

# Exploiting Sparse Semantic HD Maps for Self-Driving Vehicle Localization

Seminar Mobile Robotik

---

Max Dunger, Yannik Motzett

January 22, 2021

HTWG Konstanz – Fakultät Informatik

Autoren des Papers:

Wei-Chiu Ma, Ignacio Tartavull, Ioan Andrei Barsan, Shenlong Wang, Min Bai, Gellert Mattyus, Namdar Homayounfar, Shrinidhi Kowshika Lakshmikanth, Andrei Pokrovsky, Raquel Urtasun

(Mitarbeiter von Uber Advanced Technologies Group (Fahrdienstanbieter), ATG-Sparte wurde 12/2020 an das Roboterwagen-Start-up Aurora verkauft [8])

Inhalt des Papers: Lightway Lokalisierung für Autonome Fahrzeuge (Outdoor Localization)

Veröffentlichungsdatum: 8. August 2019

# Table of contents

1. Einleitung
2. Related Work
3. Lightweight HD Mapping
4. Localization as Bayes Inference with Deep Semantics
5. Experiments
6. Conclusion



# **Einleitung**

---



# Probleme aktueller Lokalisierungstechniken

Gängige Lokalisierung mit GPS und IMU

- Triangulation verschiedener Satelliten
- Zu große Ungenauigkeit für Autonomes Fahren

Genauere Ortserkennungstechniken mussten entwickelt werden

- Speicherung der Umwelt in Form von Geometrie, visueller Erscheinung & Semantik
- Auch Problembehaftet

## GPS:

- Erfolgt durch Triangulation verschiedener Satelliten
- Zu große Ungenauigkeit, gerade in Tunneln und neben Hochhäusern
- Um sehr große Fehler zu vermeiden: Fehlerkorrektur mit IMU (Integration über Beschleunigung, Winkelgeschwindigkeit & Magnetfeld)
- Ungeeignet für autonomes Fahren

## Probleme aktueller Ortserkennungstechniken:

- Probleme bei sich wiederholenden Mustern (z.B.: Autobahn oder Tunnel)
- Aufzeichnungen unter verschiedenen Bedingungen notwendig (z.B.: Regen, Nacht, Nebel)
- Speicherung der Geometrie (z.B.: LiDAR-Karten) benötigen sehr großen Speicherplatz
- Verwendet man Karten mit weniger Speicherplatz, ist keine metergenaue Lokalisierung möglich

# Funktionsweise der Lokalisierung

- HD-Karten enthalten Wissen über die Umwelt
- Autonome Autos müssen sich in Bezug auf die Karte lokalisieren
- Sehr strenge Anforderungen
- Vielzahl neuer, verschiedener Systementwicklungen

## **HD-Karten:**

- Sie sind ein wesentlicher Bestandteil bei autonomfahrenden Autos
- Sie beinhalten statische Karten der Umgebung, wie z.B. Fahrspuren, Ampeln, Fußgängerwege & Verkehrsregeln
- Sie beinhalten somit kodiert ein Vorwissen über jede mögliche Szene in welche ein Autonomes Auto geraten kann

# Thema dieser Arbeit

- Vorstellung einer Lightweight Lokalisierungsmethode
- Sie nutzt Fahrzeugdynamik und semantische Karte für die Lokalisierung
- Das Ziel ist eine zentimetergenaue Lokalisierung ohne der Nachteile bestehender Ortserkennungstechniken

## **Lightweight Lokalisierung:**

- Es ist kein detailliertes Wissen über Aussehen der Umwelt erforderlich (dichte Geometrie wie LiDAR-Punktwolken)
- Benötigt deutlich weniger Speicher als dichte HD-Karten

## **Semantische Karte dieses Papers:**

- Fahrspurgraphen: Liefern Informationen über die seitliche Orientierung des Fahrzeugs (horizontal)
- Standort der Verkehrsschilder: Liefert Informationen über die längliche Ausrichtung des Fahrzeugs (vertikal)

## **Fahrzeugdynamik:**

- Besteht aus Daten der IMU und des Radkodierers

## Related Work

---



# Lightweight Localization

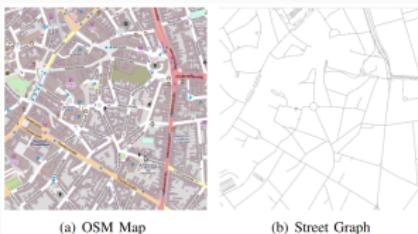
- Steigendes Interesse an kostengünstigen Lokalisierungsverfahren
- Lediglich kartografische Karten notwendig

## Vorteil:

- Deutlich weniger Speicher notwendig

## Nachteile:

- Begrenzte Genauigkeit durch Leistung der Odometrie
- Sehr große Komplexität bei sich wiederholenden Mustern



**Figure 1:** Kartographische Karten [5]

Odometrie ist eine Methode der Schätzung von Position & Orientierung eines mobilen Systems anhand Daten des Vortriebssystems. Also bei einem Roboter z.B. die Anzahl der Schritte. In diesem Fall bei einem Fahrzeug die Anzahl der Radumdrehungen

# Lightweight Localization

## Beispiele

- Lokalisierung durch Schätzung der Anfangsposition und Egotrajektorie innerhalb eines kleinen Bereichs [5]
- Mehrere Erweiterungen mit Nutzung des Sonnenstandes und des Straßentyps [12]

Zu 1: Open-Street-Maps wird in Verbindung mit Odometrie des Fahrzeugs verwendet, um eine genaue Lokalisierung zu ermöglichen.

Zu 2: Es werden nun zusätzlich semantische Hinweise in Form von Sonnenrichtung, Vorhandensein einer Kreuzung, Straßentyp & Geschwindigkeitsbegrenzungen verwendet.

# High-precision Map-based Localization

- Hochauflösende 3D-Karten: Wurde aus Offline-SLAM mit hochpräzisem GNSS-System erstellt.  
Eigenschaften: Offline, A-priori
- Im Online Modus wird ein Abgleich zwischen Sensormesswerten und a-priori Karte durchgeführt. Hierfür wird meist ein CNN verwendet.

