



## Master's Thesis

# Autonomous Driving in Urban Centers - Roundabout Monitoring

Julian-B. Scholle  
March 12, 2017

Otto-von-Guericke-Universität Magdeburg  
Fakultät für Informatik  
Universitätsplatz 2  
39106 Magdeburg  
Germany

---

---

# Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbständig und ohne unerlaubte fremde Hilfe angefertigt, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Stellen sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keinem anderen Prüfungsamt vorgelegt und auch nicht veröffentlicht.

Göteborg, den March 12, 2017

---

Julian-B. Scholle

---

# Acknowledgements

Danken möchte ich außerdem besonders Associate professor Christian Berger und J. Prof Sebastian Zug, für ihre Organisatorische und Fachliche Unterstützung während und besonders im Vorfeld dieser Arbeit und dem “Deutschen Akademischen Austauschdienst” - DAAD für ihre finanzielle Unterstützung während meines Aufenthaltes in Göteborg. Weiterhin danken möchte ich natürlich allen weiteren Kollegen aus Göteborg, welche mich bei meiner Arbeit fachlich und moralisch unterstützt haben.

---

# Index of Abbreviations

**DBSCAN** Density-Based Spatial Clustering of Applications with Noise

**SVD** Singular Value Decomposition

**LLSQ** Linear least Squares

**RANSAC** Random Sample Consensus

**MKS** multi-body simulation

---

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Ausgangssituation . . . . .	1
1.1.1	Test Platform . . . . .	1
1.2	Zielsetzung . . . . .	3
<b>2</b>	<b>Basic Knowledge</b>	<b>4</b>
2.1	Autonomous Driving . . . . .	4
2.2	Roundabouts in Law . . . . .	4
2.2.1	Elements of a Roundabout . . . . .	4
2.2.2	Types of Roundabouts . . . . .	5
2.3	Random Sample Consensus . . . . .	7
2.4	Density-Based Spatial Clustering of Applications with Noise . . . . .	8
2.5	Middleware OpenDAVINCI . . . . .	9
<b>3</b>	<b>Methodology</b>	<b>10</b>
<b>4</b>	<b>Research</b>	<b>12</b>
4.1	Objekt Detection . . . . .	12
4.1.1	Ground Removal . . . . .	12
4.1.2	Clustering . . . . .	16
4.1.3	Tracking . . . . .	17
4.1.4	Classification . . . . .	21
4.1.5	State Estimation . . . . .	21
4.2	Simulation . . . . .	22
4.2.1	Simulation Scenario . . . . .	22
4.2.2	Simulation Logic . . . . .	23

<b>5</b>	<b>Evaluation</b>	<b>25</b>
5.1	Simulation . . . . .	25
5.2	Real Measurements . . . . .	25
<b>6</b>	<b>Conclusions</b>	<b>26</b>
<b>7</b>	<b>Future Work</b>	<b>27</b>

Das autonome Fahren und die Vernetzung von Fahrzeugen mit Ihrer Umwelt sind zusammen mit der Elektromobilität die mestdiskutierten Themen der Automobilbranche. Zu Recht: Autonomes Fahren besitzt das Potenzial, im Mobilitätsmarkt völlig neue Strukturen entstehen zu lassen. <sup>1</sup>

So ebenfalls die Technische Hochschule Chalmers welche ergänzend zu Volvos “DriveMe” Projekt das Projekt “CampusShuttle” initiiert hat, “CampusShuttle” ist ein interdisziplinäres Forschungsprojekt der Technischen Hochschule Chalmers und der Universität Göteborg. Das Projekt ist dabei im ReVeRe (Chalmers Research Vehicle Resource) angesiedelt. Die Vision ist dabei ein selbstfahrendes Auto zwischen den beiden Campus der Technische Hochschule Chalmers.

Dabei soll, im Rahmen des Projekts, das Fahrzeug in verschiedenen Verkehrsszenarien untersucht werden. Der Fokus liegt dabei besonders auf den Stadtverkehr, das Fahrzeug muss dabei nicht nur in der Lage sein mit anderen Autos zu interagieren, sondern ebenfalls mit Straßenbahnen, Bussen, Fahrrädern und allen Anderen Verkehrsteilnehmern sicher agieren.

---

## 1.1 Ausgangssituation

### 1.1.1 Test Platform

Die in dieser Arbeit genutzte Testplattform ist ein Volvo XC90 (2015) SUV, gennant Snowfox. Diese Testplattform ist mit vielen Sensoren zur Umfeldwahrnehmung ausgestattet. Dazu zählen fünf Radar Sensoren, rund um das Fahrzeug. Wobei das Front Radar über eine Größere Reichweite verfügt. Sowie eine Stereo Kamera und ein Velodyne VLP-16 LiDAR. Die Anordnung der Sensoren kann fig. 2.7 entnommen werden.

Zusätzlich zur Serienmäßigen Fahrzeugsensorik (z.B. Odometer, Interiärsensorik) ist im Fahrzeug ein Applanix POS LV verbaut. Zu Zeitpunkt des Verfassens dieser Arbeit war es leider noch nicht möglich auf die Radarsensoren

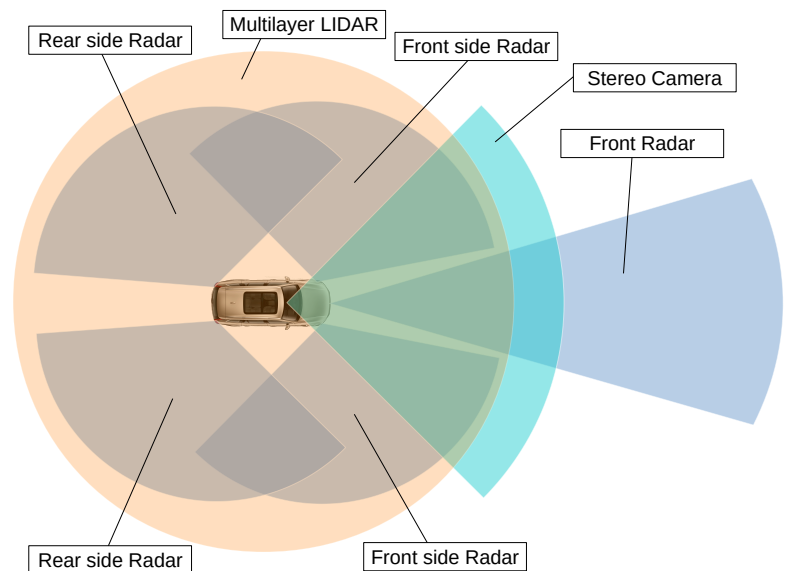
---

1. <https://www2.deloitte.com/de/de/pages/consumer-industrial-products/articles/autonomes-fahren-in-deutschland.html> (03/09/2017)

Figure 1.1: Test Platform  
Snowfox



Figure 1.2: Snowfox Sensors



und die Stereokamera zuzugreifen. Daher werden im folgenden lediglich der Velodyne Lidar und das Applanix System genauer beschrieben.

### Velodyne VLP-16 LiDAR

reichweitenanalyse (layer object mit entfernung)

Der Velodyne VLP-16 ist ein 360 Grad 3D Laserscanner mit einer Rotationsgeschwindigkeit von 5 bis 20 Umdrehungen pro Sekunde. Er bietet ein vertikales FOV von 30 Grad, bei 2 Grad Auflösung. Mit einer Reichweite von 100m kann er einen Umkreis von 200m Durchmesser abdecken. Weiterhin kann der VLP-16 mit dem Applanix POS LV synchronisiert werden, was eine jitterarme Zeimessung ermöglicht. Eine weitere Funktion des Velodyne Sensors, ist das er auf verschiedene Messimpulse reagieren kann. Durch die Auswertung des letzten Impulses statt des Stärksten Impulses ist es möglich durch Transparente Objekte zu sehen. Das ermöglicht uns im späteren Verlauf die Breite des Fahrzeuges zu ermitteln, da der Velodyne durch die Glasfenster des Fahrzeuges blicken kann. Bei einer eingestellten Geschwindigkeit von



10Hz liefert der VLP-16 eine Auflösung von 0.2 Grad bei einer Abweichung von  $\pm 3$ cm. Der VLP-16 ist mittig auf dem Dach des XC90 montiert, um eine möglichst hohe Positionierung zu erreichen, die eine Rundumsicht um das Fahrzeug zu erreichen. Zu beachten ist, dass diese Ausrichtung für den Sensor denkbar ungünstig ist, da der Sensor ein vertikales Sichtfeld von  $-15$  bis  $+15$  Grad hat. Dadurch sind nahezu alle Messungen über Null Grad quasi nutzlos. Der Blick auf die Herstellerseite <sup>2</sup> verrät, dass der VLP-16 außerdem auf die Verwendung mit Drohnen hin konstruiert wurde, während der Größere HDL64E <sup>3</sup> explizit für den Urbanen Automotivbereich beworben wird, und über ein Sichtfeld von  $+2$  bis  $-24.9$  Grad verfügt. Die dabei entstehenden Probleme werden später diskutiert.

### **Applanix POS LV**

Das POS LV ist ein kompaktes Positions- und Orientierungssystem. Es offeriert stabile, zuverlässige und reproduzierbare Positionierungslösungen für landgestützte Fahrzeuganwendungen. Das POS LV liefert dabei eine Inertialsensorik und Odometrie gestützte Positionsmessung mit einer Genauigkeit von bis zu 0.3m (bis zu 0.035m bei Verwendung von der RTK - Korrektur). Im weiteren Verlauf wird außerdem das vom POS LV gelieferte Heading genutzt, welches eine Genauigkeit von 0.2 Grad liefert. Auch nach Ausfall des GPS-Signals kann das POS-LV durch sein Odometer und der Inertialsensorik eine Position liefern. Diese wird jedoch über die Zeit schlechter, so dass 60Sek nach Ausfall des GPS-Signals eine Genauigkeit von 2.51m erwartet werden kann.[2]

---

## **1.2 Zielsetzung**

Da das Autonome Fahren ein sehr weites, interdisziplinäres Thema ist, ist es offensichtlich, dass nicht alles in dieser Arbeit abgehandelt werden kann. Im Rahmen der DARPA Challenge wurden bereits viele Veröffentlichungen zu diesem Thema erstellt. Was im Rahmen dieser Veröffentlichungen noch nicht behandelt wurde, ist die Handhabung von Kreisverkehren, mit autonomen Fahrzeugen. Ziel dieser Arbeit ist es daher zu analysieren, welche Sensorausstattung für die Beobachtung von Kreisverkehren vonnöten ist, bzw. ob die vorhandene Sensorausstattung des ReVeRe Testfahrzeuges Snowfox als ausreichend betrachtet werden kann.

