OTTO-VON-GUERICKE-UNIVERSITÄT MAGDEBURG

FAKULTÄT FÜR INFORMATIK
Institut für Intelligente Kooperative Systeme

Master's Thesis

Autonomous Driving in Urban Centers - Roundabout Monitoring

Julian-B. Scholle
March 10, 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 10, 2017

Julian-B. Scholle

Acknowledgements

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

Contents

1 Introduction

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

1.1 Motivation

Da die Technische Hochschule Chalmers ergänzend zu Volvos "DriveMe" Projekt das Projekt "CampusShuttle" initiiert, "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 Fußgängern sicher agieren.

1.2 Research Question

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

2 Related Work

2.1 Autonomous Driving

2.2 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 [14] whereupon RASt is based on.

2.2.1 Elements of a Roundabout

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. [14]*

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). [14]*

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 routeing (one or two lanes). The edge strip width is oriented on the relevant continuous roadway. [14]*

Definition 2.4 (outer diameter (D)) *The outer diameter is measured at the*

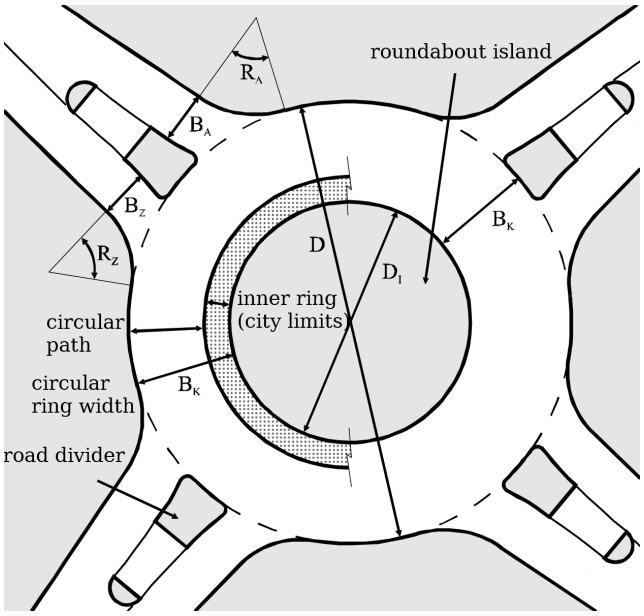


Figure 2.1: Definition of individual design elements and dimensions of a roundabout [14]

outer edge of the circular ring. It is the essential measure for describing the size of the roundabout. [14]

Definition 2.5 (inner diameter (D_I)) *The inner diameter is the diameter of the roundabout island. [14]*

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. [14]*

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. [14]*

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. [14]*

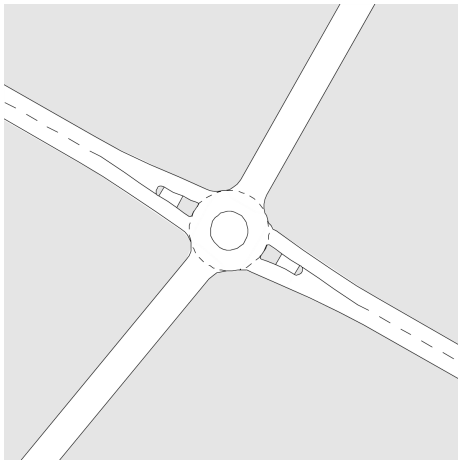
2.2.2 Types of Roundabouts

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. [14]

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

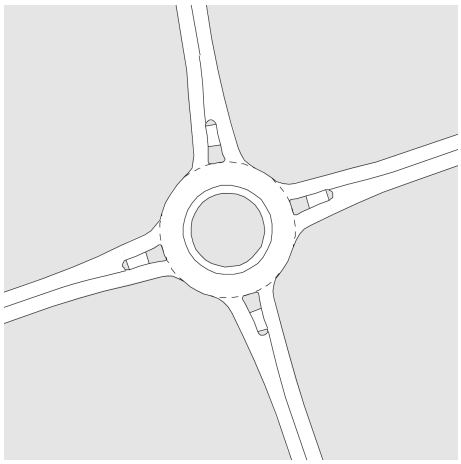
Figure 2.2: Mini Roundabout [14]



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.

Small Roundabout

Figure 2.3: Small Roundabout [14]



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

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.