## Vorteil:

- Sehr genaue Lokalisierungsmöglichkeiten

## Nachteil:

- Sehr großer Speicherbedarf, selbst bei kleinen Teststrecken

## Beispiel:

- CNN zur Lokalisierung mit Hilfe der LiDAR-Intensität [3]

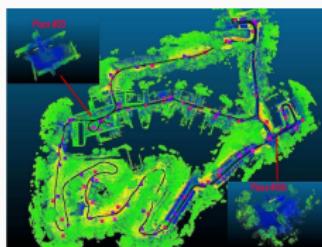
- Früher wurden solche Karten hauptsächlich mit Hilfe von LiDAR aufgenommen
- Unsicherheiten entstehen durch ein verändertes Umgebungsbild im Vergleich zu der Umgebung während der Ground-Truth-Aufnahme
- Zum Beispiel: kalibrierungsunabhängiges und effektives Echtzeit-Lokalisierungssystem für autonome Autos. Es werden Online-Lidar-Sweeps und die Intensitätskarte als Input für das CNN zur Lokalisierung verwendet

# Place Recognition

- einer der am weitesten verbreiteten Ansätze in der Selbstlokalisierung
- im Vorhinein wird von Umgebung eine 3D-Karte erstellt (z.B. Punktewolke)
- Abrufproblem: die Pose entspricht der ähnlichsten Szene

**Nachteil:** 3D-Karte muss aktuell gehalten werden, da Sensordaten stark mit Karte korrelieren)

**Erweiterung:** bildbasierter Ansatz ist robuster gegen Veränderungen, PoseNet kann direkt Kamerapose ermitteln [9]



**Figure 2:** Place Recognition mit LiDAR-Punktewolke [7]

Bemerkungen:

PoseNet (Convolutional Neural Network) ist außerdem robust gegen Helligkeits- und Witterungsänderungen, benötigt aber für jede Szene Trainingsdaten, wodurch dieser Ansatz nicht skalierbar ist

# Gemoetry-based Localization

Lösen von Perspective-n-Point (PnP)-Problem: Übereinstimmung von Bild mit voraufgenommenem 3D-Bild

**Algorithmen:** random forests, SIFT, Branch-and-Bound

**Nachteile:** 3D-Modell muss im Vorhinein berechnet werden, funktioniert nicht mit wiederholenden geometrischen Strukturen

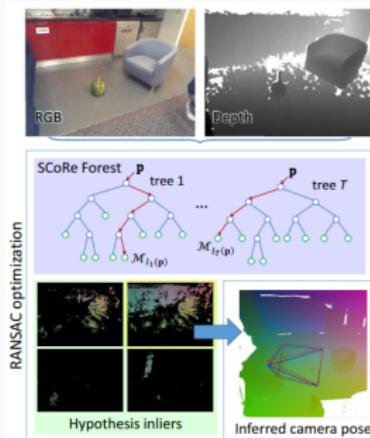


Figure 3: Regression Forests [14]

Beim *PnP-Problem* geht es um die Schwierigkeit, die Kamerapose im Raum anhand von Merkmalen im Bild und dem Wissen, welchen 3D-Punkten im Raum sie zugeordnet sind, rechnerisch zu bestimmen.

Aufgenommenes Bild kann 2D-RGB Bild oder auch RGBD (Tiefenbild) sein.

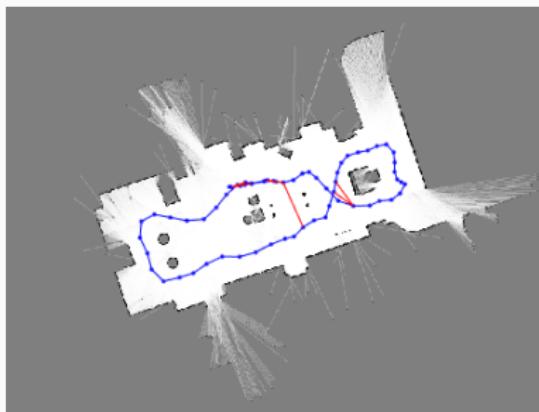
Bei *Regression Forrests* wird ein Random Forrest trainiert. Diesem werden Pixelgruppen eines Bildes übergeben. Der Forrest liefert die Pose, in welcher das Input-Bild mit einem Bild des 3D-Modells die größte Übereinstimmung hat.

*Scale-invariant feature transform* (SIFT) ist ein Algorithmus zur Detektion und Beschreibung lokaler Merkmale in Bildern. Der Detektor und die Merkmalsbeschreibungen sind, in gewissen Grenzen, invariant gegenüber Koordinatentransformationen wie Translation, Rotation und Skalierung.