---

2. <http://velodynelidar.com/vlp-16.html> (03/09/2017)

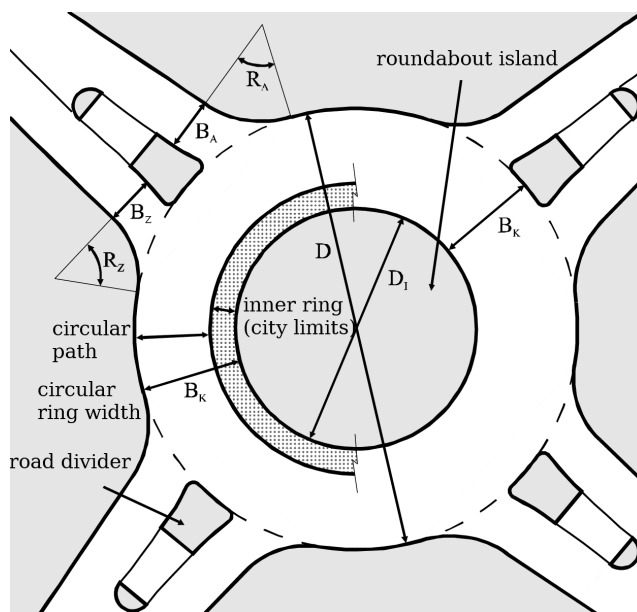
3. <http://velodynelidar.com/hdl-64e.html> (03/09/2017)

## 2.1 Roundabouts in Law

In Germany, there is no law stipulating the exact construction of roundabouts. Instead, the elements of the rural roads and city streets are dealt with in Directives for the Design of rural roads [RAL] and the Directives for the Design of Urban Roads [RASt]. These guidelines are also relevant to the choice of a convenient junction type when linking roads. The considerations discussed there are based on traffic variables, area-related characteristics, economic criteria and spatial planning or urban planning requirements. The guidelines also regulate the basic design and operational formation of roundabouts. The Directives for the Design of Urban Roads [RASt] are relevant for this dispute. Since the access the RASt ist limited, most of the information is coming from [15] whereupon RASt is based on.

### 2.1.1 Elements of a Roundabout

Figure 2.1: Definition of individual design elements and dimensions of a roundabout [15]



**Definition 2.1 (roundabout island)** *The roundabout island is the constructional area in the middle of the roundabout, which is surrounded by vehicles. For miniature roundabouts, the roundabout island is crossable. [15]*

**Definition 2.2 (circular path)** *The circular path is the road that serves to drive the roundabout island. An inner ring, if present, is not part of the circular path (VwV-StVO zu §9a V, Rn. 5). [15]*

**Definition 2.3 (circular ring with ( $B_K$ ))** *The structural width includes the circular track and a paved inner ring, if any. It is dependent on the outer diameter and the desired traffic routing (one or two lanes). The edge strip width is oriented on the relevant continuous roadway. [15]*

**Definition 2.4 (outer diameter ( $D$ ))** *The outer diameter is measured at the outer edge of the circular ring. It is the essential measure for describing the size of the roundabout. [15]*

**Definition 2.5 (inner diameter ( $D_I$ ))** *The inner diameter is the diameter of the roundabout island. [15]*

**Definition 2.6 (road divider)** *The road divider is the structurally designed island between the circular exit and circular driveway. It serves to separate the circular exit and circular driveway, the management of the traffic, as well as the pedestrians and cyclists as cross-bordering aid. [15]*

**Definition 2.7 (lane width of the circular driveway ( $B_Z$ ) and circular exit ( $B_A$ ))** *The width of the circular driveway and exit is measured at the beginning of the corner. [15]*

**Definition 2.8 (Corner rounding radius ( $R_Z$  and  $R_A$ ))** *This is the radius of the rounding at the right edge of the road between the circular driveway and the circular path. For a elliptical arch with a radius sequence of three different radii,  $R_Z$  is the radius  $R_2$  of the central arc. When the road edge is formed as a tractrix,  $R_Z$  is the smallest radius of the road edge. [15]*

### 2.1.2 Types of Roundabouts

There are several types of roundabouts, which are differentiated by the different application criteria and the partly different design principles according to the situation inside and outside built areas. Furthermore, a division is made as a function of its size. [15]

#### Mini Roundabout

Within built-up areas, smaller outer diameters are possible under certain conditions. These roundabouts are called mini roundabout. The roundabout island must then be capable of being passed over. The outer diameter should be at least 13 m, so that the circular island does not become too small. Larger outer diameters make driving easier. Outer diameters of more than 22m, however, do not offer any transport advantages. From an outside diameter of about 22 m, therefore, the installation of a small roundabout with 26 m is generally more convenient. Bypasses are generally not required in the areas where mini roundabout can be used.

Figure 2.2: Mini Roundabout  
[15]

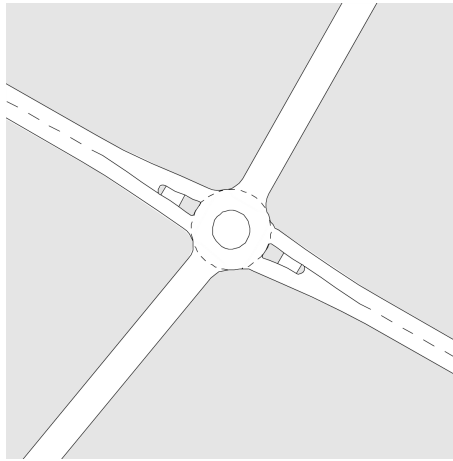


Figure 2.3: Small Roundabout  
[15]

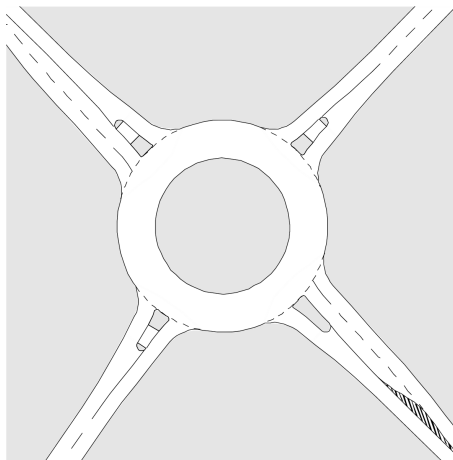


### Small Roundabout

The small roundabout has a single lane circular path and single lane circular driveways and exits. The roundabout island is not passable. The outer diameter must be at least 26 m. Bypasses can be set up for driving geometric reasons or to increase performance.

### Two-lane Passable Roundabout

Figure 2.4: Two-lane Passable  
Roundabout [15]

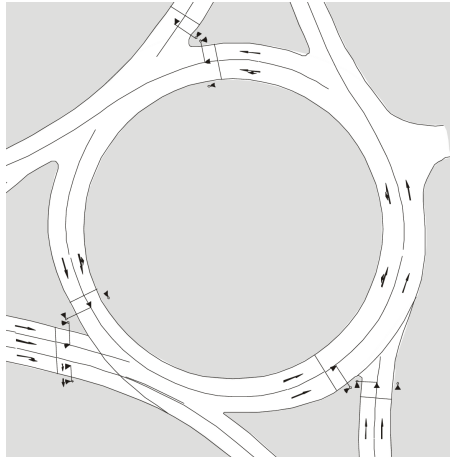


If the capacity of the small roundabout is not sufficient and can not be ensured by the installation of bypasses, the circular path of a small roundabout can be designed to be two-lane driveable. At such a roundabout, the circular path is

so wide that cars can travel side by side in a circle. If a further increase in the capacity is required, individual circular driveway can also be carried out in two lanes, if pedestrians and cyclists are not to be considered regularly. For safety reasons, circular exits are always carried out in single lanes. For geometrical reasons, the outer diameter must be at least 40 m for two-laned accessibility.

### Large Roundabout

Figure 2.5: Large Roundabout  
[15]



Large Roundabouts with two or more lanes marked by markers on the circular path should be operated with a light signaling system only, if the nodal point design and traffic control are closely coordinated.

---

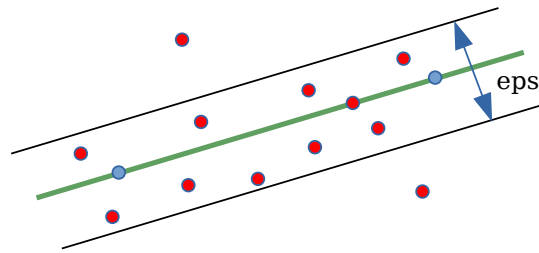
## 2.2 Random Sample Consensus

Der Random Sample Consensus (RANSAC) [4] ist ein Algorithmus zur Schätzung eines Modells innerhalb einer Reihe von Messwerten mit Ausreißern und groben Fehlern. Wegen seiner Robustheit wird er vor allem bei der Auswertung automatischer Messungen vornehmlich im Bereich der Bildverarbeitung. Hier unterstützt RANSAC durch Berechnung einer um Ausreißer bereinigten Datenmenge, des sogenannten Consensus Sets, Ausgleichsverfahren wie die Methode der kleinsten Quadrate, die bei einer größeren Anzahl von Ausreißern meist versagen.

