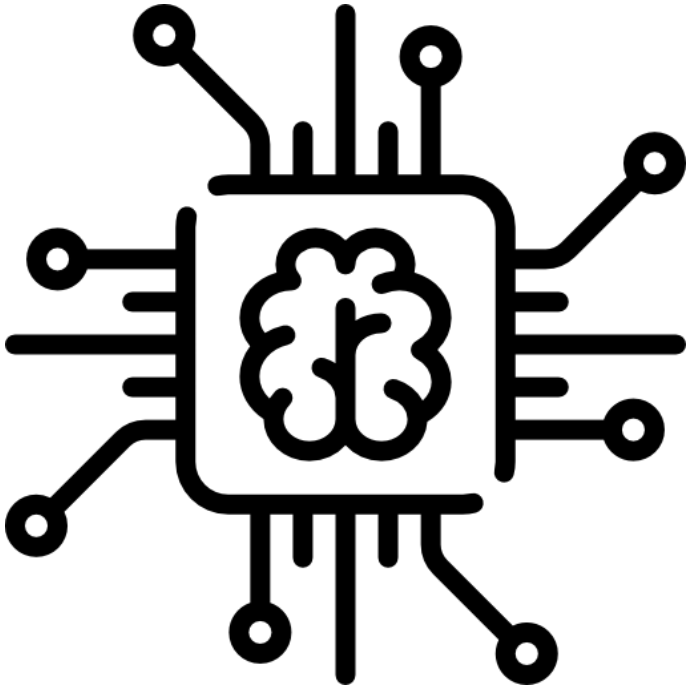
1. Foundations
2. Physics-Based Prediction
3. Pattern-Based Prediction
4. Planning-Based Prediction
5. **Summary and Outlook**

Comparison of Prediction Models

	Kinematic Models	Deep Learning Patterns	Learning from Demonstration
Prediction horizon Time span of valid forecasting	Short-term	Long-term	Long-term
Explainability human understandable relation between feature values and predicted output	High	Low	Low (IL), Medium (IRL)
Holism Consideration of semantic information and interaction between objects	Low	High	High
Complexity Capability to describe complex behavior	Low	High	High
Data dependency Necessity of data parametrize the model	Low	High	High
Adaptivity Robust application to unknown scenarios	High	Medium	High (IRL)

How to anticipate other traffic participants?

Anticipatory Driving through Riding Experience



Motion Prediction through Scenario Understanding

Physics-based Prediction

State Estimation

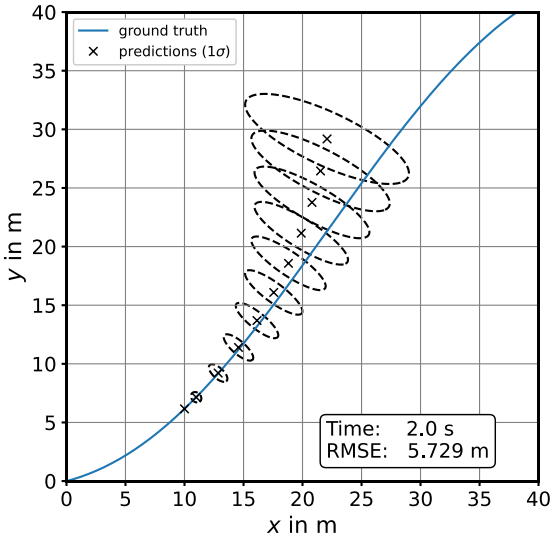
- Kinematic Models
- Bayesian Filter for Probabilistic Trajectory Prediction