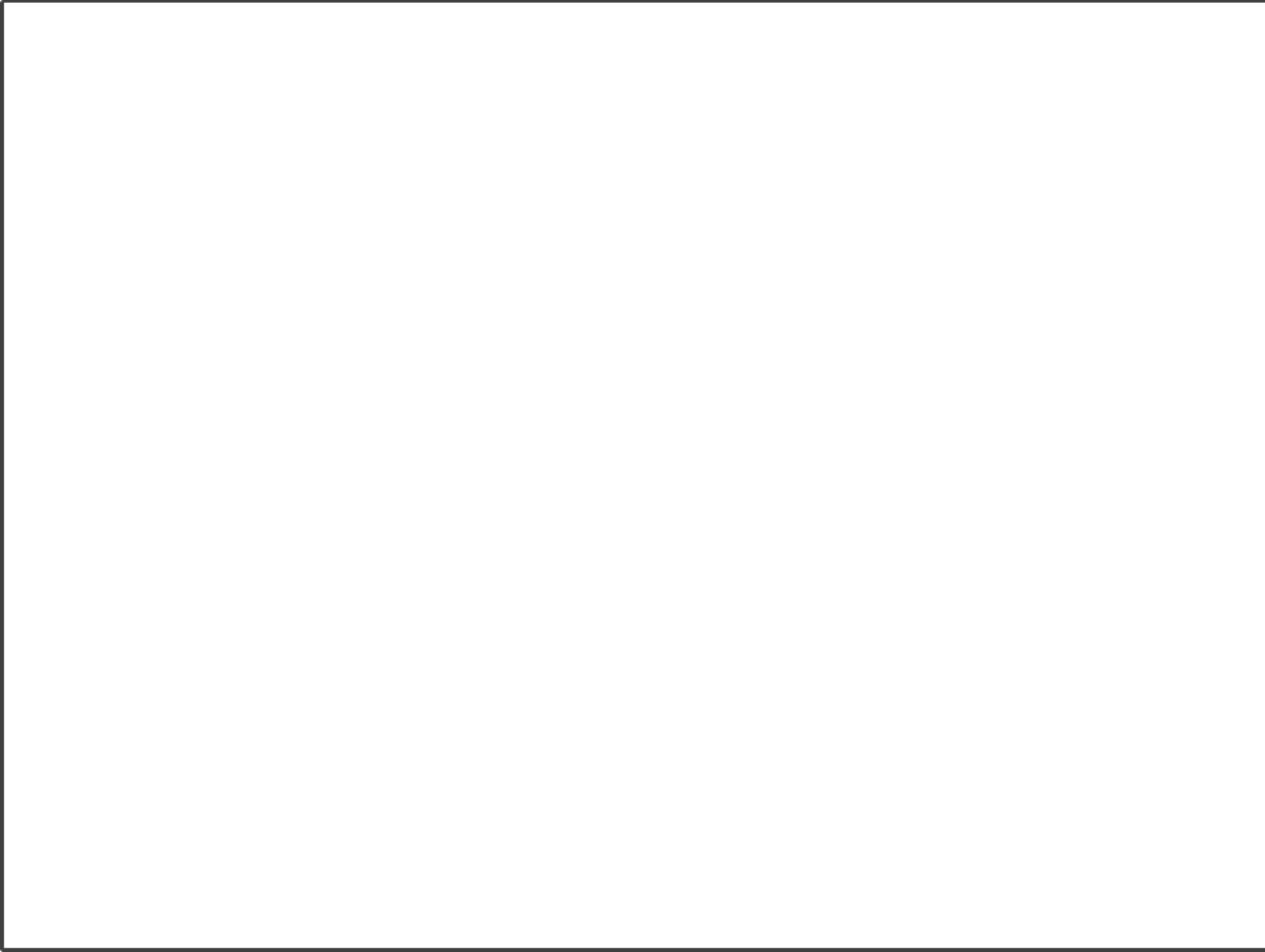
*Branch-and-Bound* ist eine im Bereich Operations Research häufig verwendete mathematische Methode, deren Ziel darin besteht, für ein gegebenes ganzzahliges Optimierungsproblem eine beste Lösung zu finden.

# Simultaneous Localization and Mapping (SLAM)

aus der Folge von Sensordaten (Bilder, Punktewolken, Tiefenbildern) wird eine 3D-Karte der Umgebung erstellt und die Kameraposition geschätzt  
Nachteil: während sich der Roboter bewegt, summieren sich Fehler auf



**Figure 4:** LiDAR Indoor SLAM [13]



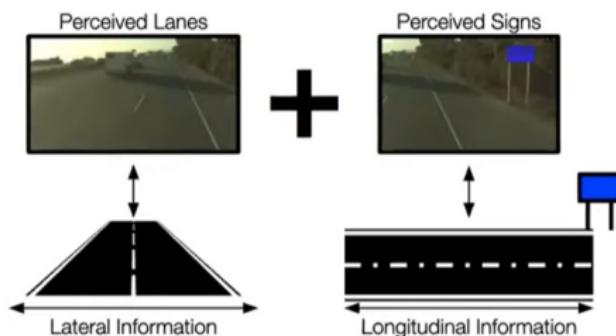
## **Lightweight HD Mapping**

---



# Anforderungen an neue HD-Map

- einfache bzw. automatische Erstellung und Instandhaltung
- Skalierbarkeit
- Änderungen sollen in Echtzeit an Flotte bereitgestellt werden
- geringer Speicherbedarf



**Figure 5:** Sparse HD-Maps [15]

Die im Paper vorgestellte spärliche HD-Map enthält nur Lane Graph und Position von Verkehrschilder. Spurmarkierung geben Hinweise in laterale Richtung, Verkehsschilder in Längssrichtung.

# Lane Graph

## Lane Graph $\mathcal{L}$

- Ist in der realen Welt meistens vorhanden
- Zeigen die erwartete Trajektorie der Fahrzeuge auf
- Strukturierte Darstellung des Straßennetzes aus einer Menge polygonaler Ketten (Polylinien)
- Liefert Informationen über die seitliche Position und den Kurs des Fahrzeuges

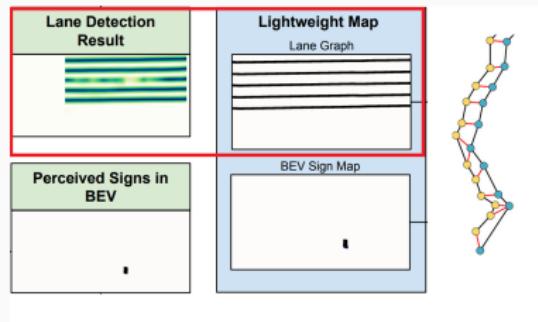


Figure 6: Lane Graph [11]

- Jede Polyline stellt eine Fahrspur dar
- Trajektorie bedeutet in der Physik etwas wie "Bahnkurve", "Pfad" oder "Weg"

# Traffic Signs

- Verkehrsschilder geben Hinweise für Lokalisierung in Längsrichtung
- es wird automatisch Map erstellt, die Informationen über Schilder enthalten