Der RANSAC benötigt mehr Datenpunkte als für die eindeutige Bestimmung des Modells benötigt werden. Aus diesem Set von Datenpunkten werden dann zufällig so viele Datenpunkte ausgewählt, wie nötig sind, um das Modell eindeutig zu bestimmen. Aus den restlichen Daten werden dann diejenigen ausgewählt, welche einen Abstand haben, der kleiner ist als ein bestimmter Grenzwert. Diese Menge stellt nun das "Consensus Set" dar. Enthält sie eine gewisse Mindestanzahl an Werten, wurde vermutlich ein gutes Modell gefunden, und der Consensus Set wird gespeichert. Diese Schritte werden mehrmals wiederholt. Dann wird diejenige Teilmenge gewählt, welche die meisten Punkte enthält. Mit dieser Teilmenge werden mit einem der üblichen Ausgleichsverfahren die Modellparameter berechnet. Der RANSAC hat also drei zu bestimmende Parameter, die das Ergebnis beeinflussen.

- number of iterations
- minimum size of the "Consensus Set"

Figure 2.6: RANSAC [4]



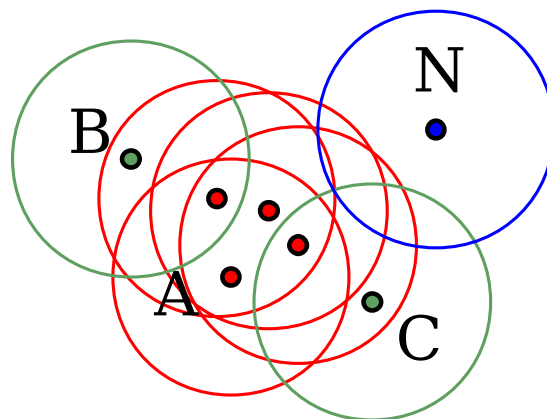
- distance threshold value (eps)

Zu beachten ist, dass der RANSAC durch die zufällige Auswahl der Datenpunkte kein deterministischer Algorithmus ist.

## 2.3 Density-Based Spatial Clustering of Applications with Noise

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [3] ist ein deterministischer Data-Mining-Algorithmus zur Clusteranalyse. Der Algorithmus basiert auf der Dichteverbundenheit, das heißt er bindet Punkte basierend auf ihrer Entfernung zu Clustern. DBSCAN iteriert über alle noch nicht bearbeiteten Datenpunkte, jeder bearbeitete Datenpunkt wird als bearbeitet markiert. Für jeden dieser Punkte wird dann eine Bereichsanfrage durchgeführt. Ist die Größe der Nachbarschaft kleiner als ein bestimmter Grenzwert, wird der Punkt als Rauschen markiert. Andernfalls wird ein neuer Cluster erstellt, indem für jeden Punkt in der Nachbarschaft eine neue Bereichsanfrage durchgeführt wird, wird dieser nicht als Rauschen klassifiziert, wird dieser zum Cluster hinzugefügt und als bearbeitet markiert. Dies wird solange ausgeführt, bis alle im Cluster befindlichen Punkte als bearbeitet markiert sind, also keine weiteren Punkte in der Nachbarschaft erreichbar sind.

Figure 2.7: DBSCAN [3]



Der DBSCAN hat also zwei zu bestimmende Parameter, welche das Ergebnis beeinflussen.

- maximum distance of the neighborhood points
- minimum number of points required to form a dense region

---

## 2.4 Middleware OpenDAVINCI

Autonome Software ist typischerweise ein verteiltes System, auf heutigen Fahrzeugen basiert dieses System auf ECUs und Bussystemen wie CAN, LIN. Verteilte Software vereinfacht es komplexe Komponenten innerhalb des Systems zu integrieren. Im Bereich des Autonomen Fahrens ist der historische Aufbau von Fahrzeugen mit ECU's und CAN jedoch nicht optimal. Um die vielen benötigten Komponenten zu handhaben, ist es von Vorteil, Komponenten auch innerhalb eines ECU's bzw. einer Recheneinheit zu entkoppeln. Für diesen Zweck gibt es bereits mehrere Middleware, die unter anderem die Kommunikation innerhalb der Komponenten handhaben und abstrahieren. Im Rahmen des Copplar Projekts, wird hier die OpenDaVINCI Middleware genutzt. OpenDaVINCI ist eine echtzeitfähige Laufzeitumgebung konzipiert für Autonome Fahrzeuge. OpenDaVINCI basiert auf Hesperia [1]. Die Kommunikation zwischen den Komponenten basiert in OpenDaVINCI auf UDP Multicast, welches eine echtzeitfähige Kommunikation zwischen den Komponenten ermöglicht [8]. Für die Kommunikation bietet OpenDaVINCI Time-triggered sender und Data-triggered receiver an, von welchem in folgenden der Data-triggered receiver für die Anbindung der Software genutzt wird. Weiterhin bietet OpenDaVINCI viele weitere Funktionalitäten, die das Handling von World Geodetic System 1984 (WGS84) Koordinaten an, welches für die Umwandlung von GPS Koordinaten in lokale kartesische genutzt werden kann. Dazu ist die Angabe einer Referenz GPS Position nötig, welche um den Berechnungsfehler klein zu halten, nicht zu weit entfernt sein sollte.

---

# 3

## State of the Art



Der eigentliche Kern der Arbeit. In diesen Kapiteln wird die Vorgehensweise methodisch erklärt. Oft werden für Teilprobleme separate Kapitel angefertigt. Oft sind diese "Methodik-Kapitel aber auch logisch getrennt. Beispielsweise Kapitel für "Problemanalyse, Verfahrensauswahl, Umsetzung und Implementierung. Hinweis: Denken Sie daran: Diese Kapitel sind der Kern der Arbeit!! Sie sollten hier insbesondere klar herausstellen, was vorhandene Vorarbeiten sind und was im Rahmen dieser Arbeit entwickelt und umgesetzt wurde, also was Eigenanteil und was Fremdanteil ist

siehe jens BA, das hier ist zu kurz

kreisverkehre überarbeiten

beleg

Im vorigen Kapitel haben wir uns die Arten von Kreisverkehren und deren Komponenten angesehen. Weiterhin haben wir die zur Verfügung stehende Testplattform und ihre Sensorik begutachtet. Dabei haben wir festgestellt, dass für die Erkennung von Objekten in anderen Arbeiten häufig mehrere und teurere Sensoren kombiniert werden um ein Zuverlässiges erkennen von anderen Verkehrsteilnehmern zu gewährleisten. In der Research Question eins haben wir festgehalten, dass wir die Verwendung eines günstigen VLP-16 Sensors in einem komplexen Verkehrsszenario, die Verkehrsbeobachtung eines Kreisverkehrs evaluieren wollen.

Dazu wird im folgenden ein Algorithmus zum erkennen und Tracken von Objekten mit Hilfe des Velodyne VLP-16 vorgeschlagen und implementiert. Die Schwierigkeit besteht dabei in der Verwendung eines einzigen und im Vergleich günstigen Umfellsensors, welcher offensichtlich nicht als standalone Lösung für diesen Einsatzzweck entwickelt wurde.

Dieser Sensor bietet in seiner aktuellen Anwendung in dem für diese Arbeit relevanten Bereich eine vergleichsweise geringe Auflösung. Daher schlagen viele in ähnlichen Projekten genutzte Gradienten basierende Algorithmen im Bereich der Segmentierung häufig fehl. Aus diesem Grund wird für die Segmentierung eine Groundplane basierender Algorithmus implementiert.

beleg

Außerdem ist es Bauartbedingt in Kreisverkehren mit bebauten Mittelinseln und Mehrspurigen Kreisverkehren nötig Fahrzeuge über ihren Messhorizont hinaus zu verfolgen, um ein sicheres Einfahren in den Kreisverkehr zu gewährleisten. Zu diesem Zweck wird in Section ein Tracking und State Estimation Algorithmus entwickelt welches dies gewährleisten soll.

section reference

Zur Evaluation dieser Algorithmen wurden mehrere Datensammlungen auf den nahe gelegenen schwedischen AstaZero fig. 3.1 Prüfgelände in Sandhult durchgeführt, für alle nicht dort durchgeführten Experimente werden in einer dafür erstellten Simulation durchgeführt. In dieser Simulation wird ein Innerstädtischer Kreiverkehr mit Fuß und Radweg nachgebaut, welche der Kreisverkehr auf AstaZero nicht bieten kann.

Die Evaluierung findet dabei von Hand anhand der grafisch aufbereiteten Messdaten statt. Dabei wird besonders auf False-Negativ und False Positiv erkannte Hindernisse eingegangen. Grobe Außreißer bei der Position oder Orientierung der Objekte werden ebenfalls vermerkt.

Zur Evaluation der Handbarkeit des Kreisverkehrs wird außerdem eine Statemachine Implementiert welche das Fahrzeug sicher und Unfallfrei durch den Kreisverkehr bewegen soll. Dazu wird die Simulation über einen längeren Zeitraum beobachtet, und die Anzahl der eventuellen Kollisionen notiert.