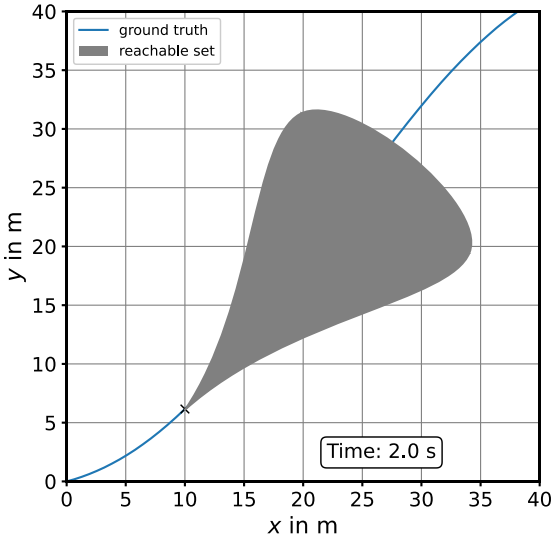
Reachability Analysis

- Coverage of all possible states within the dynamic limits of the object

Robust algorithms

- High accuracy in short-term prediction
- No comprehensive trajectory prediction





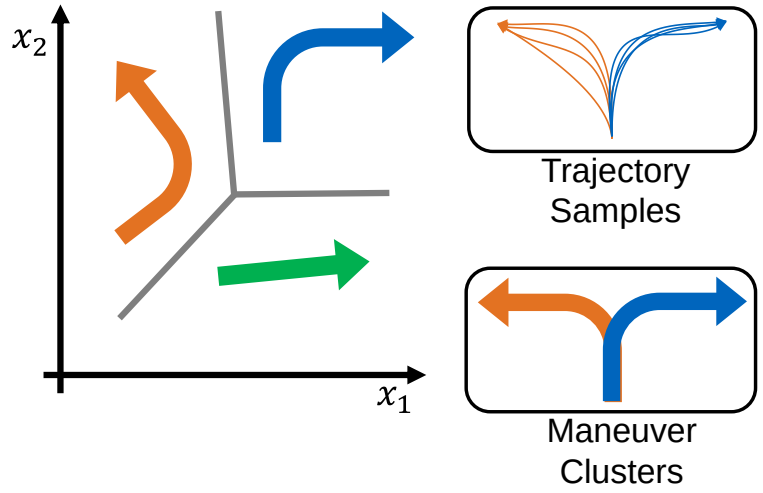
Pattern-based Prediction

Clustering and Classification

- Maneuver Prediction
- Structured Environments

Encoder-Decoder Algorithms

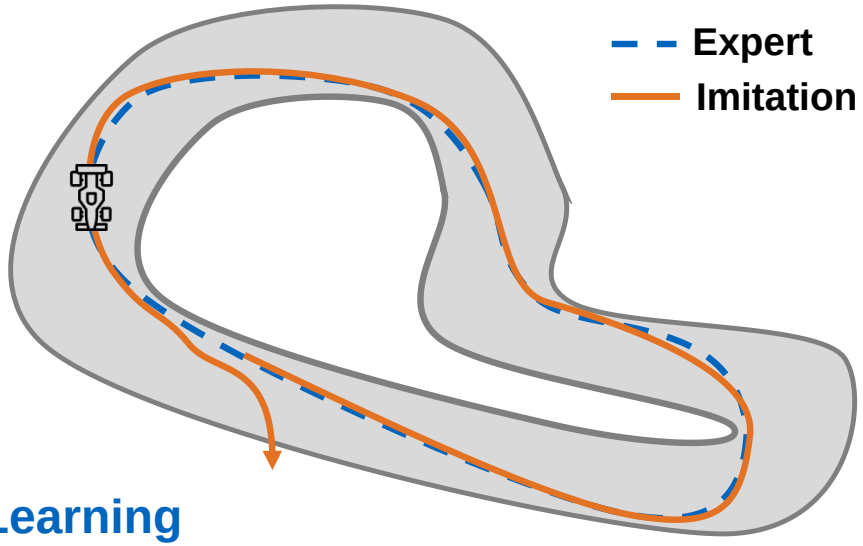
- State of the Art-Algorithms for comprehensive Motion Prediction
- Use of RNN- and CNN-architectures
- Bottleneck Latent Variables
- Lack of Explainability