Figure 2.4: Two-lane Passable Roundabout [14]

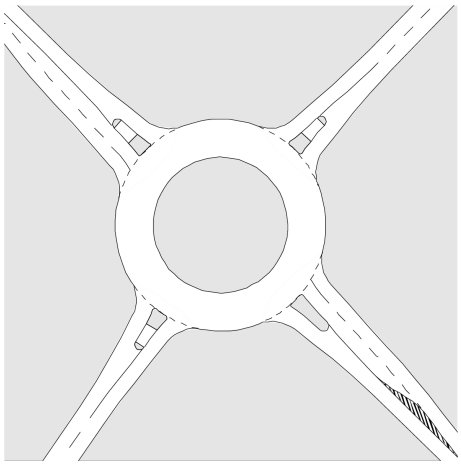
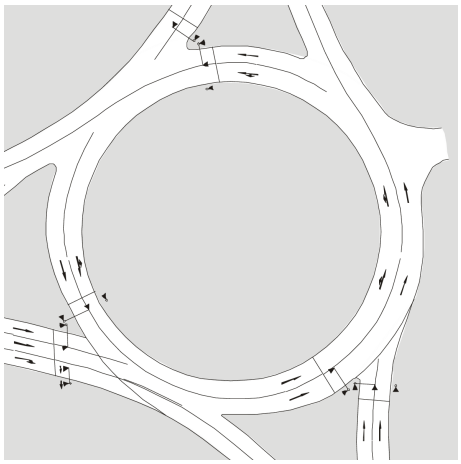


Figure 2.5: Large Roundabout [14]



Large Roundabout

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.

2.3 Middleware OpenDAVINCI

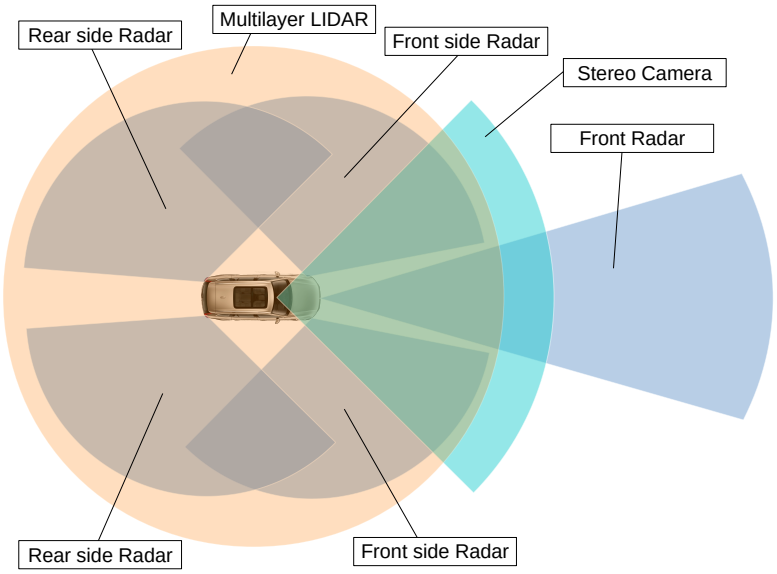
Autonome Software ist typischer weise ein verteiletes System, auf heutigen Fahrzeugen basiert dieses System auf ECUs und Bussystemen wie CAN,LIN. Verteilete Sysoftware vereinfacht es komplexe Komponenten innheralb des Systems zu integreieren. Im bereich des Autonomen Fahrens iust der Historische aufbau von Fahrzeugen mit ECU'S und can jedoch nicht optimal. Um die vielen benötigekten Komponenten zu handhaben, ist es von vorteil Komponenten auch innerhalb eines ECU's bzw einer Recheneinheit zu entkoppeln. Für diesen Zweck gibt es beireits mehrer middelwares die unteranderm die Kommunikation innerhalb der Komponenten handhaben und abstrahieren. m Rahmen des Copplar Projekets, wird hier die OpenDaVINCI middleware genutzt. OpenDaVINCI ist eine echtzeitfähige laufzeitumgebeung konzipiert für Autonome Fahrzeuge. OpenDaVINCI basiert auf Hesperia [1]. Die Kommunikation zwischen den Komponenten basiert in OpenDaVINCI UDP Multicast, welches eine Echtzeitfähige Kommunikation zwischen den Komponenten Ermöglicht [7]. 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 weiter Funktionalitäten die das Handling von World Geodetic System 1984 (WGS84) Korridinaten an, welches für die Umwandlung von GPS koordinaten in lokale kartesische genutzt werden kann. Dazu ist die Angabe einer referenz GPS postion nötig, welche um den Berechnugsfehler klein zu halten, nicht zu weit entfernt sein sollte.

2.4 Test Platform

Die in dieser Abreit genutze Testplatform ist ein Volvo XC90 (2015) SUV. Dies Testpaltform ist mit vielen Sensoren zur Umfeldwarnehmung 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. 2.6 entnommen werden.