Figure 4.1: AstaZero Proving Ground

[http://www.astazero.com/wp-content/uploads/2016/09/%C3%96versiktsskiss\\_mod.pdf](http://www.astazero.com/wp-content/uploads/2016/09/%C3%96versiktsskiss_mod.pdf)



---

## 5.1 Objekt Detection

### 5.1.1 Ground Removal

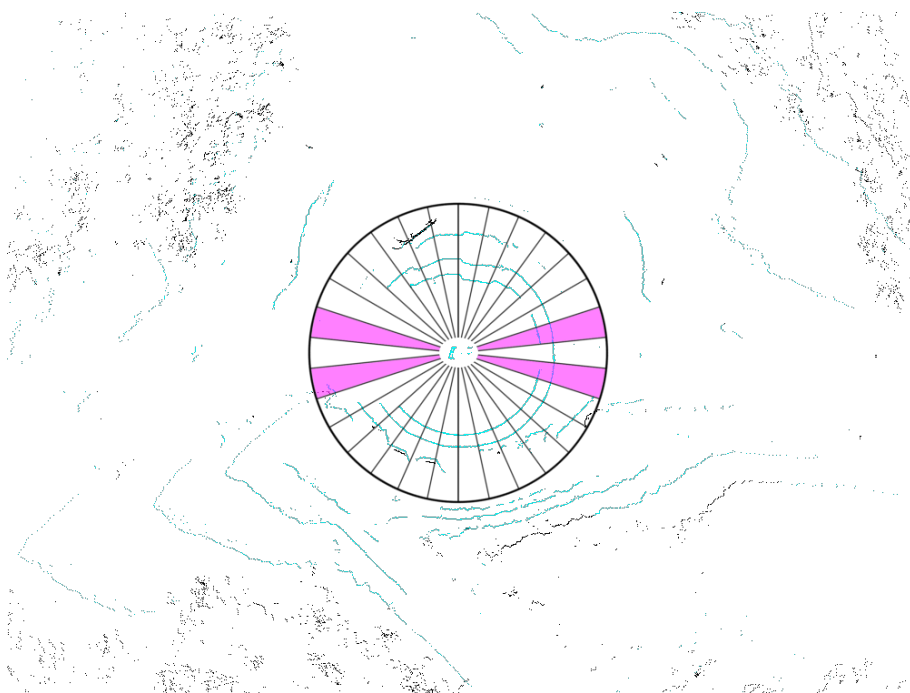
Um in einer PointCloud Objekte zu erkennen ist es nötig, zu wissen, welche Messungen zu Boden und welche zu Objekten gehören. Es gibt viele Möglichkeiten dieses zu erreichen. Die Naivste Methode ist das Entfernen der Bodenplatte anhand ihrer Z-Koordinate. Diese Methode hat allerdings viele Nachteile, zum einen muss der LIDAR Sensor exakt gerade auf ein Fahrzeug angebracht werden, zum anderen muss das Fahrzeug ein sehr steifes Fahrwerk haben, um eventuelle Neigungen des Sensors zu verhindern. Weiterhin erlaubt dies ausschließlich die Entfernung von planaren Grundflächen, also flache nicht hügelige Untergründe. Eine weitere verbreitete Methode ist das Entfernen der Bodenplatte auf Basis eines statistischen Mittelwertes [17]. Diese Methode benötigt allerdings auch eine Kalibrierung der Sensorabstände zum Boden. Und die Bestimmung weiterer Schwellwerte, welche umgebungsabhängig sind. Die Vorteile beider Methoden sind ihre geringe Rechenleistung und Laufzeit  $O(n)$ . Bessere Methoden wie Gradientenbasierende Algorithmen benötigen einen Startpunkt der als Bodenplatte identifiziert werden kann. Eine weitere Möglichkeit ist die Beschreibung von Objekten als konvexe Objekte [10], die ebenfalls auf Basis der Gradienten beschrieben werden kann. Vorteil dieser Methode ist, dass keine initiale Position für die Bodenplatte benötigt wird.

Für unseren Anwendungsfall mit dem Velodyne VLP-16 besteht das Problem darin, dass die Auflösung des Sensors in der Höhe sehr gering ist. Abhängig von der Entfernung des Fahrzeuges innerhalb der benötigten Reichweite fallen nur zwei Lagen auf die Testfahrzeuge, weshalb Gradientenbasierende Methoden hier zuverlässig versagen. Da die Gradienten zu klein sind und die Verkleinerung der nötigen Thresholds zu häufigen falschen Positiven führt. Die Methode des statistischen Mittelwertes und die Methode auf Basis der Z-Koordinate, leiden am Fahrwerk des Volvo XC90 SUV. Die Höhe des Fahrzeuges ändert sich unter anderem durch Veränderung des Fahrprofils (Sport/Eco, etc.) um mehrere Zentimeter. Auch leicht erhöhte Geschwindigkeiten im Kreisverkehr (ca 30 km/h) führen zu einer deutlichen Seitenneigung des Fahrzeuges. Darum

wird nun eine weitere Methode vorgeschlagen. Die Erkennung einer Grundfläche in den Messdaten.

Für die Erkennung des Bodens gehen wir von folgenden Annahmen aus, die Straße lässt sich approximativ als Ebene im R3 darstellen. Weiterin ist die Grundfläche die niedrigste Fläche im gesuchten Bereich. Daher wird im ersten Schritt der in Polarkoordinaten vorliegende Datensatz in 30 Tortenstückförmige Segmente geteilt. Aus diesem Tortenstück werden dann jeweils vorne und hinten zwei Segmente [fig. 4.1] ausgewählt, welche nicht beachtbar sind. Die Auswahl der Segmente folgt aus der Annahme, dass sich die Straße vor, bzw. hinter dem Fahrzeug befindet. Zukünftig könnte die Auswahl der Segmente auch mit Hilfe des Fahrzeuglenkwinkes optimiert werden. Oder der gültige Bereich von einer Lane Detection geliefert werden.

Figure 5.1: LiDAR Segments



beschreiben warum hinten messwerte fehlen,  
in TestPlatform Kapitel

Innerhalb dieser Tortenstücke wird dann eine Suche nach den 10 Messungen mit dem niedrigsten Z Wert gesucht. Die Suche beschränkt sich dabei auf die drei niedrigsten Lagen (-15,-13,-11 Grad), da alle höheren Lagen zu weit weg wären. Die Einteilung in Segmente ist deshalb nötig um zu verhindern, dass alle Messerte in ein einziges lokales Minima laufen. Aus diesem Vorgefilterten Messwerten werden nun für einen RANSAC drei zufällig herausgesucht. Aus diesen drei Punkten wird nun eine Ebene in der Hesseschen Normalform gebildet, was eine effiziente Distanzberechnung zu anderen Punkten erlaubt. Danach sammeln wir alle weiten Punkte aus unseren Minima, anhand eines Distanzkriteriums. Danach wird aus der Ebene und den neu gesammelten Punkten durch einen Planefitting Algorithmus [section 4.1.1] eine neue Ebene und deren Fehler berechnet. Der Fehler wird über die Summe der quadratischen Abstände aller Punkte zur Ebene berechnet.

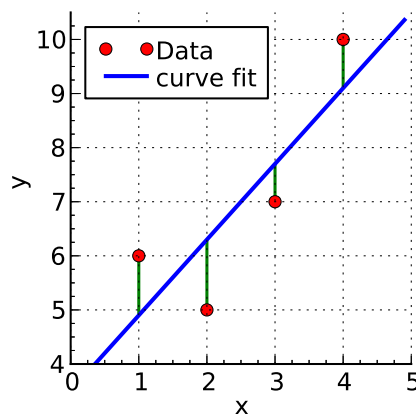
Bevor wir die Ebene jedoch als eventueller Lösungskandidat hinzufügen wird geprüft ob sich die Ebene innerhalb von einem plausiblen Parameterbereich befindet. Dazu zählt, dass die Entfernung der Ebene zwischen 1.9m und 2.2m bewegen sollte, dies entspricht in etwa der Montagehöhe des Velodyne Sensors.

Die Anzahl an Iterationen des RANSAC ist auf 50 Begrenzt. Nach dem Durchlauf des RANSAC werden alle Punkte in der Pointcloud anhand ihrer Distanz zur Ebene als Groundfläche makiert. Als threshold wurde hier experimtel ein optimaler Wert von 0.5m ermittelt.

### Planefitting

Zum Planefitting einer be ne wird üblicherweise eine Singular Value Decomposition (SVD) [11, 12, 14]. SVD hat eine Komplexität von  $\mathcal{O}(\min\{mn^2, m^2n\})$  [6], da das Planefitting innerhalb des RANSAC sehr häufig mit einer großen Anzahl an Punkten ausgeführt wird, führt das Ausführen des SVD innerhalb des RANSAC zu einer sehr hohen laufzeit. Deshalb wird an dieser Stelle ein Linear least Squares (LLSQ) Algorithmus mit einigen optimierungen eingesetzt. Bei der verwendung des LLSQ gilt es zu beachten, dass nicht der abstand der Punkte zur eben optimiert wird, sondern der Abstand der Punkte zur Ebene entlang einer Achse (in unserem Fall der z Achse) siehe fig. 4.2. Das kann zu Problemen führen, wenn die Punkte weit gestreut, also weit von der Optimalen Ebene entfernt sind. Da wir unsere Punkte innerhalb des RANSAC allerdings anhand eines Distanzkriteriums vorselektieren, stellt dies kein Problem dar.