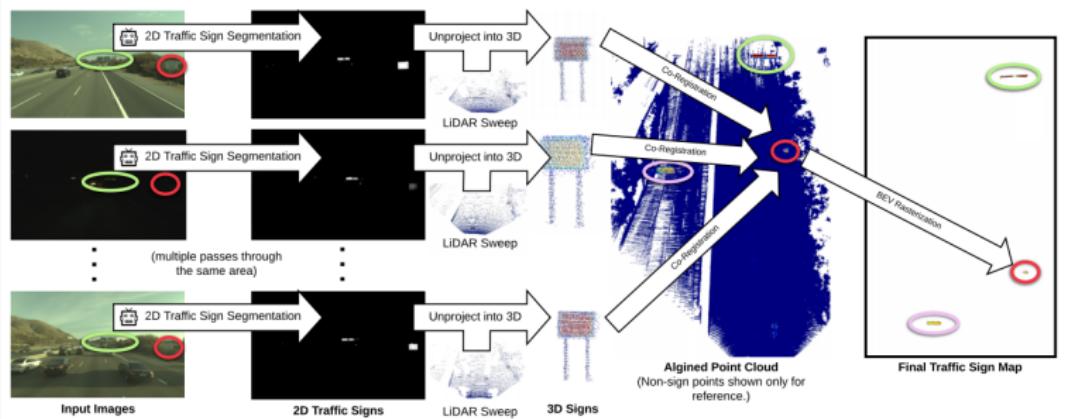


Figure 7: Erstellungsprozess Sign Map [11]

Erstellung der Traffic Sign Map mit Multisensor-SLAM: Fahrzeug mit Kamera und LiDAR fährt mehrfache die gleiche Strecke ab, diese Durchläufe werden ausgewertet, die Erkennung der Schilder erfolgt durch Bild-basierte semantische Segmentierung. Anschließend werden die Pixel mithilfe der 3D-Daten des einen LiDAR-Sensors in 3D-Raum projiziert. Abgespeichert werden die Bildpunkte der Schilder in einer Sign Map  $\mathcal{T}$  in Vogelperspektive (Auflösung von 5cm pro Pixel).

# **Localization as Bayes Inference with Deep Semantics**

---



# Probabilistic Pose Filter Formulation

## Aufbau



**Figure 8:** Zusammensetzung der Wahrscheinlichkeitsformel

- Wahrscheinlichkeits-Histogramm über Position des Fahrzeugs in Weltkoordinaten
- Alle Sensoren sind üblich bei der Nutzung autonomer Fahrzeuge
- GPS: Positionsbestimmung auf mehrere Meter
- IMU: Fahrdynamische Messung (Beschleunigung, Winkelgeschwindigkeit & Magnetfeld)
- Rad-Encoder: Liefert Informationen über den Gesamtfahrweg
- Lidar: Erfasst die genaue Geometrie der Umgebung
- Bilder-Kamera: Erfassen Erscheinungsbildinformationen
- Annahme: Alle Sensoren kalibriert und Fehler werden vernachlässigt (Federung, Vibration, Unwucht der Räder) Ein späteres Experiment bestätigt diese Vernachlässigbarkeit dieser Fehler

# Probabilistic Pose Filter Formulation

Wahrscheinlichkeitsformel für die Positionsbestimmung

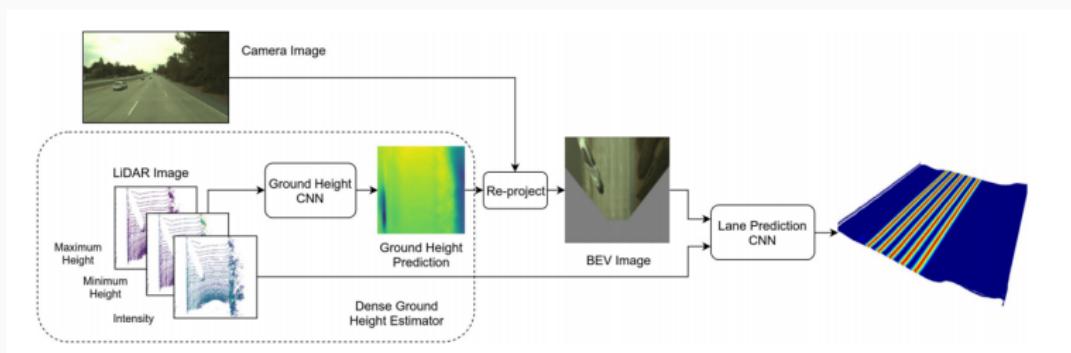
$$\text{Bel}_t(\mathbf{x}) = \eta \cdot P_{\text{LANE}}(\mathcal{S}_t | \mathbf{x}, \mathcal{L}; \mathbf{w}_{\text{LANE}}) P_{\text{SIGN}}(\mathcal{S}_t | \mathbf{x}, \mathcal{T}; \mathbf{w}_{\text{SIGN}})$$
$$P_{\text{GPS}}(\mathcal{G}_t | \mathbf{x}) \text{Bel}_{t|t-1}(\mathbf{x} | \mathcal{X}_t),$$

- $Bel$ : Wahrscheinlichkeit für Position zur Zeit t
- $\eta$ : Normierungsfaktor Summe aller Wahrscheinlichkeiten sollen 1 ergeben
- $P_{Lane}$ : Wahrscheinlichkeit der Position ermittelt über Fahrspuren
- $S_t$ : sensorisches Messtupel aus Lidar und Kamera (Input)  $S_t = (I_t, C_t)$
- $L$ : Fahrspurgraphen
- $w_{LANE}$ : Lernparameter
- $P_{SIGN}$ : Wahrscheinlichkeit der Position ermittelt über Verkehrsschilder
- $T$ : Verkehrsschilder-Karte
- $w_{SIGN}$ : Lernparameter
- $P_{GPS}$ : Wahrscheinlichkeit der Position ermittelt über GPS
- $\mathcal{G}_t = \text{GPS zum Zeitpunkt t}$
- $Bel_{t|t-1}$ : Wahrscheinlichkeit der Position durch Fahrzeugdynamik
- $X_t = \text{Fahrzeugdynamik durch IMU \& Rad-Encoder: Kalman-Filter mit Update von 100Hz}$