Figure 2.6: Test Platform



Zusätlich zur Fahrzeuginternen Sensorik (Odometer, Interialsensorik) ist im Fahrzeug ein Applanix POS LV verbaut. Zu Zeitpunkt des Verfassens dieser Arbeit war es leider nicht möglich auf die Radarsensoren und die Stereokamera zuzugreifen. Daher werden im folgenden lediglich der Velodyne Lidar und das Applanix System genauer beschrieben.

2.4.1 Velodyne VLP-16 LiDAR

Der Velodyne VLP-16 ist ein 360 Grad 3D Laserscannermit 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 syncronisiert werden, was eine jitterarme Zeimessung ermöglicht. Eine weiter 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 Transparent Objekte zu sehen. Das ermöglicht uns im späteren Verlauf die Breite des Fahrzeues zu ermitteln, da der Velodyne durch die Glasfenster des Fahrzeues blicken kann. Bei einer eingestellten Geschwindigkeit von 10Hz liefert der VLP-16 eine Auflösung von 0.2 Grad bei einer Abreichung von +-3cm. Der VLP-16 ist mittig auf dem Dach des XC90 moniert, um eine möglichst hohe Positionierung zu erreichen, die eine Rundumsicht umd Das Fahrzeug zu erreichen. Zu beachten ist, das diese Ausrichtung für den Sensor denkbar ungünstig ist, da der Sensor ein vertikales Sichtfeld von -15 bis +15 Grad hat. Dadurch sind nachezu alle messungen über Null grad quasi nutzlos. Der blick auf die Herstellerseite [1] verrät, das Der VLP-16 aunteranderem auf die verwendung mit Drohnen hin konstuiert wurde, während der Größere HDL64E [2] explizit für den Urbanen Automotivebereich beworben wird, und über ein Sichtfeld von +2 bis -24.9 Grad verfügt. Die dabei entstehenden Probleme werden später diskutiert.

2.4.2 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 Inertialsensork und Odometrie gestützte Positionsmessung mit einer Genauigkeit von bis zu 0.3m (bis zu 0.035m bei verwendung von der 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 Odeomerter und der Inertialsensork eine Position liefern. Diese wird jedoch über die Zeit schlechter, so das 60Sek nach Ausfall des GPS-Signals nich eine Genauigkeit von 2.51m erwartet werden kann.[2]

1. `http://velodynelidar.com/vlp-16.html` (03/09/2017)
2. `http://velodynelidar.com/hdl-64e.html` (03/09/2017)

3 Methodology

kreisverkehre überabreiten

Im vorigen Kapitel haben wir uns die Arten von Kreisverkehren und derren Komponenten angesehen. Weiterhin haben wir die zur verfügung stehende Testplatform und ihre Sensorik begutachtet. Dabei haben wir festgestellt, das für die Erkennung von Objekten in andren Arbeiten häufig mehrere und teurere Sensoren kombiniert werden um ein Zuverlässiges erkennen von anderen Verkehrsteilnehmern zu gewährleisten. In der Research Questian eins haben wir festgehalten, das wir die verwendung eines günstigen VLP-16 Sensors in einem Komplexen Verkehrszenario, die Verkehrsbeobachtung eines Kreisverkehrs evaluieren wollen.

beleg

Dazu wird im folgenden ein Algorithms 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 Umfeldsensors, 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 Algorythmen im Bereich der Segmentierung häufig fehl. Aus diesem Grund wird für die Segmentierung eine Groundplane basierender Algorithms 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 Algorithms entwickelt welches dies gewähreisten soll.

section reference

Zur Evaluation dieser Algorythmen wurden mehrere Datensammlungen auf den nahe gelegenen schwedischen AstaZero fig. 3.1 Prüfgelände in Sandhult durchgeführt, für alle nicht dort durchgeführten Expirimente werden in einer dafür erstellten Simuation durchgeführt. In dieser Simulation wird ein Innerstätischer Kreiverkehr mit Fuß un 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

dr Objekte werden ebenfalls verkmerkt.