Figure 5.2: Linear least Squares (LLSQ) [16]



Die Darstellung einer Ebene in Koordinatenform sieht wie folgt aus:  $a\vec{x} + b\vec{y} + c\vec{z} + d = 0$ . Da wir eine Ebene im R3 betrachten, ist dieses Gleichungssystem überbestimmt. Da wir unsere Ebene in Richtung der Z-Achse optimieren wollen setzten wir Parameter c auf 1 und können unser Gleichungssystem nun einfach nach z auflösen:  $a\vec{x} + b\vec{y} + d = -\vec{z}$ . Die Vektoren  $\vec{x}, \vec{y}, \vec{z}$  stellen dabei die zu fittenden Punkte dar. In Matrixschreibweise:

$$X\vec{\beta} = \vec{z}$$

$$\begin{bmatrix} x_0 & y_0 & 1 \\ x_1 & y_1 & 1 \\ \dots & \dots & \dots \\ x_n & y_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ d \end{bmatrix} = \begin{bmatrix} -z_0 \\ -z_1 \\ \dots \\ -z_n \end{bmatrix}$$

Dieses Sytem hat üblicherweise keine Lösung, unser eigentliches Ziel ist jeoch auch nicht extakte lösungen für  $\vec{\beta}$  zu finden sondern eine gute näherung  $\hat{\beta}$  dafür:

$$\hat{\beta} = \min (||\vec{z} - X\vec{\beta}||^2)$$

Das können wir tun indem wir unsere Gleichung mit der Transponierten unserer Punktmatrix  $X$  multiplizieren:

$$(X^T X) \hat{\beta} = X^T \vec{z}$$

$$\begin{bmatrix} x_0 & x_1 & \dots & x_n \\ y_0 & y_1 & \dots & y_n \\ 1 & 1 & \dots & 1 \end{bmatrix} \begin{bmatrix} x_0 & y_0 & 1 \\ x_1 & y_1 & 1 \\ \dots & \dots & \dots \\ x_n & y_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ d \end{bmatrix} = \begin{bmatrix} x_0 & x_1 & \dots & x_n \\ y_0 & y_1 & \dots & y_n \\ 1 & 1 & \dots & 1 \end{bmatrix} \begin{bmatrix} -z_0 \\ -z_1 \\ \dots \\ -z_n \end{bmatrix}$$

Dieses Gleichungssystem könne man nun mit der Berechnung der Inverse von  $(X^T X)$  auflösen. Da die Berechnung von Inversematritzen mit  $\mathcal{O}(n^3)$  ebenfalls aufwändig ist, nun ein weiterer Trick um rechenleistung zu sparen. Nach dem Multiplizieren der Transponierten erhalten wir:

$$\begin{bmatrix} \sum x_i x_i & \sum x_i y_i & \sum x_i \\ \sum y_i x_i & \sum y_i y_i & \sum y_i \\ \sum x_i & \sum y_i & N \end{bmatrix} \begin{bmatrix} a \\ b \\ d \end{bmatrix} = \begin{bmatrix} \sum x_i z_i \\ \sum y_i z_i \\ \sum z_i \end{bmatrix}$$

Gut zu sehen sind hier die Summen in den Randbereichen der Matrix  $X$  und dem Vektor  $\vec{z}$ . Diese können wir auf Null setzten, wenn wir alle Punkte relativ zum Mittelwert-Punkt aller Punkte definieren, also  $P_i = P_i - \bar{P}$ . Nun erhalten wir:

$$\begin{bmatrix} \sum x_i x_i & \sum x_i y_i & 0 \\ \sum y_i x_i & \sum y_i y_i & 0 \\ 0 & 0 & N \end{bmatrix} \begin{bmatrix} a \\ b \\ d \end{bmatrix} = \begin{bmatrix} \sum x_i z_i \\ \sum y_i z_i \\ 0 \end{bmatrix}$$

Nun können wir  $d$  ebenfalls auf Null setzten, denn wenn alle unsere Punkte relativ zum Mittelwert-Punkt sind, dann läuft auch unsere Ebene immer durch diesen Punkt. Daher können wir nun eine komplette Dimension streichen:

$$\begin{bmatrix} \sum x_i x_i & \sum x_i y_i \\ \sum y_i x_i & \sum y_i y_i \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum x_i z_i \\ \sum y_i z_i \end{bmatrix}$$

Das Gleichungssystem können wir nun einfach mit der Cramer's rule lösen

$$D = \sum x_i x_i \cdot \sum y_i y_i - \sum x_i y_i \cdot \sum x_i y_i$$

$$a = \frac{\sum y_i z_i \cdot \sum x_i y_i - \sum x_i z_i \cdot \sum y_i y_i}{D}$$

$$b = \frac{\sum x_i y_i \cdot \sum x_i z_i - \sum x_i x_i \cdot \sum y_i z_i}{D}$$

$$\vec{n} = [a, b, 1]^T$$

Dabei gibt es zu beachten, dass die Determinante nicht Null oder nahe Null sein darf. Da der Winkel zwischen dem Fahrzeug und der Ebene jedoch immer nahe 90 Grad liegt, ist die Determinante typischerweise sehr groß. Sollte die Determinante doch nahe 0 (nicht gleich 0) sein, wird die Berechnung trotzdem durchgeführt, da dies auch zu einem großen Fehler im Fitting führt. Dies ist an dieser Stelle erwünscht, da der RANSAC ungeültige Ebenen anhand des Fehlers aussortiert. Ist die Determinante exakt Null, wird die Berechnung stattdessen mit einem kleinen Wert für  $D$  fortgesetzt.

Aus dem Normalenvektor  $\vec{n}$  und dem Mittelwert-Punkt  $\bar{P}$  können wir nun wieder die Hessische Normalenvektor bestimmen.

Letztendlich haben wir so den Algorithmus von  $\mathcal{O}(m^2n)$  auf  $\mathcal{O}(n)$  runterbrechen können.

### 5.1.2 Clustering

In aktuellen Abreiten mit 3D-LIDAR Daten werden die Daten häufig als erstes in eine Heightmap projiziert [17, 5, 9]. Danach werden direkt benachbarte Messungen mit ähnlichen Messwerten zusammengefasst. Alternativ werden die Messungen auch anhand eines Distanzkriteriums zusammengefasst. Erstere Methode hat den Nachteil, dass einzelne Ausreißer dazu führen, dass das Objekt in mehrere Cluster zerfällt. Letztere wird meist mit einem KD-Tree oder einer ähnlichen Datenstruktur kombiniert, welche typischerweise hohe Kosten für die Erstellung verursachen. Da der Baum nach jeder 360 Grad messung neu aufgebaut werden muss ist das Problematisch.

Hier wird eine Methode vorgeschlagen, welche die Vorteile beider Methoden kombiniert. Dazu ist es nötig zu wissen, wie die Daten von der OpenDAVINCI Middleware geliefert werden. Das OpenDAVINCI auf der Übertragung der Daten mit UDP Multicast setzt, werden die Daten in einer kompakten Form übertragen, welche in einen einzigen UDP Frame passt.

CompactPointCloud
startAzimuth : float endAzimuth : float entriesPerAzimuth : uint32 distances : byte[]
getStartAzimuth : float ...

Dabei wird von einer konstanten Drehrate des Sensors ausgegangen, was in einer äquidistanten der Messwerte resultiert. Die Anzahl der Messungen pro Azimuth wird in entriesPerAzimuth festgehalten und entspricht für den Velodyne VLP16 16. Um nun an die eigentlichen Messwerte zu kommen müssen jeweils zwei distance Werte zu einem Unsigned 16Bit Integer umgewandelt werden, welcher dann die Messung in cm enthält. Jeweils 16 dieser Werte ergeben dann einen Messframe in dem der Polarwinkel auf einen Bereich zwischen -15 und +15 abgebildet werden muss. Nachdem die sphärische Daten wiederhergestellt wurden, werden diese in Kartesische umgewandelt und in eine Punkt Datenstruktur gespeichert.

Point
azimuth : float measurement : float visited : bool isGround : bool point : vector3f
getAzimuth : float ...

Diese wird wiederum in ein Statisches 2 Dimensionales Array Gespeichert: Points[2000][16]. Die Reihenfolge der Daten wird dabei beibehalten. Diese Datenstruktur stellt nun im weiteren Verlauf unsere Pointcloud dar.

warum DBSCAN mit Noise!

Auf dieser Basis wird nun ein DBSCAN [3] ausgeführt. Der DBSCAN Algorithmus hat dabei folgende Vorteile. Im Gegensatz beispielsweise zum K-Means-Algorithmus, muss nicht im vornherein bekannt sein, wie viele Cluster existieren. Der Algorithmus kann Cluster beliebiger Form (z.B. nicht nur kugelförmige) erkennen. Das macht den DBSCAN damit für uns zu optimalen Kandidaten. DBSCAN selbst ist von linearer Komplexität. Die meiste Rechenzeit wird jedoch überlicherweise durch die Nachbarschaftsberechnung verursacht. Genau hier setzen wir an, anstatt der Bereichsanfrage über eine Baumstruktur nutzen wir aus, dass Messungen in einer kleinen Nachbarschaft einen ähnlichen Azimuth Winkel haben. Dazu untersuchen wir für jeden Messwert jeweils zwei weitere Einträge nach links und rechts in unserem Array. Effektiv müssen wir daher  $5 \cdot 16 = 80$  Werte überprüfen. Die Laufzeit der Bereichsanfrage kann deshalb in linearer Komplexität durchgeführt werden. Alle Messwerte die zuvor als Grund klassifiziert wurden, werden bei der Berechnung übersprungen, zusätzlich entfällt der Aufbau eines KD-Trees.

### 5.1.3 Tracking

datenstruktur Objekte hinzufügen, weil positionen und attribute verwirrend

Das Tracking ist in zwei Abschnitte unterteilt. Dem Tracking der Cluster vom DBSCAN und dem Erstellen und Tracken von Hindernissen.

#### Cluster Tracking

Für das Tracking der Cluster nehmen wir an, dass sich Objekte von Zeitschritt zu Zeitschritt nur geringfügig bewegen, weiterhin ändert sich die Form der Cluster ebenfalls nur leicht. Das ist wichtig, da die Position eines Clusters durch einen Mittelwertpunkt definiert ist. Das Tracking wird im R2 durchgeführt. Im Initialen Schritt wird jedem Cluster eine aufsteigende ID zugeordnet. In jedem weiteren Schritt wird jedem neuen Cluster die ID des alten Clusters zugeordnet welcher über die Zeit hinweg die geringste Entfernung aufweist. Für diese Entfernung gibt es eine großzügige obere Schranke von 3m, Cluster die nicht innerhalb dieser Grenze sind erhalten eine neue ID. Das führt dazu, dass mehreren Cluster die selbe ID zugeordnet werden kann, das ist wichtig, da Objekte manchmal in mehrere Cluster zerfallen.

#### Object Tracking

Basis für das Objekt Tracking sind die zuvor getrackten Cluster. Im Initialen Schritt werden aus allen Cluster mit der selben IDs Objekte gebildet. In jedem weiteren Schritt werden alle Cluster mit der zuvor gleichen ID zum Update der Objekte genutzt. Aus Clustern mit neuen IDs werden neue Objekte gebildet.