# Probabilistic Pose Filter Formulation

## Lane Observation Model

$$\text{Bel}_t(\mathbf{x}) = \eta [P_{\text{LANE}}(\mathcal{S}_t | \mathbf{x}, \mathcal{L}; \mathbf{w}_{\text{LANE}}) P_{\text{SIGN}}(\mathcal{S}_t | \mathbf{x}, \mathcal{T}; \mathbf{w}_{\text{SIGN}}) \\ P_{\text{GPS}}(\mathcal{G}_t | \mathbf{x}) \text{Bel}_{t-1}(\mathbf{x} | \mathcal{X}_t),$$



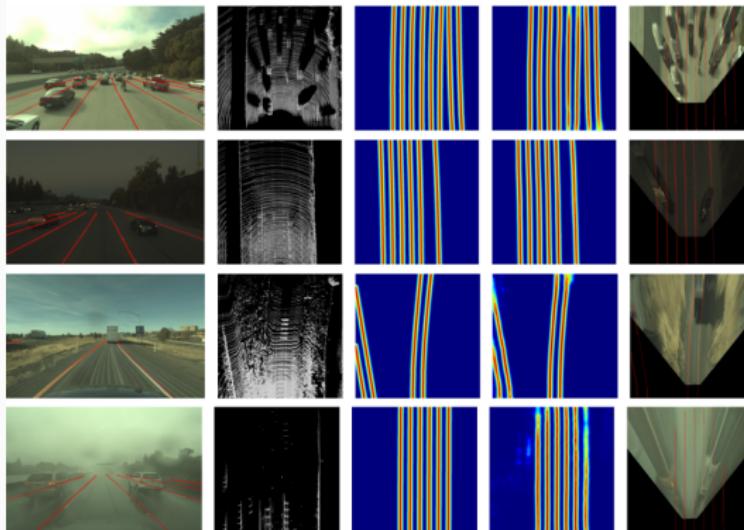
**Figure 9:** Aufbau des Lane Observation CNNs [2]

Am rechten Ende ist der Output des trainierten CNNs zu sehen: Die detektierten Fahrspuren. Input für dieses CNN ist einerseits eine aktive LiDAR-Intensitätsmessung und andererseits das aktuelle Kamerabild in Birds Eye View. Um jedoch das ursprüngliche Kamerabild in die Vogelperspektive zu projizieren sind weitere Schritte nötig. Die x- & y-Werte des Kamerabildes ändern sich aufgrund der festinstallierten Kamera nicht. Die Höhe z kann aufgrund von steigenden Straßen jedoch schwanken. Somit ist ein weiteres CNN nötig, welches diese Höhe vorraussagt. Anschließend kann das Kamerabild über eine Rotationsmatrix in Birds Eye View projiziert werden.

# Probabilistic Pose Filter Formulation

## Lane Observation Model

$$\text{Bel}_t(\mathbf{x}) = \eta [P_{\text{LANE}}(\mathcal{S}_t | \mathbf{x}, \mathcal{L}; \mathbf{w}_{\text{LANE}}) P_{\text{SIGN}}(\mathcal{S}_t | \mathbf{x}, \mathcal{T}; \mathbf{w}_{\text{SIGN}}) \\ P_{\text{GPS}}(\mathcal{G}_t | \mathbf{x}) \text{Bel}_{t|t-1}(\mathbf{x} | \mathcal{X}_t),$$



**Figure 10:** Beispiele der Fahrspurenerkennung [2]

## CNN Ergebnisse bei verschiedenen Bedingungen (Verkehr, Nacht, Brücken)

- (1 Spalte) Die erkannten Fahrspuren werden in das aufgezeichnete Kamerabild projiziert. Die Fahrspuren werden bis zu 48 m vorausgesagt.
- (2 Spalte) LiDAR-Bild
- (3 Spalte) Die Ground-Truth Linien
- (4 Spalte) Die erkannten Fahrspuren des CNNs
- (5 Spalte) Kamerabild mit erkannten Fahrspuren in Birds Eye View (aus Kombination zwischen Ground-Truth-Linien & erkannten Fahrspurgraphen)

Fahrzeuge können Linien verdecken, Linien können verschmutzt sein oder es kann sich Regen auf dem Sensor befinden: Durch die Nutzung mehrerer Informationsquellen ist trotzdem eine qualitativ hochwertige und ausreichende Fahrspurerkennung möglich.

# Probabilistic Pose Filter Formulation

## Lane Observation Model

$$\text{Bel}_t(\mathbf{x}) = \eta [P_{\text{LANE}}(\mathcal{S}_t | \mathbf{x}, \mathcal{L}; \mathbf{w}_{\text{LANE}}) P_{\text{SIGN}}(\mathcal{S}_t | \mathbf{x}, \mathcal{T}; \mathbf{w}_{\text{SIGN}}) \\ P_{\text{GPS}}(\mathcal{G}_t | \mathbf{x}) \text{Bel}_{t|t-1}(\mathbf{x} | \mathcal{X}_t),$$

Übereinstimmung zwischen Fahrspurbeobachtung und Karte

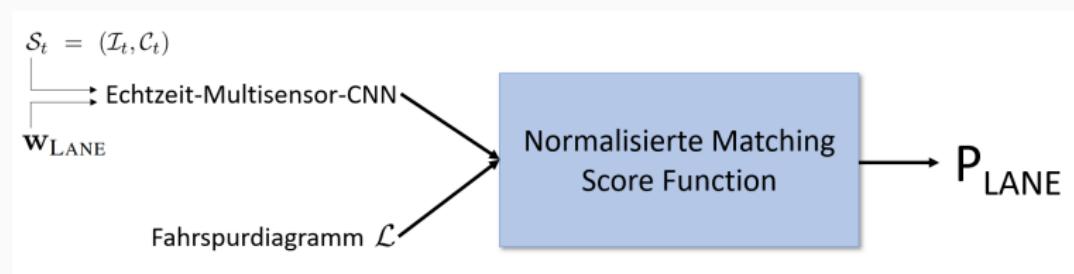


Figure 11: Matching Score Function

Die Ergebnisse des eben vorgestellten CNNs zur Detektion von Fahrspuren und das vorher erstelle Fahrspurendiagramm werden in einer Matching Score Function abgeglichen. Hierfür wird der aufgenommene Fahrspurengraph auf eine Draufsicht (Birds Eye View) projiziert sodass die Ausgabe der Spurenerkennung und die vorher erstelle Karte in einem Koordinatensystem liegen. Nun wird bei einer gegebenen Hypothese für die Fahrzeugposition  $x$  durch drehen und verschieben des Graphen eine neue Vorhersage getroffen. Mathematisch berechnet die Matching Score Function das innere Produkt zwischen beiden Graphen.