Planning-based Prediction

Learning from Demonstration

Idea: Observation of Expert Behavior



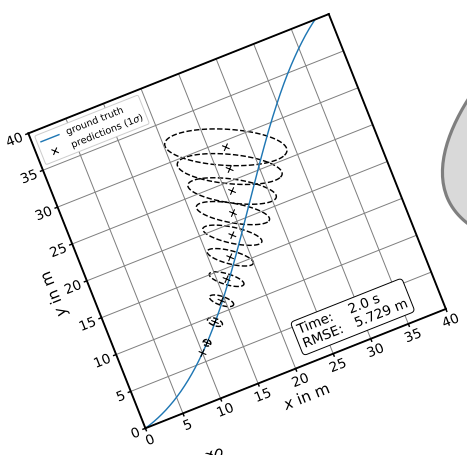
1st Method: Behavior Cloning

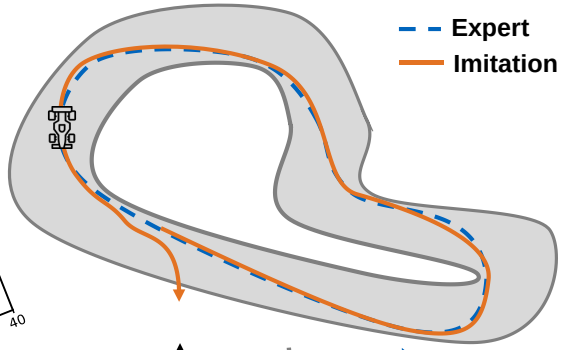
- Direct Policy Imitation
- Not extrapolatable

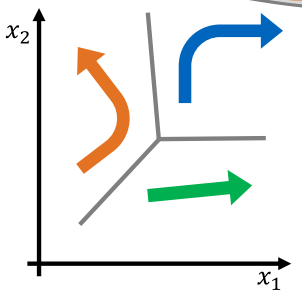
2nd Method: Inverse Reinforcement Learning

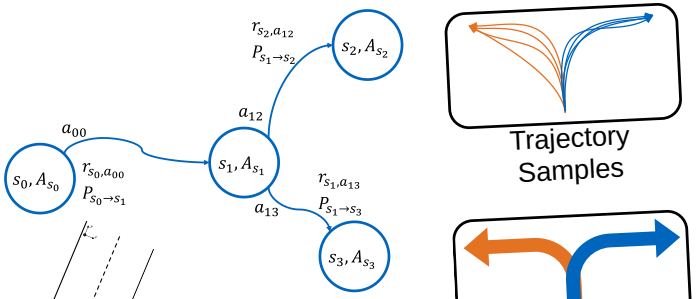
- Reward function learning
- Hardly trainable
- Ambiguous reward function

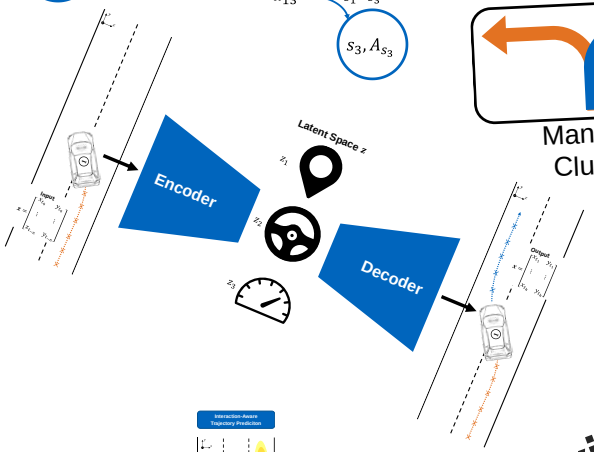
Outlook – Motion Prediction is an open research topic