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

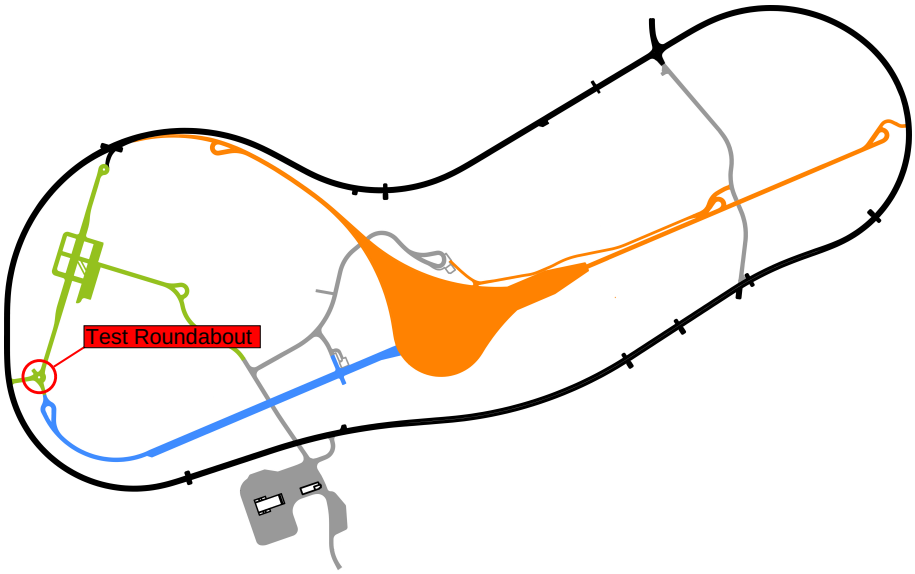


Figure 3.1: AstaZero Proving Ground
http://www.astazero.com/wp-content/uploads/2016/09/%C3%96versiktsskiss_mod.pdf

4 Research

4.1 Objekt Detection

4.1.1 Ground Removal

Um in einer PointCloud Ojekte 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 Mehode hat allerdings viele Nachteile, zum einen muss der LIDAR Sensor exakt gerade auf em 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 Palanaren Grundflächen, alo flache nicht hügelige Untergründe. Eine weitere Verbreitete Methode ist das Entfernen der Bodenplatte auf basis eines Statistischer mittelwertes [16]. Diese Methode benötigt allerdings auch eine Kalibireirung der Sensorabstandes zum Boden. Und die Bestimmung weiterer Schwellwerte, welche umgebungsabhäng sind. Die Votreile beider Methoden sind ihre gering nötige rechenleistung und laufzeit O(n). Bessere Methoden wie Gradientenbasierende explansions algrythmenm benötigen einen Startpunkt der als Bodenplatte identifiziert werden kann. Eine weitere Möglichkeit ist die Beschreibung von Objekten als Konvexe Objekte [9], die ebenfalls auf Basis der Gradienten beschrieben werden kann. Vorteil dieser Methode ist das keine Initiale Position für die Bodenplatte benötigt wird.

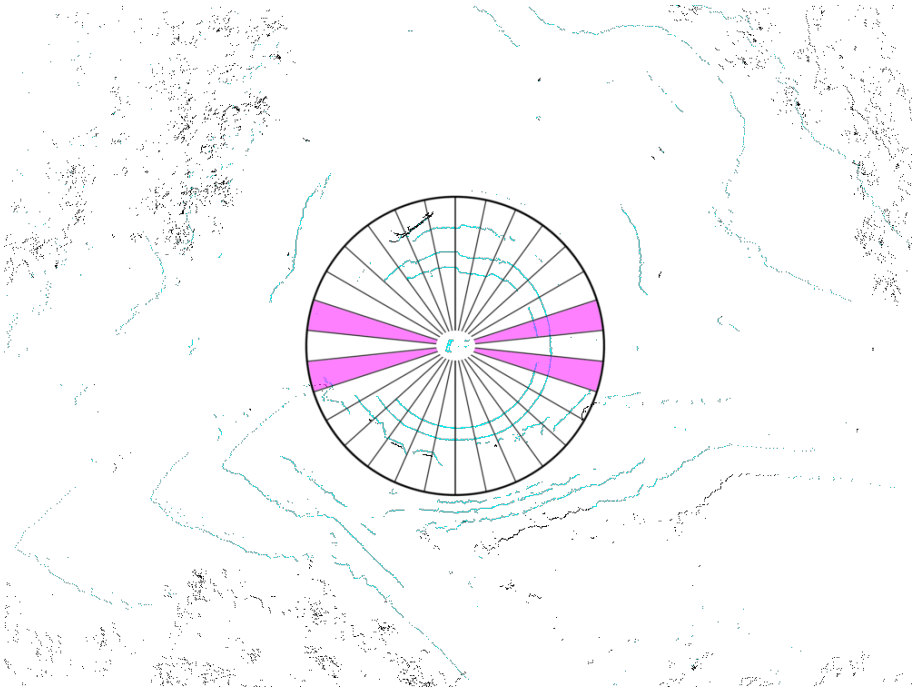
Für unseren Anwendungsfall mit dem Velodyne VLP-16 besteht das Problem darin, dass die Auflösung des Sensor in der Höhe sehr gering ist. Abhängig von der Entfernung des Fahrzeuges innerhalb der benötigten Reichweite fallen nur zwei Lagen auf die Testfahrzeuge, wesshalb Gradientenbasierende Methoden hier zuverlässig versagen. Da die Gradienten zu klein sind und die verkelinerung der nötigen Thresholds zu haufigen false Postitives führt. Die Methode des Statistischen Mittelwertes und die Methode auf basis der Z-Koordinate, leiden am Fahrwerk des Volvo XC90 SUV. Die Höhe das Fahrzeuges ändert sich aunteranderem durch veränderung des Fahrprofiles (Sport/Eco, etc.) um mehrere Zentimeter. Auch leicht erhöhte Geschwindigekiten im Kreisverkehr (ca 30 km/h) führen zu einer deutlichen Seitenneigung des Fahrzeuges. Darum