# Probabilistic Pose Filter Formulation

## Traffic Sign Observation Model

$$\text{Bel}_t(\mathbf{x}) = \eta \cdot P_{\text{LANE}}(\mathcal{S}_t | \mathbf{x}, \mathcal{L}; \mathbf{w}_{\text{LANE}}) P_{\text{SIGN}}(\mathcal{S}_t | \mathbf{x}, \mathcal{T}; \mathbf{w}_{\text{SIGN}})$$
$$P_{\text{GPS}}(\mathcal{G}_t | \mathbf{x}) \text{Bel}_{t|t-1}(\mathbf{x} | \mathcal{X}_t),$$

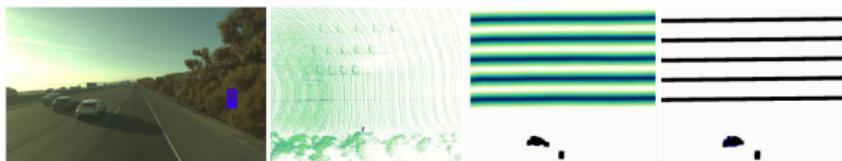


Figure 12: Traffic Sign Model [11]

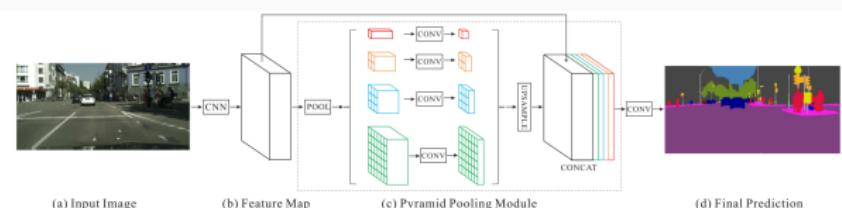


Figure 13: PSPnet [16]

Erkennung der Schilder basiert auf PSPnet (Proposed Pyramid Scene Parsing Network), welches vortrainiert wurde

$w_{SIGN}$ : Netzwerkparameter von Sign Segmentation Network

Umrechnung in online BEV Map, indem jeder Bildpixel einem LiDARs-Strahl zugeordnet wird

um Pose zu ermitteln wird Auschnitt der Online Map in der Offline Map gesucht

# Probabilistic Pose Filter Formulation

## GPS Observation Model

$$\text{Bel}_t(\mathbf{x}) = \eta \cdot P_{\text{LANE}}(\mathcal{S}_t | \mathbf{x}, \mathcal{L}; \mathbf{w}_{\text{LANE}}) P_{\text{SIGN}}(\mathcal{S}_t | \mathbf{x}, \mathcal{T}; \mathbf{w}_{\text{SIGN}})$$
$$P_{\text{GPS}}(\mathcal{G}_t | \mathbf{x}) \text{Bel}_{t|t-1}(\mathbf{x} | \mathcal{X}_t),$$

- GPS-Messung  $\mathcal{G}_t$  liefert Wahrscheinlichkeit  $P_{\text{GPS}}(\mathcal{G}_t | \mathbf{x})$
- Fahrzeug kann auf wenige Meter genau lokalisiert werden
- mit UTM-Koordinaten (Universal Transverse Mercator) wird Fahrzeug auf HD-Karte lokalisiert

GPS-Koordinaten sind räumlichen Polarkoordinaten mit Breitengrad, Längengrad und Höhe

UTM-Koordinaten spannt die Erdoberfläche auf ein zweidimensionales kartesisches Koordinatensystem auf und unterteilt sie in Zonen

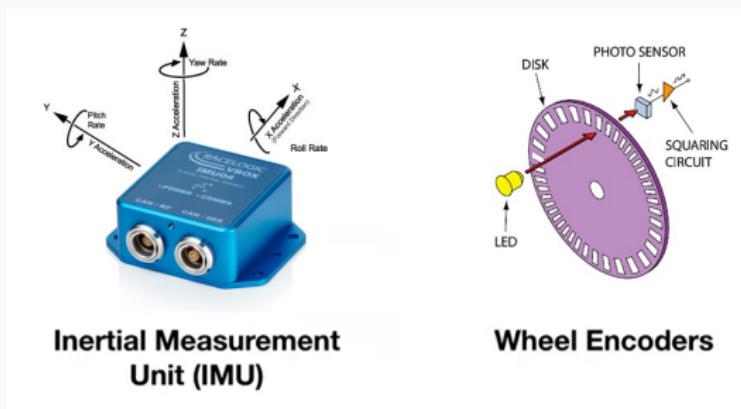
gemessener UTM-Punkt wird in auf HD-Map transformiert  $[g_x, g_y]^T = T * \mathcal{G}$

# Probabilistic Pose Filter Formulation

## Dynamics Model

$$\text{Bel}_t(\mathbf{x}) = \eta \cdot P_{\text{LANE}}(\mathcal{S}_t | \mathbf{x}, \mathcal{L}; \mathbf{w}_{\text{LANE}}) P_{\text{SIGN}}(\mathcal{S}_t | \mathbf{x}, \mathcal{T}; \mathbf{w}_{\text{SIGN}}) \\ P_{\text{GPS}}(\mathcal{G}_t | \mathbf{x}) \boxed{\text{Bel}_{t|t-1}(\mathbf{x} | \mathcal{X}_t)}$$

aus vorausgegangener Pose wird aus Fahrzeuggbewegung mit Extended Kalman Filter die aktuelle Pose geschätzt