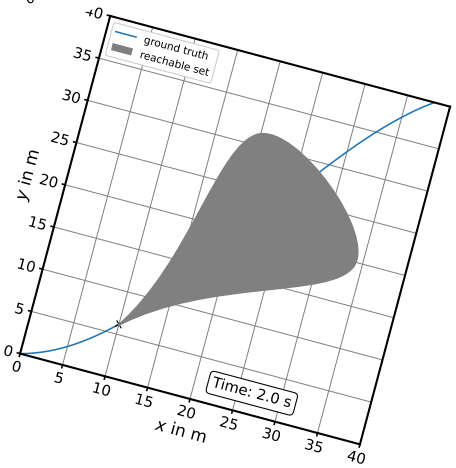


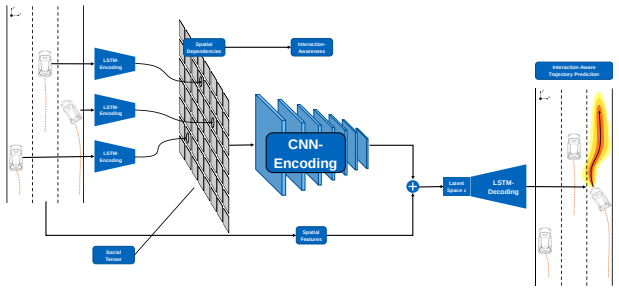












Self-Supervision

Outlook

Dynamic Planning

Prediction is essential for dynamic planning in complex environment

Scenario Understanding

Comprehensive prediction aims to encode the human driving behavior

Optimal Driving Behavior

Prediction of surrounding objects and planning the ego-behavior are dual problems and only differ from the view of perspective

Learning from Demonstration

Prediction could be the enabler for new planning algorithms

→ **Chapter 11: End-to-End**

References

[1] T. Lin, E. Tseng, and F. Borrelli, “Modeling driver behavior during complex maneuvers,” in American Control Conference, 2013, pp. 6448–6453.

[2] J. H. Yoo and R. Langari, “Stackelberg Game Based Model of Highway Driving,” in Dynamic Systems and Control Conference, pp. 499–508.

[3] S. Ammoun and F. Nashashibi, “Real time trajectory prediction for collision risk estimation between vehicles,” in IEEE 5th International Conference on Intelligent Computer Communication and Processing, 2009, pp. 417–422.

[4] S. Lefèvre, Y. Gao, D. Vasquez, H. E. Tseng, R. Bajcsy, and F. Borrelli, “Lane Keeping Assistance with Learning-Based Driver Model and Model Predictive Control,” in 12th International Symposium on Advanced Vehicle Control, Tokyo, Japan, 2014.

[5] D. Petrich, T. Dang, G. Breuel, and C. Stiller, “Assessing map-based maneuver hypotheses using probabilistic methods and evidence theory,” in 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), 2014, pp. 995–1002.

[6] A. Carvalho, S. Lefévre, G. Schildbach, J. Kong, and F. Borrelli, “Automated driving: The role of forecasts and uncertainty—A control perspective,” European Journal of Control, vol. 24, pp. 14–32, 2015, doi: 10.1016/j.ejcon.2015.04.007.

[7] S. J. Julier and H. F. Durrant-Whyte, “On the role of process models in autonomous land vehicle navigation systems,” IEEE Transactions on Robotics and Automation, vol. 19, no. 1, pp. 1–14, 2003, doi: 10.1109/TRA.2002.805661.

[8] J. S. Gill, P. Pisu, V. N. Krovi, and M. J. Schmid, “Behavior Identification and Prediction for a Probabilistic Risk Framework,” in Dynamic Systems and Control Conference, Park City, Utah, USA, 2019.

[9] R. Pepy, A. Lambert, and H. Mounier, “Reducing Navigation Errors by Planning with Realistic Vehicle Model,” in IEEE Intelligent Vehicles Symposium, 2006, pp. 300–307.

[10] R. E. Kalman, “A New Approach to Linear Filtering and Prediction Problems,” Transactions of the ASME-Journal of Basic Engineering, vol. 82, Series D, pp. 35–45, 1960.

[11] N. Deo, A. Rangesh, and M. M. Trivedi, “How Would Surround Vehicles Move? A Unified Framework for Maneuver Classification and Motion Prediction,” IEEE Transactions on Intelligent Vehicles, vol. 3, no. 2, pp. 129–140, 2018, doi: 10.1109/TIV.2018.2804159.

[12] C. Barrios and Y. Motai, “Improving Estimation of Vehicle's Trajectory Using the Latest Global Positioning System With Kalman Filtering,” IEEE Transactions on Instrumentation and Measurement, vol. 60, no. 12, pp. 3747–3755, 2011, doi: 10.1109/TIM.2011.2147670.

[13] A. H. Jazwinski, Stochastic processes and filtering theory, 12th ed. San Diego: Acad. Press, 1997.