beleg

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 lösst 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 in 30 Tortenstück förmige Segmente geteilt. Aus diesem Tortenstück werden dann jeweils vorne und hinten zwei Segmente [fig. 4.1] ausgewählt, welche nicht beachbart sind. Die Auswahl der Segmente folgt aus der Annhame, dass sich die Straße vor, bzw hinter dem Fahrzeug befindet. Zukünfitg könnte die Auswahl der Segmente auch mit hilfe des Fahrzeuglenkwinkes optimiert werden. Oder der Gültige Bereich von einer Lane Detection geliefert werden.

Figure 4.1: 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 wit weg währen. Die Einteilung in Segmente ist desshalb nötig um zu verhindern, dass alle Messerte in ein einziges lokales Minima laufen. Aus diesem Vorgefilterten Messwerten werden nun für einen Random Sample Consensus (RANSAC) drei zufällig herausgesucht. Aus diesm dei Punkten wird nun eine Ebene in der Hessischen Normalform gebildet ,was eine effiziente Distanzberechnung zu anderen Punken erlaubt. Danach sammeln wir alle weiten Punkte aus unseren Minima, anahnd eines Distanzkriteriums. Danach wird aus der Ebene und den neu Gesammelten Punkte durch einen Planefitting Algorithms [section 4.1.1] eine neue Ebene und derren Fehler berechnet. Der Fehler wird über die Summe der quadratischen Abstände aller Punkte zur Ebene berechnet.

Bevor wir die Ebene jedoch als eventueller Lösungskanidat 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 Groundflache makiert. Als threshold wurde hier expirimentel ein optimaler Wert von 0.5m ermittelt.

Planefitting

Zum Planefitting einer be ne wird üblicherweise eine Singular Value Decomposition (SVD) [10, 11, 13]. SVD hat eine Komplexität von $\mathcal{O}(\min\{mn^2, m^2n\}$ [5], da das Planefitting innehalb des RANSAC sehr häufig mit einer großen Anzahl an Punkten ausgeführt wird, führt das Ausführen des SVD innhalb des RANSAC zu einer sehr hohen laufzeit. Deshalb wird an dieser Stelle ein Linear least Squares (LLSQ) Algorithus 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 entfert sind. Da wir unsere Punkte innhalb des RANSAC allerdings anhand eines Distanzkriteriums vorselektieren, stellt dies kein Problem dar.

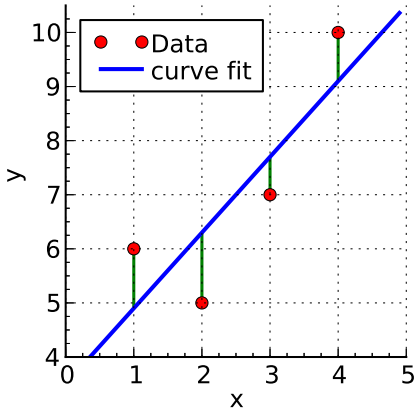


Figure 4.2: Linear least Squares (LLSQ) [15]

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:

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

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

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:

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

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

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

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

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 $P_i = P_i - \overline{P}$. 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

$$\begin{aligned} D &= \sum x_i x_i \cdot \sum y_i y_i - \sum x_i y_i \cdot \sum x_i y_i \\ a &= \frac{\sum y_i z_i \cdot \sum x_i y_i - \sum x_i z_i \cdot \sum y_i y_i}{D} \\ b &= \frac{\sum x_i y_i \cdot \sum x_i z_i - \sum x_i x_i \cdot \sum y_i z_i}{D} \\ \vec{n} &= [a, b, 1]^T \end{aligned}$$

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 typischweise sehr groß. Sollte die Determinante doch nahe 0 (nicht gleich 0) sein, wird die Berechnung trotzudem 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 ausssortiert. Ist die Determinante extakt Null, wird die Berechnungstatdessen mit einem kleinen Wert für D fortgesetzt.

