

Visuelle Odometrie zur Lokalisierung autonomer Fahrzeuge

vorgelegt von

Gina Hortenbach

Matrikel-Nr.:897854

Fachbereich VI – Informatik und Medien – Technische Informatik
der Berliner Hochschule für Technik Berlin
vorgelegte Masterarbeit
zur Erlangung des akademischen Grades

Master of Engineering (M.Eng.)

im Studiengang

Technische Informatik - Embedded Systems

Tag der Abgabe 21. Mai 2022



Gutachter

Prof. Dr.-Ing. V. Sommer Berliner Hochschule für Technik
Prof. Dr.-Ing. H. Höfig Berliner Hochschule für Technik

Hiermit erkläre ich,

- dass ich die vorliegende Arbeit selbstständig verfasst habe,
- dass keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommenen Aussagen als solche gekennzeichnet sind,
- dass die eingereichte Arbeit weder vollständig noch in wesentlichen Teilen Gegenstand eines anderen Prüfungsverfahrens gewesen ist,
- dass ich die Arbeit weder vollständig noch in Teilen bereits veröffentlicht habe und
- dass ich mit der Arbeit keine Rechte Dritter verletze und die Universität von etwaigen Ansprüchen Dritter freistelle.

Berlin, den 21. Mai 2022

Kurzfassung

In der vorliegenden Abschlussarbeit "Visuelle Odometrie zur Lokalisierung autonomer Fahrzeuge" wurde die gefahrene Trajektorie eines Fahrzeugs auf mehreren Teststrecken ausschließlich anhand von Kameradaten geschätzt. Die Testfahrten wurden zu unterschiedliche Licht- und Witterungsverhältnisse durchgeführt mit Streckenlängen bis zu 2 km.

Ein Stereokamerasystem und ein RGB-D Kamerasystem wurden am Fahrzeug montiert und miteinander verglichen. Die Fahrzeugpose wurde für beide Kamera-systeme mit der selben merkmalsbasierten Visuellen Odometrie geschätzt. Projektive Geometrie und die Extraktion von Bildmerkmalen zählt daher zu den zentralen Themen der Arbeit.

Als Testumgebung wurde der CARLA Simulator verwendet. Geschätzten Odometrie Daten werden in einem ROS2 Netzwerk veröffentlicht wodurch die Implementation hoch modular ist und problemlos in bestehende ROS2 basierte Systeme integriert werden kann.

Insgesamt konnten Orientierung und Position mittels visueller Odometrie zuverlässig geschätzt werden, wobei das System mit den Daten der RGB-D Kamera-system geringer Schätzfehler erzielte.

Abstract

In the present thesis "Visual odometry for the localization of autonomous vehicles", the driven trajectory of a vehicle on test routes was estimated exclusively using camera data. The test drives were carried out under different light and weather conditions with distances of up to 2 km.

A stereo camera system and an RGB-D camera system were mounted on the vehicle and recorded together. Vehicle pose was estimated for both camera systems using the same feature-based visual odometry. Projective geometry and the extraction of image features are therefore among the most important topics of the work.

The CARLA simulator was used as a test environment. Estimated odometry data is published on a ROS2 network, making the implementation highly modular and easily integrated into ROS2 based systems.

Overall, orientation and position could be reliably estimated using visual odometry, with the system using the data from the RGB-D camera system achieving lower estimation errors.