[14] E. A. Wan and R. van der Merwe, “The Unscented Kalman filter for Nonlinear Estimation,” in Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium, 2000, pp. 153–158.

[15] S. J. Julier and J. K. Uhlmann, “Unscented filtering and nonlinear estimation,” Proceedings of the IEEE, vol. 92, no. 3, pp. 401–422, 2004.

[16] G. Xie, H. Gao, L. Qian, B. Huang, K. Li, and J. Wang, “Vehicle Trajectory Prediction by Integrating Physics- and Maneuver-Based Approaches Using Interactive Multiple Models,” IEEE Transactions on Industrial Electronics, vol. 65, no. 7, pp. 5999–6008, 2018, doi: 10.1109/TIE.2017.2782236.

[17] C. Yang, W. Shi, and W. Chen, “Comparison of Unscented and Extended Kalman Filters with Application in Vehicle Navigation,” Journal of Navigation, vol. 70, no. 2, pp. 411–431, 2017, doi: 10.1017/S0373463316000655.

[18] J. S. Liu and R. Chen, “Sequential Monte Carlo Methods for Dynamic Systems,” Journal of the American Statistical Association, vol. 93, no. 443, pp. 1032–1044, 1998, doi: 10.1080/01621459.1998.10473765.

References

[19] S. Hoermann, D. Stumper, and K. Dietmayer, “Probabilistic long-term prediction for autonomous vehicles,” in IEEE Intelligent Vehicles Symposium (IV), 2017, pp. 237–243.

[20] M. Schreier, V. Willert, and J. Adamy, “An Integrated Approach to Maneuver-Based Trajectory Prediction and Criticality Assessment in Arbitrary Road Environments,” IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 10, pp. 2751–2766, 2016, doi: 10.1109/TITS.2016.2522507.

[21] B. Kim, K. Park, and K. Yi, “Probabilistic Threat Assessment with Environment Description and Rule-based Multi-Traffic Prediction for Integrated Risk Management System,” IEEE Intelligent Transportation Systems Magazine, vol. 9, no. 3, pp. 8–22, 2017, doi: 10.1109/MITS.2017.2709807.

[22] M. Althoff, “Reachability Analysis and its Application to the Safety Assessment of Autonomous Cars,” Dissertation, Technische Universität München, München, 2010.

[23] M. Althoff, C. Le Guernic, and B. H. Krogh, “Reachable set computation for uncertain time-varying linear systems,” in Proceedings of the 14th international conference on Hybrid systems: computation and control, 2011, pp. 93–102.

[24] M. Althoff, O. Stursberg, and M. Buss, “Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes,” Nonlinear analysis: hybrid systems, vol. 4, no. 2, pp. 233–249, 2010.

[25] M. Althoff and J. M. Dolan, “Online Verification of Automated Road Vehicles Using Reachability Analysis,” IEEE Transactions on Robotics, vol. 30, no. 4, pp. 903–918, 2014, doi: 10.1109/TRO.2014.2312453.

[26] S. M. LaValle, Planning algorithms: Cambridge university press, 2006.

[27] M. E. Celebi, Ed., Partitional Clustering Algorithms. Cham: Springer, 2015.

[28] J. Bian, D. Tian, Y. Tang, and D. Tao, “A survey on trajectory clustering analysis,” arXiv preprint arXiv:1802.06971, 2020.

[29] C. Cortes and V. Vapnik, “Support-vector networks,” Machine Learning, vol. 20, no. 3, pp. 273–297, 1995, doi: 10.1007/BF00994018.

[30] P. Kumar, M. Perrollaz, S. Lefèvre, and C. Laugier, “Learning-based approach for online lane change intention prediction,” in IEEE Intelligent Vehicles Symposium (IV), 2013, pp. 797–802.

[31] H. M. Mandalia and M. D. D. Salvucci, “Using Support Vector Machines for Lane-Change Detection,” Proceedings of the Human Factors and Ergonomics Society Annual Meeting, vol. 49, no. 22, pp. 1965–1969, 2005, doi: 10.1177/154193120504902217.