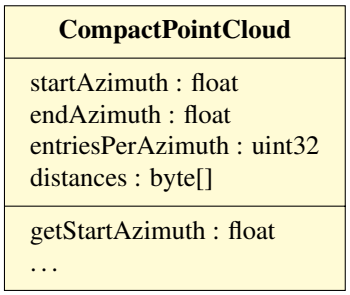
Aus dem Normalenvektor $\vec{n}$ und dem Mittelwert-Punkt $\overline{P}$ können wir nun wieder die Hessische Normalenvektor bestimmen.

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

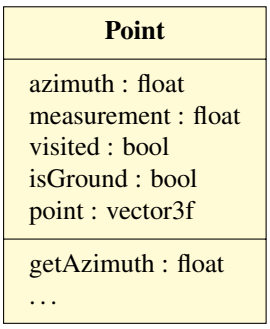
4.1.2 Clustering

In aktuellen Abreiten mit 3D-LIDAR Daten werden de Daten haufig als erstes in eine Heightmap projeziert [16, 4, 8]. Danach werden direckt benachbarte Messungen mit ähnlichen Messwerten zusammengefasst. Alternativ werden die Messungen auch anhand eines Distanzkritterums zusammengefasst. Erstere Methode hate den Nachteil, dass einzelne Ausreißer dazu führen, das das Objekt in mehrer Cluster zerfällt. Letztere wird meißt mit einem KD-Tree oder einer ähnlichen Datenstrucktur 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. Dauzu 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.



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 abgebildtet werden muss. Nachdem die sphärische Daten wiederhergestellt wurden, werden diese In Kartesische umgewandelt und in eine Punkt Datenstruktur gespeichert.



Diese wird wiedrrum 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 Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [3] ausgeführt. Der DBSCAN Algorithms 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 überlichweise durch die Nachbarschafts berechnung verursacht. Genau hier setzen wir an, anstatt der Bereichsanfrage über eine Baum-Struktur nutzten 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 Berchnung übersprungen, zusätzlich entfällt der Aufbau eines KD-Trees.

4.1.3 Tracking

datenstrucktur Onjekte 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 geringefügig bewegen, weiterhin ändert sich die Form der Cluster ebenfalls nur leicht. Das ist wichtig, da die Position eine Clusters durch einen Mittelwertpunkt definiert ist. Das Tracking wird im R2 durchgeführt. Im Initialen Schritt wird jeden Cluster eine aufsteigende ID zugerordnet. In jedem weiteren Schritt wird jedem neuen Cluster die ID des alten Clusters zugeordnet welcher über die Zeit hinweg die geringste Entfernung aufweißt. Fur diese Entfernung gibt es eine großzügige obere Schranke von 3m, Cluster die nicht innerhalb in 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 gtrackten Cluster. Im Initialen Schritt werden aus allen Cluster mit der Selben IDs Objekte gebildet. In jeden weiteren Schritt werden Alle Cluster mit der zuvor gleichen ID zum updaten 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 Applanix POS-LV genutzt. Da sowohl die Positionsdaten des Applanix Systems, als auch die Erkannte Postition des Fahrzeuges Fehlerbehaftet sind, wird die Bewegungrichtung nur bei einer minimalen Bewegung

von 2m geupdatet.

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

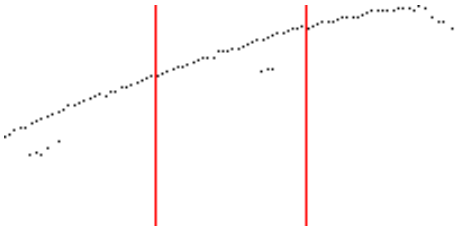
mit P - Postition des Objekts, C - Position des eigenen Fahrzeuges

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

Figure 4.3: Obstacle Movement

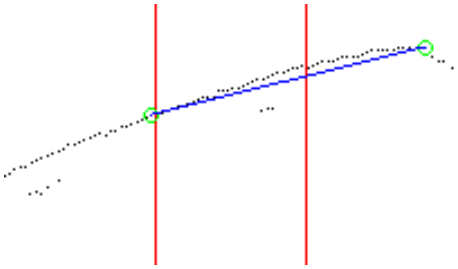
Basierend auf der Bewegungrichtung wird nun die Ausrichtung des Fahrzeiges berechnet. Dazu werden alle dem Objekt zugewiesenden Cluster zusammengefasst und um $-\theta$ gedreht. Danach wird das Objekt wird in 3 gleich große Segmente unterteilt (fig. 4.4).