Der vorerst wichtigste Schritt beim Updatevorgang ist die Berechnung der Bewegungsrichtung eines Objektes, da folgende Berechnungen auf dieser basieren. Bei der Berechnung der Bewegungsrichtung ist zu beachten, dass die Bewegung des eigenen Fahrzeuges herausgerechnet werden muss. Dazu werden die Positionsdaten des Applix POS-LV genutzt. Da sowohl die Positionsdaten des Applix Systems, als auch die erkannte Position des Fahrzeuges fehlerbehaftet sind, wird die Bewegungsrichtung nur bei einer minimalen Bewegung von 2m geupdated.



$$\begin{aligned}\Delta x &= P_x(t) - P_x(t-2m) + \Delta C_x \\ \Delta y &= P_y(t) - P_y(t-2m) + \Delta C_y \\ \theta &= \text{atan2}(\Delta y, \Delta x)\end{aligned}$$

mit  $P$  - Position des Objekts,  $C$  - Position des eigenen Fahrzeuges

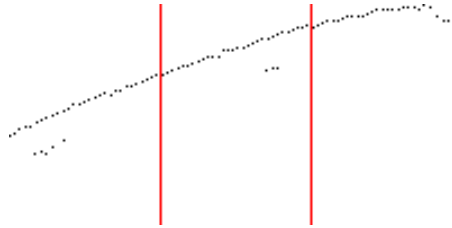
Das Ergebnis kann in fig. 4.3 betrachtet werden. Gut zu erkennen ist, dass die Bewegungsrichtung (Pfeil) nicht mit der Ausrichtung des Objekts (schwarz) übereinstimmt, die Boundingbox jedoch korrekt ausgerichtet ist. Wie diese Berechnung zustande kommt wird im folgenden geklärt.

Figure 5.3: Obstacle Movement



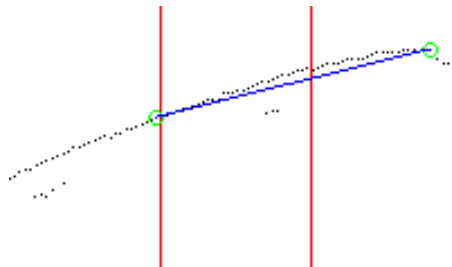
Basierend auf der Bewegungsrichtung wird nun die Ausrichtung des Fahrzeuges berechnet. Dazu werden alle dem Objekt zugewiesenen Cluster zusammengefasst und um  $-\theta$  gedreht. Danach wird das Objekt in 3 gleich große Segmente unterteilt (fig. 4.4).

Figure 5.4: Obstacle Cutting



Weiterhin wird bestimmt ob sich das Objekt oberhalb oder unterhalb der  $x$ -Achse befindet. Dies ist wichtig, da wir wissen müssen, welche Seite des Objekts wir messen. Befindet sich das Objekt also unterhalb der  $x$ -Achse wird im nächsten Schritt der  $y$ -Wert maximiert, befindet es sich oberhalb, wird er minimiert. Im folgenden gehen wir davon aus, dass sich das Objekt unter der  $x$ -Achse befindet. Deshalb maximieren wir nun im linken und rechten Segment des Geteilten Hindernises die  $y$ -Werte. Die Unterteilung in 3 Segmente ist nötig um zu verhindern, dass bei perfekt wagerechte ausgerichteten Objekt beide Maxima in den selben Punkt laufen. Mit diesen Punkten  $(\vec{R}; \vec{L})$  wird nun eine Korrektur der Drehung des Objektes berechnet:

Figure 5.5:  $\theta$  - Correction



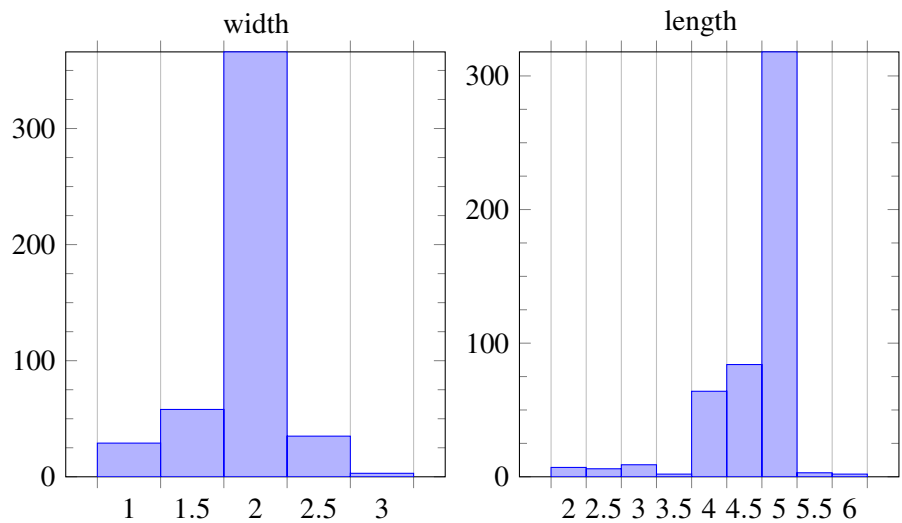
$$\Delta x = R_x - L_x$$

$$\Delta y = R_y - L_y$$

$$\theta_{correction} = \text{atan2}(\Delta y, \Delta x)$$

Nach Anwendung der Korrektur wird die Größe des Hindernisses berechnet. Dazu werden die maximalen und minimalen x und y Werte herangezogen. Mit diesen Werten wird nun über die Zeit ein auf 0.5m gerundetes Histogramm für die Länge und Breite des Hindernisses aufgebaut. Anhand diesem wird dann der wahrscheinlichste Wert ausgewählt. Dadurch ändert sich die Größe des Hindernisses zu begin häufiger, bevor die Größe auf einen stabilen wert konvergiert. Da die größe des Objekts zu begin sehr klein sein kann, gibt es für beide Werte einen unteren Grenzwert. Messerte für ein Beispielobjekt sind in fig. 4.6 zu sehen.

Figure 5.6: Object Size Histogram



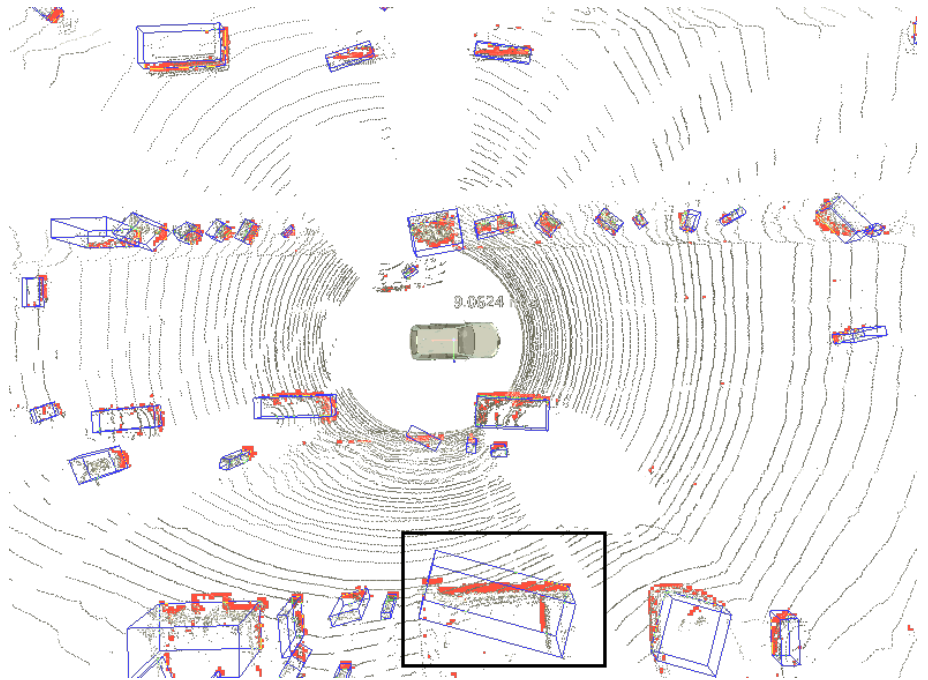
Leicht zu sehen wird für das Object eine breite von 2m und eine Länge von 5m berechnet. Das Objekt ist in diesem Fall ein Volvo S60, welcher Außenmaße von ca. 1.9m und 4.6m hat, womit die Abweichungen sich korrekt innerhalb der Rundung der Werte befinden. Für die nachfolgende Filterung der Messwerte mit Hilfe eines Kalman Filters, wird nun die Position des Fahrzeuges aus dem Mittelpunkt der Boundingbox bestimmt.

Die für die Berechnung der Bewegungsrichtung genutzte Position ist jedoch eine andere, da die so eben berechnete Position zu diesem Zeitpunkt noch nicht zur Verfügung steht und die Position kurz nach der Initialen Erkennung durch die häufigen Größenänderungen sehr instabil ist. Deswegen wird als Position immer die maximale x-Koordinate der Cluster genutzt. Diese Position kann ebenfalls in fig. 4.3, als grüner Punkt begutachtet werden. Da  $\theta$  im initialen Zeitschritt Null ist, entspricht dies der globalen Maximalen x-Koordinate des Clusters. Dies führt dazu, dass wir im initialen Zeitschritt annehmen, dass sich das Objekt in die positive x-Richtung bewegt. Bei Objekten bei denen das nicht der Fall ist, führt das zu einem kurzzeitigen oszillieren der Orientierung, welche sich jedoch über die Zeit schnell stabilisiert.

Der eine oder andere möge sich wundern warum die Boundingbox so aufwendig berechnet wurde. Eine einfache Art und Weise eine Boundingbox für die Objekte zu berechnen wäre die Berechnung der Minimum Boundingbox über die

konvexe Hülle, so wie es in vielen Anderen Arbeiten gemacht wird [17, 5]. Die Minimum Bounding box liefert jedoch unter Umständen nicht das gewünschte Ergebnis, zum einen hat sie immer nur die Größe der aktuellen Messung und zum anderen kann sie eine falsche Orientierung liefern, wie in fig. 4.7 zu sehen.