[32] H. Woo et al., “Lane-Change Detection Based on Vehicle-Trajectory Prediction,” IEEE Robotics and Automation Letters, vol. 2, no. 2, pp. 1109–1116, 2017, doi: 10.1109/LRA.2017.2660543.

[33] G. S. Aoude, B. D. Luders, K. K. H. Lee, D. S. Levine, and J. P. How, “Threat assessment design for driver assistance system at intersections,” in 13th International IEEE Conference on Intelligent Transportation Systems, 2010, pp. 1855–1862.

[34] E. Ward and J. Folkesson, “Multi-classification of Driver Intentions in Yielding Scenarios,” in IEEE 18th International Conference on Intelligent Transportation Systems, 2015, pp. 678–685.

[35] G. S. Aoude, V. R. Desaraju, L. H. Stephens, and J. P. How, “Driver Behavior Classification at Intersections and Validation on Large Naturalistic Data Set,” IEEE Transactions on Intelligent Transportation Systems, vol. 13, no. 2, pp. 724–736, 2012, doi: 10.1109/TITS.2011.2179537.

[36] Z. Ghahramani, “An Introduction to Hidden Markov Models and Bayesian Networks,” in Hidden Markov Models: Applications in Computer Vision, USA: World Scientific Publishing Co., Inc, 2001, pp. 9–42.

References

[37] H. Caesar et al., “nuScenes: A multimodal dataset for autonomous driving,” Mar. 2019. [Online]. Available: http://arxiv.org/pdf/1903.11027v4

[38] M.-F. Chang et al., “Argoverse: 3D Tracking and Forecasting with Rich Maps,” Nov. 2019. [Online]. Available: http://arxiv.org/pdf/1911.02620v1

[39] J. Mercat, T. Gilles, N. E. Zoghby, G. Sandou, D. Beauvois, and G. P. Gil, “Multi-Head Attention for Joint Multi-Modal Vehicle Motion Forecasting,” in IEEE International Conference on Robotics and Automation, 2020.

[40] K. Messaoud, N. Deo, M. M. Trivedi, and F. Nashashibi, “Trajectory Prediction for Autonomous Driving based on Multi-Head Attention with Joint Agent-Map Representation,” arXiv preprint arXiv:2005.02545, 2020.

[41] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, “Trajectron++: Dynamically-Feasible Trajectory Forecasting With Heterogeneous Data,” Jan. 2020. [Online]. Available: http://arxiv.org/pdf/2001.03093v3

[42] C. Luo, L. Sun, D. Dabiri, and A. Yuille, “Probabilistic Multi-modal Trajectory Prediction with Lane Attention for Autonomous Vehicles,” arXiv preprint arXiv:2007.02574, 2020.

[43] K. Cho et al., “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation,” Jun. 2014. [Online]. Available: https://arxiv.org/pdf/1406.1078

[44] S. H. Park, B. Kim, C. M. Kang, C. C. Chung, and J. W. Choi, “Sequence-to-Sequence Prediction of Vehicle Trajectory via LSTM Encoder-Decoder Architecture,” in IEEE Intelligent Vehicles Symposium (IV), Changshu, 2018, pp. 1672–1678.

[45] B. Kim, C. M. Kang, J. Kim, S. H. Lee, C. C. Chung, and J. W. Choi, “Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network,” in IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), Yokohama, 2017, pp. 399–404.

[46] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, “Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks,” Mar. 2018. [Online]. Available: http://arxiv.org/pdf/1803.10892v1

[47] J. Li, H. Ma, and M. Tomizuka, “Interaction-aware Multi-agent Tracking and Probabilistic Behavior Prediction via Adversarial Learning,” in International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 2019, pp. 6658–6664.

[48] X. Huang et al., “DiversityGAN: Diversity-Aware Vehicle Motion Prediction via Latent Semantic Sampling,” Nov. 2019. [Online]. Available: https://arxiv.org/pdf/1911.12736

[49] R. Chandra et al., “Forecasting Trajectory and Behavior of Road-Agents Using Spectral Clustering in Graph-LSTMs,” Dec. 2019. [Online]. Available: https://arxiv.org/pdf/1912.01118

[50] H. Ma, J. Li, W. Zhan, and M. Tomizuka, “Wasserstein Generative Learning with Kinematic Constraints for Probabilistic Interactive Driving Behavior Prediction,” in IEEE Intelligent Vehicles Symposium, 2019, pp. 2477–2483.