Figure 4.4: 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 aso 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, das bei perfekt wagerechte ausgerichteten Objekt beide Maxima in den selben Punkt laufen. Mit diesen Punkten $(\vec{R};\vec{L})$ wird nurn eine korrektur der Drehung des Objektes berechnet:

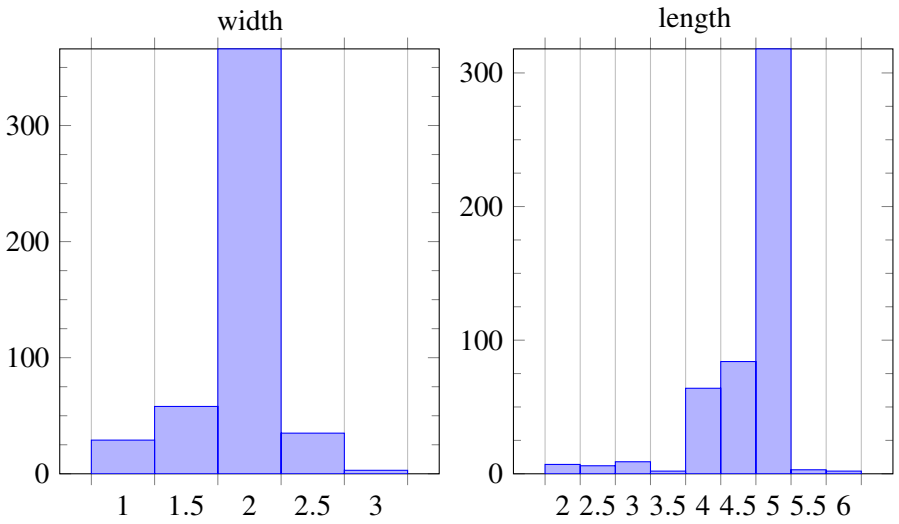
Figure 4.5: θ - Correction



$$\Delta x = R_x - L_x$$
$$\Delta y = R_y - L_y$$
$$\theta_{correction} = \text{atan2}(\Delta y, \Delta x)$$

Nach Anwendung der Korrektur wird die größe des Hindernises 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 Histogram für die Länge und Breite des Hindernises aufgebaut. Anhand diesem wird dann der warscheinlichste 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 4.6: Object Size Histogram



Leicht zu sehen wird für das Object eine breite von 2m und eine Länge von 5m berechnet. Das Objektist in diesm 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, wurd 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 Größenändrungen sehr instabil ist. Deswegen wird als Position immer die maximale x-Koordinate der Cluster gesnutzt. Diese Postition kann ebenfalls in fig. 4.3, als grüner Punkt begutachtet werden. Da θ im initialen Zeitschritt Null ist, entpricht dies der globalen Maximalen x-Koordinate des Clusters. Dies Führt dazu, das wir im Initialen Zeitschritt annhemen, 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 ander 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 [16, 4]. Die Minimum Bounding box liefet jedoch unter umständen nicht das gewünschte ergebnis, zum einen hat die immer nur die größe der aktuellen Messung und zum anderen kann sie eine falsche Orientierung liefern, wie in fig. 4.7 zu sehen.

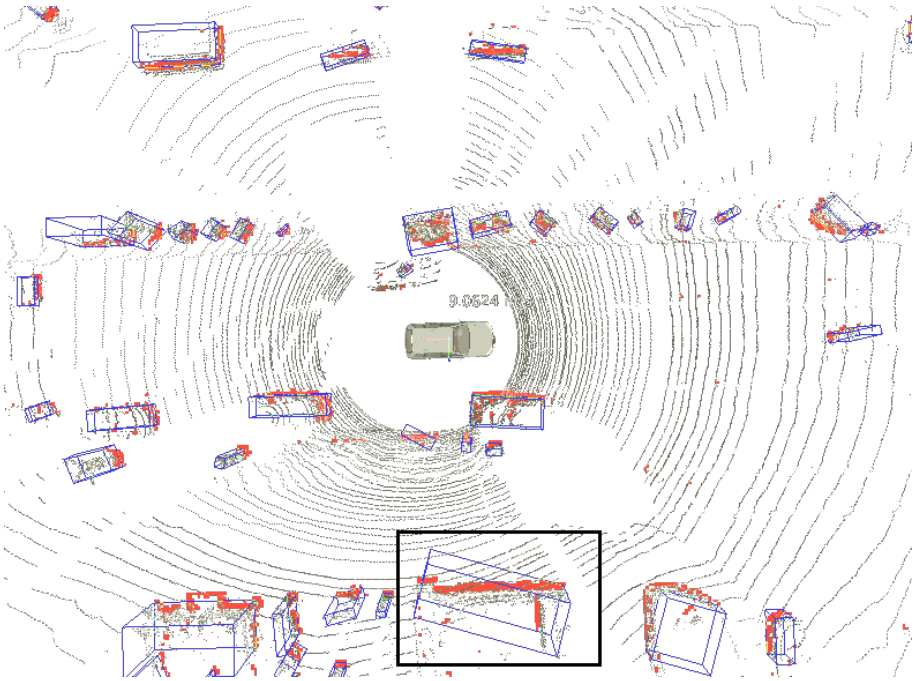


Figure 4.7: Error with minimum Boundingbox [4]