Inhaltsverzeichnis

1 Einleitung	1
1.1 Motivation - Forschungsprojekt CARS	1
1.2 Ziel der Arbeit	1
2 Grundlagen	2
2.1 Lokalisierungsverfahren	2
2.1.1 Globale Lokalisierung	2
2.1.2 Position Tracking	2
2.2 3-D Strukturen und Kamerapose aus 2-D Bildern	3
2.2.1 Koordinatensysteme	4
2.2.2 Korrespondenzanalyse	6
2.2.3 Verbinden von zwei Ansichten durch Epipolareometrie	7
2.2.4 Sonderfall Stereo Vision	8
2.2.5 Structure From Motion - SfM	9
2.2.6 Simultaneous Localization and Mapping - SLAM	9
2.2.7 Verschiedene Ansätze für Visuelle Odometrie	10
3 Extraktion von Bildmerkmalen	13
3.1 Auffinden von Merkmalen	13
3.1.1 Merkmale Beschreiben durch Nachbarschaftsoperationen	13
3.1.2 Algorithmen zum Auffinden von Interest-Operatoren	14
3.2 Matching	18
3.2.1 Übereinstimmungen finden	18
3.2.2 Matching Ergebnisse Filtern	19
3.2.3 Stereo Matching	19
3.2.4 Semi Global Block Matching	19
3.3 Deskriptoren	20
3.3.1 Histogram Deskriptoren	21
3.3.2 Binäre Deskriptoren	21
3.3.3 ORB - Oriented FAST and Rotated BRIEF	22
4 Kamerasysteme	23
4.1 Kamerasensoren	23
4.1.1 RGB Kamera mit CCD- oder CMOS-Sensoren	24
4.1.2 Tiefenbildkameras auf Basis von Projektionen	25
4.1.3 Tiefenbildkameras auf Basis von Stereoskopie	25
4.1.4 Time-of-Flight Tiefenbild	26
4.2 Kamerakonfigurationen	26

4.2.1	Monokulare Kamerasysteme	26
4.2.2	Stereo Kamerasysteme	27
4.2.3	RGB mit Tiefenbildkamera	27
4.2.4	Rektifizierung	28
5	Systementwurf	29
5.1	Erzeugen von Sensoroutput	29
5.1.1	Kameradaten	29
5.1.2	Node für Kamerasynchronisation	29
5.2	Stereo Visuelle Odometrie Ablauf	31
5.2.1	Merkmalendetektion	31
5.2.2	Matching von Merkmalen	32
5.2.3	Tiefeninformation	34
5.2.4	Bewegungsschätzung	34
5.2.5	Pose Update und Verlorene Odometrie	35
6	Implementierung	37
6.1	Simulationsumgebung	37
6.1.1	Simulator CARLA	37
6.1.2	ROS2 Framework für Prozesskommunikation	37
6.1.3	Kameradaten	38
6.1.4	Transformationsbaum	39
6.2	Bibliotheken	40
6.2.1	Computer Vision Bibliothek OpenCV	40
6.2.2	V-SLAM Bibliothek RTAB-Map	41
7	Ergebnisse der Lokalisierung	42
7.1	Methodik	42
7.2	Ergebnisse	46
7.3	Diskussion	52
7.4	Ausblick	52
	Literatur- und Quellenverzeichnis	53

Abbildungsverzeichnis

2.1	Abbildung von 3-D zu 2-D	3
2.3	Beziehung zwischen inhomogenen und homogenen Koordinaten	5
2.4	Die Kameraprojektion	6
2.5	Bildebene mit Epipolarlinien	7
2.6	Parallele Epipolarlinien	8
2.7	SLAM Map aus CARLA Testfahrt	10
2.8	Visuelle Odometrie Problemstellung	11
3.1	Harris Interest-Operator Detektor	15
3.2	GFTT Detektor Merkmalsfindung	16
3.3	FAST Detektor Speedtest	17
3.4	Effekt der Fenstergöße bei Disparitätenkarten	20
3.5	Histogram Deskriptor aus Gradienten	21
3.6	Muster binärer Deskriptoren	21
4.1	Aufbau CCD vs CMOS	23
4.2	3 Transistor CMOS Schaltung	25
4.3	Verschneite Straße	27
4.4	Rektifikation der Stereobildebenen	28
5.1	Eingangsdaten Bilder aus Stereoaufbau in CARLA Simulator	29
5.2	Graph der ROS2 Topics	30
5.3	Visuelle Odometrie Pipeline	31
5.4	Merkmalsdetektion in einer CARLA Szene	32
5.5	Brute Force Matching ohne Crosscheck	33
5.6	Brute Force Matching mit Crosscheck	33
5.7	Disparitätenkarte	34
6.1	ROS2 Node Netzwerk	37
6.2	CARLA Tiefenkarte	38
6.3	ROS2 Transformationsbaum für Visuelle Odometrie	39
6.4	OpenCV Bibliothek Download Statistik	40
7.1	CARLA Simulator Karte Town03	43
7.2	Plot der Trajektorien Teststrecke 'Default 01'	46
7.3	Testfahrten Streckenlängen	46
7.4	Witterungsverhältnisse der Simulation in verschiedenen Datensätzen	47
7.5	Absolute Trajectory Error (ATE) für Stereokamera	48
7.6	Absolute Trajectory Error (ATE) für RGB-D	48