[51] T. Phan-Minh, E. C. Grigore, F. A. Boulton, O. Beijbom, and E. M. Wolff, “CoverNet: Multimodal Behavior Prediction using Trajectory Sets,” Nov. 2019. [Online]. Available: http://arxiv.org/pdf/1911.10298v1

[52] H. Cui et al., “Multimodal Trajectory Predictions for Autonomous Driving using Deep Convolutional Networks,” in International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 2019, pp. 2090–2096.

[53] X. Li, X. Ying, and M. C. Chuah, “GRIP: Graph-based Interaction-aware Trajectory Prediction,” Jul. 2019. [Online]. Available: http://arxiv.org/pdf/1907.07792v1

References

[54] L. Fang, Q. Jiang, J. Shi, and B. Zhou, “TPNet: Trajectory Proposal Network for Motion Prediction,” in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[55] N. Djuric et al., “Uncertainty-aware Short-term Motion Prediction of Traffic Actors for Autonomous Driving,” in The IEEE Winter Conference on Applications of Computer Vision (WACV), 2020.

[56] A. Breuer, S. Elflein, T. Joseph, J.-A. Bolte, S. Homoceanu, and T. Fingscheidt, “Analysis of the Effect of Various Input Representations for LSTM-Based Trajectory Prediction,” in IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, New Zealand, 2019, pp. 2728–2735.

[57] N. Deo and M. M. Trivedi, “Convolutional Social Pooling for Vehicle Trajectory Prediction,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPR), 2018, pp. 1468–1476.

[58] K. Messaoud, I. Yahiaoui, A. Verroust-Blondet, and F. Nashashibi, “Non-local Social Pooling for Vehicle Trajectory Prediction,” in IEEE Intelligent Vehicles Symposium, Paris, France, 2019.

[59] R. Chandra, U. Bhattacharya, A. Bera, and D. Manocha, “Traphic: Trajectory prediction in dense and heterogeneous traffic using weighted interactions,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 8483–8492.

[60] T. Zhao et al., “Multi-Agent Tensor Fusion for Contextual Trajectory Prediction,” in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[61] M. Schreiber, S. Hoermann, and K. Dietmayer, “Long-Term Occupancy Grid Prediction Using Recurrent Neural Networks,” in International Conference on Robotics and Automation (ICRA), 2019, pp. 9299–9305.

[62] M. Khakzar, A. Rakotonirainy, A. Bond, and S. G. Dehkordi, “A Dual Learning Model for Vehicle Trajectory Prediction,” IEEE Access, vol. 8, pp. 21897–21908, 2020, doi: 10.1109/ACCESS.2020.2968618.

[63] D. Ridel, N. Deo, D. Wolf, and M. Trivedi, “Scene Compliant Trajectory Forecast With Agent-Centric Spatio-Temporal Grids,” IEEE Robotics and Automation Letters, vol. 5, no. 2, pp. 2816–2823, 2020, doi: 10.1109/LRA.2020.2974393.

[64] J. Li, H. Ma, and M. Tomizuka, “Conditional Generative Neural System for Probabilistic Trajectory Prediction,” arXiv preprint arXiv:1905.01631, 2019.

[65] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. S. Torr, and M. Chandraker, “DESIRE: Distant Future Prediction in Dynamic Scenes with Interacting Agents,” Apr. 2017. [Online]. Available: http://arxiv.org/pdf/1704.04394v1

[66] F. Marchetti, F. Becattini, L. Seidenari, and A. D. Bimbo, “MANTRA: Memory Augmented Networks for Multiple Trajectory Prediction,” in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[67] Y. Yuan and K. Kitani, “Diverse Trajectory Forecasting with Determinantal Point Processes,” arXiv preprint arXiv:1907.04967.

[68] O. Sigaud and O. Buffet, Markov Decision Processes in Artificial Intelligence. London: Wiley, 2013.