Da die orientierung und Postition der Objekte jedoch als eingang für den nachfolgenden Kalmanfilter genutzt wird, welcher sehr sensible auf falsch Orientierungen reagiert, wurde eben jeder Algorithmus entwickelt.

Object Confidence

implementierung Konfidence wert auf 1 setzen und entternen, wenn wert unter 1, oder objekte auf basis ihrer Position entfernen

Um zu verhindern, dass kurzzeitig erkannte False Positives direckt 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 Konfidence Wert eines Objektes ist Null. Der Konfidence wert wird um eins erhöht, wenn das Objekt in zwei aufeinader folgenden Zeitschritten getrackt werden kann und folgende bedingungen erfüllt:

- Die breite des objkts muss kleiner seine als die länge des hindernisses zuzüglich 1.5m
- Die länge des Hindernises muss kleiner sein als 10m
- Die Breite des Hindernisses muss kliener sein als 4m

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

4.1.4 Classification

filterung in code entfernen, weil größe bereits getiltert wird

implementierung Plausibilitätstest

Die Klassifizierung der Objekte finedt anhand ihrer Größe statt, es findet keine Klassifizierung nach bewegten und unbewegten Objeten Stadt. Unterschieden werden lediglich Fußgänger, Radfahrer, und Fahrzeuge, und Sonstige. Als Klassifizierungs kritterium wird die Größe der Onjekte 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 andhand der Geschwindigkeit eine Plausibilitätstest durchgeführt. So darf eine Fußgänger eine Geschwindigkeit von 10km/h nicht überschreiten und die Maximalgeschwindigkeit dür einen Radfahrer bertägt 30km/h. Da für Fahrzeuge keine sinnvolle geschwindigkeitsgrenze angenommen werden kann, wir an ihrer stelle die Änderung der Orientierung genutzt. Als Maximale Drehrate wird eine Messung von 0.3 rads/sec aus [6] angenommen. Da der Wert eine Obere Grenze darstellen soll nehmen wir einen etwas höheren Wert von 0.3 rads/sec an.

4.1.5 State Estimation

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

Aus dem zuvor durchgführten Tracking können wir die Aktuelle Position, Geschwindigkeit, Drehung und Drehgeschwindigkeit erhalten. Für eine Zustandschätzung mit den für Fahrzeuge üblichen Bicycle Model fehlen und Angangen über den Radstand und Gewicht des Fahrzeues. 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 nutzten. Da dieses Model ebenfalls auf Fußgänger und Radfahre angewendet werden kann.

Constant Turn Rate and Velocity Model

Der Zustandvektor [12] des CTRV- Modells sieht wie folgt aus:

$$\vec{x}(t) = \begin{bmatrix} x & y & \theta & v & \omega \end{bmatrix}^T$$

x - Y Axis

y - X Axis

θ - Object Yaw Angle

v - Object Velocity

ω - Yaw Rate

Die Dynamikmatrix erhalten wir durch eine nichtlinieare 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 Rechihweite des Velodyne Sensors und wird im Darauffolgenden Zeitschritt nicht erkannt, wird der Prediktionsschritt 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 Postion mit allen Onjekten in der Prediktionsphase abgeglichen. Befindet sich das neue Objekt nahe an der predizierten Position wird der Cluster dem Onjekt zugewiesen, und der Korrekturschritt des EKF durchgeführt.

4.2 Mapping

4.2.1 OpenDAVINIC Map

4.2.2 Simplification

5 Evaluation

5.1 Simuation

5.2 Real Measurements

6 Conclusions

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 eine Ausblick auf mögliche Konsequenzen oder notwendige weitere zu lösende Probleme

7 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] 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.

[5] 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.

[6] Alonzo Kelly. *A 3D state space formulation of a navigation Kalman filter for autonomous vehicles*. Pittsburgh: Carnegie Mellon University, 1994, p. 95.

[7] 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.

[8] 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.

[9] 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.

[10] 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/.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] Wikipedia. *Linear least squares (mathematics)*. 2017. URL: https://en.wikipedia.org/wiki/Linear_least_squares_(mathematics) (visited on 03/06/2017).

[16] 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.