Figure 5.7: Error with minimum Boundingbox [5]



Da die Orientierung und Position der Objekte jedoch als Eingang für den nachfolgenden Kalmanfilter genutzt wird, welcher sehr sensibel auf falsche Orientierungen reagiert, wurde eben jeder Algorithmus entwickelt.

### Object Confidence

Implementierung Confidence Wert auf 1 setzen und entfernen, wenn Wert unter 1, oder Objekte auf Basis ihrer Position entfernen

Um zu verhindern, dass kurzzeitig erkannte False Positives direkt als Objekt erkannt werden und damit die nachfolgende Logik beeinflusst, wird nun ein Confidence Wert eingeführt. Bevor ein erkanntes Objekt als gültig erachtet wird, muss dieses einen gewissen Confidence Wert erreichen. Der initiale Confidence Wert eines Objektes ist Null. Der Confidence Wert wird um eins erhöht, wenn das Objekt in zwei aufeinander folgenden Zeitschritten getrackt werden kann und folgende Bedingungen erfüllt:

- Die Breite des Objekts muss kleiner sein als die Länge des Hindernisses zuzüglich 1.5m
- Die Länge des Hindernisses muss kleiner sein als 10m
- Die Breite des Hindernisses muss kleiner sein als 4m

Ist eine dieser Bedingungen nicht erfüllt, wird der Confidence Wert stattdessen halbiert. Damit ein Objekt als gültig erachtet wird, muss es einen Confidence Wert von 3 erreichen, sodass ein Objekt erst nach mindestens drei Iterationen erkannt werden kann.

#### 5.1.4 Classification

filterung in code entfernen, weil größe bereits  
geteilt wird  
implementierung Plausibilitätstest

Die Klassifizierung der Objekte findet anhand ihrer Größe statt, es findet keine Klassifizierung nach bewegten und unbewegten Objekten statt. Unterschieden werden lediglich Fußgänger, Radfahrer, und Fahrzeuge, und Sonstige. Als Klassifizierungskriterium wird die Größe der Objekte genutzt. Die Klassifizierung sieht wie folgt aus:

**pedestrian:** length < 1.5 and width < 1.5

**cyclist:** length < 2 and width < 1.5

**car:** length < 10 and width < 4

**undefined:** length >= 10 and width >= 4

Weiterhin wird anhand der Geschwindigkeit eine Plausibilitätstest durchgeführt. So darf ein Fußgänger eine Geschwindigkeit von 10km/h nicht überschreiten und die Maximalgeschwindigkeit für einen Radfahrer beträgt 30km/h. Da für Fahrzeuge keine sinnvolle Geschwindigkeitsgrenze angenommen werden kann, wird an ihrer Stelle die Änderung der Orientierung genutzt. Als Maximale Drehrate wird eine Messung von 0.3 rads/sec aus [7] angenommen. Da der Wert eine Obere Grenze darstellen soll, nehmen wir einen etwas höheren Wert von 0.3 rads/sec an.

#### 5.1.5 State Estimation

Nachfolgende des Trackings wird auf den Erkannten Objekten eine Zustandsschätzung durchgeführt. Dies ist nötig, da Objekte während der Fahrt verdrängt werden können, sei es durch andere bewegte Objekte oder Gebäude. Eine Schätzung des Zustandes über den Erkennungshorizont hinaus erlaubt es uns eine Aussage über die Position von Objekten zu treffen, welche im Moment nicht sichtbar sind. Weiterin erlaubt es auf einfache Weise zeitweise nicht erkannte Objekte wiederzuerkennen, also dem Objekt die gleiche ID zuzuweisen wie zuvor.

Aus dem zuvor durchgeführten Tracking können wir die Aktuelle Position, Geschwindigkeit, Drehung und Drehgeschwindigkeit erhalten. Für eine Zustandsschätzung mit den für Fahrzeuge üblichen Bicycle Model fehlen Angaben über den Radstand und Gewicht des Fahrzeuges. Daher müssen wir uns auf ein relativ einfaches "Constant Turn Rate and Velocity" Model beschränken. Dies erlaubt es uns allerdings das gleiche Model für alle Klassen von Objekten zu nutzen. Da dieses Model ebenfalls auf Fußgänger und Radfahrer angewendet werden kann.

##### Constant Turn Rate and Velocity Model

Der Zustandvektor [13] des CTRV- Modells sieht wie folgt aus:

$$\vec{x}(t) = [x \ y \ \theta \ v \ \omega]^T$$

$x$  - Y Axis

$y$  - X Axis

$\theta$  - Object Yaw Angle

$v$  - Object Velocity

$\omega$  - Yaw Rate

Die Dynamikmatrix erhalten wir durch eine nichtlineare Zustandsübergang:

$$\vec{x}(t+T) = \begin{bmatrix} \frac{v}{\omega}(-\sin(\theta)) + \sin(T\omega + \theta) + x(t) \\ \frac{v}{\omega}(\cos(\theta)) - \cos(T\omega + \theta) + y(t) \\ \omega T + \theta \\ v \\ \omega \end{bmatrix}$$

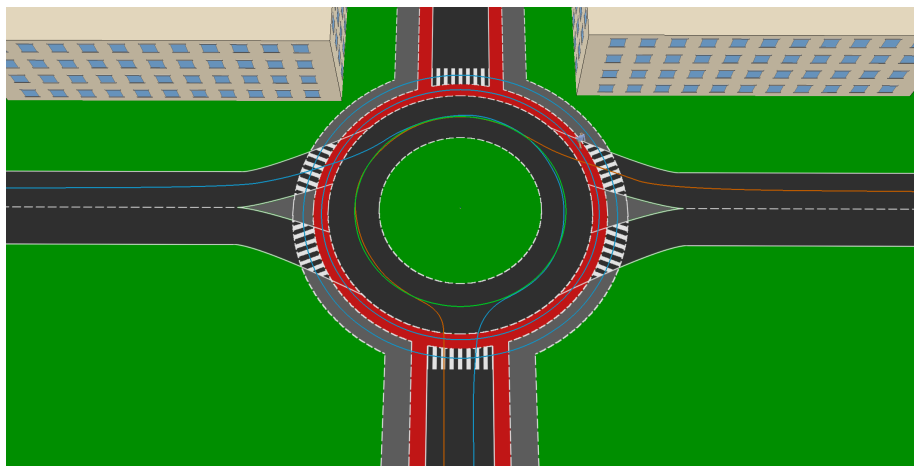
### Prediction

Befindet sich ein Objekt innerhalb der Reichweite des Velodyne Sensors und wird im darauffolgenden Zeitschritt nicht erkannt, wird der Prediktionschritt des Kalman filter weiterhin ausgeführt. Dies geschieht solange die Unsicherheit der Position einen gewissen Schwellwert überschreitet. Sobald das Clustertracking dann ein neues Objekt detektiert, dem keine bisher bekannte ID zugewiesen werden kann, wird die Position mit allen Objekten in der Prediktionsphase abgeglichen. Befindet sich das neue Objekt nahe an der predizierten Position wird der Cluster dem Objekt zugewiesen, und der Korrekturschritt des EKF durchgeführt.

## 5.2 Simulation

Die Simulationsumgebung, welche verwendet wird ist VREP<sup>1</sup>. VREP wurde für verschiedene Robottikanwendungen entwickelt. VREP erlaubt es innerhalb eines Graphischen beliebige multi-body simulation (MKS), basierend auf diversen Physikengines zu konstruieren. Weiterhin verfügt VREP bereits über viele fertige Sensormodelle, wie beispielsweise den Velodyne VLP-16. Die ganze Simulation kann über ein RemoteAPI Interface mit nahezu jeder Programmiersprache kommunizieren.

Figure 5.8: VREP



### 5.2.1 Simulation Scenario

Für das Simulations Szenario wurde eine einfacher kleiner Kreisverkehr mit einem Außendurchmesser von 26m designt, da dieser aufgrund seiner Größe

1. <http://www.coppeliarobotics.com/>

das interessanteste Objekt ist. Und im Innerstädtischen Bereich häufig anzutreffen ist. Zum testen der Objektdetektion bietet dieser aufgrund seiner bebaubaren Mittelinsel gute Möglichkeiten das Tracking zu testen. Weiterhin ist es aufgrund seiner Größe möglich den kompletten Kreisverkehr zu überblicken. Das ganze Szenario wurde aufgrund von Limitierungen in Vrep um den Faktor 10 herunterskaliert.

Innerhalb des Szenarios bewegen sich innerhalb des Kreisverkehrs ein Fahrrad, ein Fußgänger und andere 2 Fahrzeuge. Die Objekte bewegen sich dabei auf festgelegten Pfaden. Die Geschwindigkeit aller Verkehrsteilnehmer ist dabei auf den Typ angepasst. Der Fußgänger bewegt sich mit für Fußgänger übliche 5km/h. Das Fahrrad mit 15km/h. Die beiden Autos bewegen sich mit unterschiedlichen Geschwindigkeiten zwischen 25 und 35 Km/h. Um eine Kollision der Fahrzeuge zu vermeiden, sind beide Fahrzeuge an der Vorderseite mit Distanzsensoren ausgestattet. Sollte dieser Sensor ein Fahrzeug erkennen, geht das Fahrzeug in eine Adaptive cruise control Modus und passt sich dem vorderen Fahrzeug an. Das Adaptive cruise control ist dabei als einfacher proportional Regler umgesetzt.

Das autonome Fahrzeug bewegt sich dabei ebenfalls auf einem festen Pfad, dieser führt von rechts in den Kreisverkehr ein und verlässt den Kreisverkehr an der dritten Ausfahrt. Die Aufgabe des Fahrzeuges ist dabei sich sicher in den Kreisverkehr einzusortieren und den Kreisverkehr sicher zu verlassen. Dabei muss das Fahrzeug sowohl auf den Fußgänger, den Radfahrer und die anderen Fahrzeuge innerhalb des Kreisverkehrs achten. Dazu ist das Fahrzeug mit einem virtuellen Velodyne VLP-16 ausgestattet.