**Inertial Measurement  
Unit (IMU)**

**Wheel Encoders**

**Figure 14:** Sensoren für Dynamik Model [15]

IMU (Inertial Measurement Unit) misst die Fahrzeugbewegungen (z.B. Gierrate), ein Raddrehzahlsensoren misst die zurückgelegte Strecke

EKF (Extended Kalman Filter) bestimmt mit Sensordaten aus Pose des vergangenen Zeitschritts die aktuelle Position indem Messfehler geschätzt und berücksichtigt werden, wird mit 100 Hz aktualisiert

# Systemarchitektur

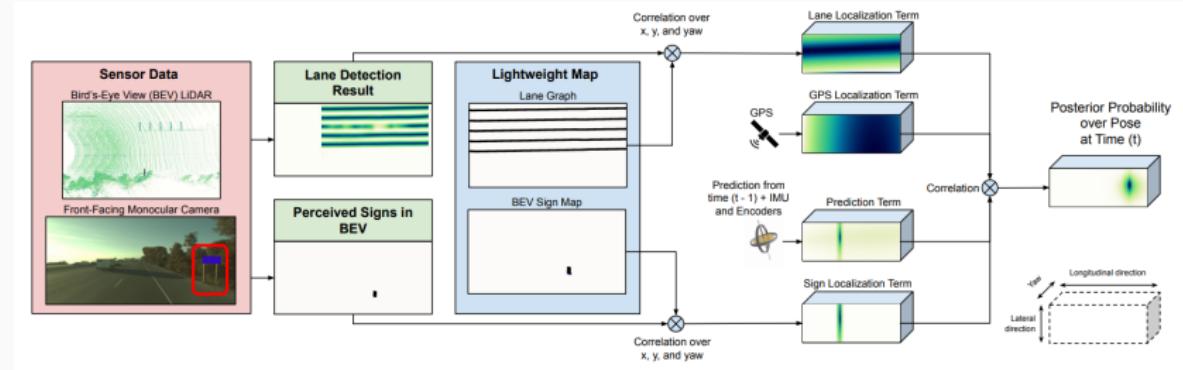


Figure 15: Systemarchitektur [11]

Alle vier Modelle liefern Wahrscheinlichkeitsverteilung der aktuellen Pose, Wahrscheinlichkeitsverteilungen werden miteinander verknüpft und man erhält die Posterior Wahrscheinlichkeit der Fahrzeugpose.

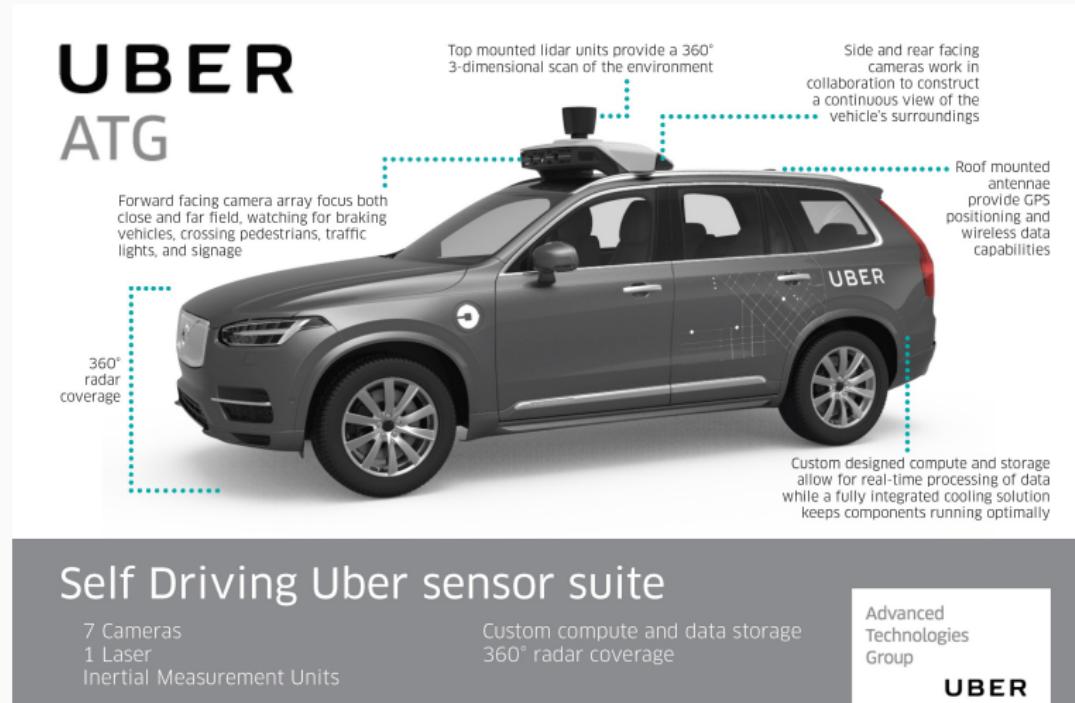
Anhand der Heat Map der berechneten Wahrscheinlichkeitsverteilungen sieht man, dass die Spurlokalisierung in lateraler Richtung und die Schildlokalisierung in Längsrichtung eine geringe Varianz haben.

## Experiments

---



# Sensors



**Figure 16:** Uber Sensor Set – Volvo XC90 [4]

- 64-Linien-LiDAR-Sweeps mit 10 Hz (Velodyne HDL-64E)
- Gerichtete Globale-Shutter-Kamera mit 1900x1280 Auflösung und Aufnahmefrequenz von 10 Hz
- IMU
- GPS
- Sensorkalibrierung nach externem Paper [1]

# Datenset

- 312 km nordamerikanische Autobahn
- Aufteilung in 2 km Datensatzschnipsel
- Ground-Truth-3D-Lokalisierung: Geschätzt durch hochpräzises IPC-basiertes Offline-Graphen-SLAM
- Aufteilung der Daten in Trainings-, Validierungs- und Testdaten

Offline Graphen SLAM unter verwendung hochauflösender vorabgescannter Szenen-Geometrie

# Fahrspurerkennungs-CNN

- 50k Frames auf Trainingsregion
- Mini-Batch-Größe: 16
- Lernrate:  $10^{-4}$
- Gaußsche Initialisierung
- Konvergenz nach 10 Epochen

Optimierer aus externem Paper [10]

# Verkehrszeichen-CNN

- Separat auf 4 GPUs
- Gesamt-Mini-Batch-Größe: 8
- Lernrate:  $10^{-4}$
- Training des Netzwerkes von Grund auf

Für Multi-GPU-Batch Normalisierung: synchrone Batch-Normalisierung

# Performance Analyse

- Laufzeitanalyse: 7 Hz
- Speicheranalyse: 0,55 MiB pro Quadratmeter

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

Figure 17: Genauigkeit [11]

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

Figure 18: Vergleich Anteil Komponenten [11]

**Laufzeitanalyse:** Ein Rechenschritt auf NVIDIA GTX 1080 GPU benötigt durchschnittlich 153 ms (32 ms Spurerkennung, 110 ms Semantische Segmentierung, 11 ms Matching) → 7 fps

**Genauigkeit:** Median-Fehler in Längsrichtung 1,12 m, Median-Fehler in laterale Richtung 0,05 m

**Speicheranalyse der Map:** HD-Map benötigt 0,55 MiB pro Quadratmeter (0,3 % von LiDAR Intensitäts Karte, 0,03 % von Punktewolkenkarte)

**Vergleich Anteil Komponenten:** alle Komponenten haben positiven Effekt auf Genauigkeit; Lane Observation Model verbessert Genauigkeit speziell in laterale Richtung, Sign Observations verbessert Genauigkeit in Längsrichtung

## Conclusion

---



# Conclusion

- Karte benötigt 3-mal weniger Speicher als traditionelle Methoden
- Potential: Skalierbarkeit und geringere Kosten bei Ausrollen von Updates
- laterale Lokalisierung durch Spurmarkierungen, in Längsrichtung mit Verkehrsschilder
- Algorithmus wird durch GPS, IMU und Radsensoren unterstützt
- System läuft in Echtzeit mit 7 Hz
- getestet auf 300 km Autobahnabschnitt

Kritik:

Algorithmus funktioniert nicht, wenn Fahrbahnmarkierungen bzw. Schilder nicht vorhanden oder erkennbar sind (z.B. bei Schnee)

Funktioniert auch nicht wenn Verkehrsschilder zu tief montiert sind

LiDAR-Sensor wird benötigt, die aktuell noch sehr teuer sind (Velodyne HDL-64E: 75,000 € [6])

bisher nur auf Autobahn getestet, auf Landstraße mit Straßenpfosten denkbar

**Questions?**

## References i

- [1] R. Arandjelovic, P. Gronát, A. Torii, T. Pajdla, and J. Sivic.  
**Netvlad: CNN architecture for weakly supervised place recognition.**  
*CoRR*, abs/1511.07247, 2015.
- [2] M. Bai, G. Mátyus, N. Homayounfar, S. Wang, S. K. Lakshmikanth, and R. Urtasun.  
**Deep multi-sensor lane detection.**  
*CoRR*, abs/1905.01555, 2019.

## References ii

- [3] I. A. Barsan, S. Wang, A. Pokrovsky, and R. Urtasun.  
**Learning to localize using a lidar intensity map.**  
In A. Billard, A. Dragan, J. Peters, and J. Morimoto, editors,  
*Proceedings of The 2nd Conference on Robot Learning*, volume 87  
of *Proceedings of Machine Learning Research*, pages 605–616.  
PMLR, 29–31 Oct 2018.
- [4] D. Etherington.  
**Uber's self-driving cars start picking up passengers in san francisco.**  
<https://techcrunch.com>.  
Eingesehen am 30.12.2020.

## References iii

- [5] G. Floros, B. Van der Zander, and B. Leibe.  
**Openstreetslam: Global vehicle localization using openstreetmaps.**  
pages 1054–1059, 05 2013.
- [6] F. Greis.  
**Laserscanner für den massenmarkt kommen.**  
<https://www.golem.de>.  
Eingesehen am 14.01.2021.
- [7] J. Guo, P. V. K. Borges, C. Park, and A. Gawel.  
**Local descriptor for robust place recognition using lidar intensity.**  
*IEEE Robotics and Automation Letters*, 4(2):1470–1477, 2019.

- [8] Handelsblatt.  
**Über steigt aus roboTerwagen-entwicklung aus.**  
<https://www.handelsblatt.com>.  
Eingesehen am 15.01.2021.
- [9] A. Kendall, M. Grimes, and R. Cipolla.  
**Convolutional networks for real-time 6-dof camera relocalization.**  
*CoRR*, abs/1505.07427, 2015.
- [10] D. Kingma and J. Ba.  
**Adam: A method for stochastic optimization.**  
*International Conference on Learning Representations*, 12 2014.

## References v

- [11] W. Ma, I. Tartavull, I. A. Bârsan, S. Wang, M. Bai, G. Mátyus, N. Homayounfar, S. K. Lakshmikanth, A. Pokrovsky, and R. Urtasun.  
**Exploiting sparse semantic HD maps for self-driving vehicle localization.**  
*CoRR*, abs/1908.03274, 2019.
- [12] W.-C. Ma, S. Wang, M. Brubaker, S. Fidler, and R. Urtasun.  
**Find your way by observing the sun and other semantic cues.**  
06 2016.
- [13] MathWorks.  
**Implement simultaneous localization and mapping (slam) with lidar scans.**  
<https://mathworks.com>.  
Eingesehen am 28.12.2020.

- [14] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. W. Fitzgibbon.  
**Scene coordinate regression forests for camera relocalization in rgb-d images.**  
In *CVPR*, pages 2930–2937. IEEE Computer Society, 2013.
- [15] R. Urtasun.  
**Localization for self-driving.**  
[https://www.youtube.com/watch?v=V8JMwE\\_L5s0](https://www.youtube.com/watch?v=V8JMwE_L5s0).  
Eingesehen am 29.12.2020.
- [16] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia.  
**Pyramid scene parsing network.**  
In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6230–6239, 2017.