### 5.2.2 Simulation Logic

Um das zuvor beschriebene Szenario durchzuführen musste eine Logik entwickelt werden, diese umfasst nicht nur die State Machine zum Durchfahren des Kreisverkehrs, sondern auch die Lokalisierung des Fahrzeuges in dem Szenario als auch die Anbindung aller Sensoren an die Objekterkennung. Da alle nachfolgenden Berechnungen nur wenig Ressourcen benötigen wurde die entsprechende Software in Python entwickelt.

### Sensor Connection

Die Objektdetektion benötigt als Sensor Eingänge die Daten des Velodyne VLP-16 und die Daten des Applix POS-LV. Von den Applix Daten werden jedoch nur die aktuelle Position im WGS84 Format benötigt und das aktuelle Heading. Da sich der Applix Sensor in VREP nicht einfach nachbauen lässt, wird das Applix System einfach aus den in Vrep auslesbaren Positionen und Rotationsdaten generiert. Da die Position in VREP in Kartesischen Koordinaten angegeben ist, werden dies über die in OpenDAVINCI enthaltene Transformation anhand einer Referenzkoordinate in WGS84 Koordinaten transformiert.

Die Anbindung des Velodyne-VLP gestaltet sich schwieriger, da der Aufbau der Messdaten sich signifikant vom Originalvelodyne unterscheidet. Zwar werden die Daten ebenfalls in Polarkoordinaten ausgegeben, jedoch werden nur Messwerte ausgegeben, wenn diese auf ein Objekt treffen. Da ein Großteil des Szenarios leerer Raum ist weisen die Messdaten viele Lücken auf, weshalb diese in eine geeignete Form transformiert werden müssen.

Dabei ist zu beachten, dass das Objekttracking eine vollständige 360 Grad

Messung erwartet, dazu muss zusätzlich gewartet werden, bis alle Nötigen Messdaten vorliegendn. Der virtuelle Velodynle liefert eine Messrate von 10Hz, und ist dabei in 4 Segmente eingeteilt, welche nacheinander ausgelesen werden. Der zur simulation verwendete Zeitschritt beträgt 50ms, so dass wir 2 Segemente in einem Zeitschritt auslesen können. Solbald alle Daten gesammelt sind, werden diese zu einem geeigneten Messframe zusammengesetzt.

Die glieferten Messdaten liegen als liste von Kugelkoordinaten vor (Radius  $r$ , Polarwinkel  $\theta$ , Azimutwinkel  $\Phi$ ). Da die Azimut und Polarwinkel dabei nicht in Aquidistanzen Abständen vorliegen, jedoch eine sehr viel höhere Auflösung als der originale Sensor haben, werden die Messwerte auf die Originale auflösung von 0.2 bzw. 2 Grad heruntergerundet und die Messerte in ein Entsprechendes zweidimensionales Array fester größe eingetragen eingetragen. Sodass nicht erfasste bereiche den Wert Null erhalten.

Nach dem alle Erforderlichen Messdaten gesammelt und umgewandelt wurden, werden diese über ein seperat entwickeltes OpenDAVINCI-Python interface an die Objekterkennung gesendet.

### **Mapping and Statemachine**

Laut RASt [RAS] soll ein Kreisverkehr möglichst kreisförmig sein, der einfachheit halber nehmen wir ür die Simulation an, dass dieser ein perfekter kreis ist. Das heißt, dass Sowohl Straßen, Radwege und fußwege perfekt kreisförmig sind und duch einen Radius beschrieben werden können.

---

# 6

## Evaluation

---

### 6.1 Simulation

---

### 6.2 Real Measurements



Kann in mehrere Unterkapitel gegliedert werden

Greift Thesen oder Fragestellungen aus der Einleitung wieder auf

Fasst die Arbeit knapp und prägnant zusammen

Ordnet die Ergebnisse in Gesamtzusammenhänge ein

Zieht Schlussfolgerungen aus den erarbeiteten Ergebnissen

Kann auch eigene Bewertungen oder Meinungen enthalten

Gibt einen Ausblick auf mögliche Konsequenzen oder notwendige weitere zu lösende Probleme

---

# 8

## Future Work

---

# Bibliography

- [1] Christian Berger. “Automating Acceptance Tests for Sensor- and Actuator-based Systems on the Example of Autonomous Vehicles”. PhD thesis. 2010, p. 298. ISBN: 9783832293789. URL: <http://www.christianberger.net/Ber10.pdf>.
- [2] Applanix Corporation. *POSLV SPECIFICATIONS*. 2015.
- [3] Martin Ester et al. “A density-based algorithm for discovering clusters in large spatial databases with noise”. In: AAAI Press, 1996, pp. 226–231.
- [4] Martin a Fischler and Robert C Bolles. “Random Sample Consensus: A Paradigm for Model Fitting with Applicatlons to Image Analysis and Automated Cartography”. In: *Communications of the ACM* 24.6 (1981), pp. 381–395. ISSN: 0001-0782. DOI: 10.1145/358669.358692. URL: <http://dx.doi.org/10.1145/358669.358692>.
- [5] M Himmelsbach, T Luettel, and H.-J. Wuensche. “LIDAR-based 3D Object Perception”. In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2009), pp. 994–1000. DOI: 10.1109/IROS.2009.5354493. URL: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5354493>.
- [6] M Holmes, a Gray, and C Isbell. “Fast SVD for large-scale matrices”. In: *Workshop on Efficient Machine Learning* 1.1 (2007), pp. 2–3.
- [7] Alonzo Kelly. *A 3D state space formulation of a navigation Kalman filter for autonomous vehicles*. Pittsburgh: Carnegie Mellon University, 1994, p. 95.
- [8] James F. Kurose and Keith W. Ross. *Computer Networking A Top-Down Approach*. 5. 2013, p. 4. ISBN: 9780132856201. DOI: 10.1017/CBO9781107415324.004. arXiv: 8177588788.

- [9] Bo Li, Tianlei Zhang, and Tian Xia. “Vehicle Detection from 3D Lidar Using Fully Convolutional Network”. In: *Robotics: Science and Systems XII*. Robotics: Science and Systems Foundation, 2016. ISBN: 9780992374723. DOI: 10.15607/RSS.2016.XII.042. arXiv: 10.15607. URL: <http://www.roboticsproceedings.org/rss12/p42.pdf>.
- [10] F. Moosmann, O. Pink, and C. Stiller. “Segmentation of 3D lidar data in non-flat urban environments using a local convexity criterion”. In: *2009 IEEE Intelligent Vehicles Symposium*. June 2009, pp. 215–220. DOI: 10.1109/IVS.2009.5164280.
- [11] Abdul Nurunnabi, David Belton, and Geoff West. “Diagnostic-Robust Statistical Analysis for Local Surface Fitting in 3D Point Cloud Data”. In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* I-3. September (July 2012), pp. 269–274. ISSN: 2194-9050. DOI: 10.5194/isprsannals-I-3-269-2012. URL: <http://www.isprs-ann-photogramm-remote-sens-spatial-inf-sci.net/I-3/269/2012/isprsannals-I-3-269-2012.pdf> %20http://www.isprs-ann-photogramm-remote-sens-spatial-inf-sci.net/I-3/269/2012/.
- [12] Meghashyam Panyam mohan Ram. “Least Squares Fitting of Analytic Primitives on a GPU”. PhD thesis. Clemson University, 2007, p. 103. URL: [http://tigerprints.clemson.edu/all%7B%5C\\_%7Dtheses](http://tigerprints.clemson.edu/all%7B%5C_%7Dtheses).
- [13] Robin Schubert, Eric Richter, and Gerd Wanielik. “Comparison and evaluation of advanced motion models for vehicle tracking”. In: *Information Fusion, 2008 11th International Conference on* 1 (2008), pp. 1–6. DOI: 10.1109/ICIF.2008.4632283. URL: [http://ieeexplore.ieee.org/xpls/abs%7B%5C\\_%7Dall.jsp?arnumber=4632283](http://ieeexplore.ieee.org/xpls/abs%7B%5C_%7Dall.jsp?arnumber=4632283).
- [14] Inge Söderkvist. *Using SVD for some fitting problems*. Tech. rep. 2. Sweden: Lulea University of Technology, 2009, pp. 2–5. URL: [https://www.ltu.se/cms%7B%5C\\_%7Dfs/1.51590!/svd-fitting.pdf](https://www.ltu.se/cms%7B%5C_%7Dfs/1.51590!/svd-fitting.pdf).
- [15] Forschungsgesellschaft für Straßen- und Verkehrswesen. Arbeitsgruppe Straßenentwurf. *Merkblatt für die Anlage von Kreisverkehren*. 2006.
- [RAL] Forschungsgesellschaft für Straßen- und Verkehrswesen. Arbeitsgruppe Straßenentwurf. *Richtlinien für die Anlage von Landstrassen (RAL)*. FGSV (Series). FGSV-Verlag, 2013.
- [RASt] Forschungsgesellschaft für Straßen- und Verkehrswesen. Arbeitsgruppe Straßenentwurf. *Richtlinien für die Anlage von Stadtstraßen: RASt 06*. FGSV (Series). FGSV-Verlag, 2007. ISBN: 9783939715214.
- [16] Wikipedia. *Linear least squares (mathematics)*. 2017. URL: [https://en.wikipedia.org/wiki/Linear\\_least\\_squares\\_\(mathematics\)](https://en.wikipedia.org/wiki/Linear_least_squares_(mathematics)) (visited on 03/06/2017).

- [17] Liang Zhang et al. “Multiple Vehicle-like Target Tracking Based on the Velodyne LiDAR\*”. In: *IFAC Proceedings Volumes* 46.10 (June 2013), pp. 126–131. ISSN: 14746670. DOI: 10.3182/20130626-3-AU-2035.00058. URL: <http://linkinghub.elsevier.com/retrieve/pii/S1474667015349211>.