7.7 Boxplots Absoluter Fehler von Streckenabschnitten	49
---	----

1. Einleitung

1.1 Motivation - Forschungsprojekt CARS

Die vorliegende Abschlussarbeit ist im Rahmen des "Cooperative Autonomy based on Reliable Services"(CARS) Forschungsprojekts entstanden. CARS ist ein von der IFAF Gefördertes Projekt unter der Leitung von Prof. Dr.-Ing. Carsten Thomas (HTW Berlin) und Prof. Dr.-Ing. Volker Sommer (BHT). Der Forschungsschwerpunkt liegt auf der Erhöhung der Sicherheit hochautonomen Fahrens. Als Ansatz werden hierfür Verfahren für die Fahrzeug-zu-Fahrzeug Kommunikation erforscht.

Die Fahrzeuge sollen gegenseitig interne und externe Zuständen austauschen. Dadurch soll ein umfangreiches und zuverlässiges Situationsbewusstsein der einzelnen Systeme erreicht werden.

Die Lokalisierung durch Visuelle Odometrie kann die Position und Orientierung eines Fahrzeugs schätzen. Aus dieser Information ergibt sich auch die Trajektorie. Neben dem Austausch mit anderen Systemen, können die Daten intern genutzt werden. Beispielsweise kann Visuelle Odometrie das Front-End für Simultaneous Localization And Mapping (SLAM) stellen.

1.2 Ziel der Arbeit

Im Rahmen der Abschlussarbeit soll ein System implementiert werden, dass mittels Visueller Odometrie die Position und Orientierung (Pose) eines Fahrzeugs schätzt. Die Implementierung soll sich mit dem CARLA Simulator nutzen lassen und Odometriedaten innerhalb eines ROS2 Netzwerks zur Verfügung stellen können.

Die Schätzung der Pose soll ausschließlich auf Kameradaten basieren, es wird nicht auf andere Umgebungsdaten des Simulators zugegriffen.

Für eine Software Implementierung sollen zwei Kamera-Setups verglichen werden; ein Stereokamera Aufbau mit zwei RGB Kamerasensoren und ein RGB-D Setup bestehend aus einer RGB und ToF Kamera.

2. Grundlagen

2.1 Lokalisierungsverfahren

Lokalisieren bedeutet immer, eine Pose innerhalb einer Umgebung zu bestimmen. Bei einigen Verfahren ist die Umgebung bekannt weil eine Karte gegeben ist, bei anderen ist die Umgebung unbekannt.

Prinzipiell werden Lokalisierungsverfahren anhand der Aufgabenstellung in zwei Kategorien unterschieden. Zum einen gibt es globale Verfahren und zum anderen Position Tracking bzw lokale Verfahren. Die Assoziation global und lokal bezieht sich auf das Koordinatensystem bzw Frame zu dem Bezug genommen wird.

2.1.1 Globale Lokalisierung

Globale Lokalisierung schätzt die Pose im globalen Referenzsystem, d.h. in Bezug zum Koordinatenursprung der Umgebungskarte. Dabei müssen die Wahrscheinlichkeiten für mögliche Posen geschätzt werden. Ein bekanntes Beispiel für globale Lokalisierungsverfahren ist die Markov Lokalisation.

Eine Aufgabe, in dem die globalen Lokalisierung angewendet wird, ist die Ermittlung der Startposition. Denn autonome Agenten müssen sich häufig orientieren, nachdem sie durch externe Kräfte bewegt wurden. Globale Lokalisierung löst beispielsweise das Kidnapped-Robot Problem, bei dem der Agent plötzlich versetzt wurde oder bestimmt den Zustand nach dem Anschalten. Der Wake-Up eines Roboters zählt als ein Sonderfall des Kidnapped-Roboters, wobei dem Agenten hier i.d.R. bekannt ist, dass sich seine Position verändert hat.

Globalen Lokalisierungsverfahren müssen also in der Lage sein auch große Positionsänderungen kompensiert zu können. [1]

2.1.2 Position Tracking

Beim Position Tracking ist die Ausgangsposition in der Regel ungefähr bekannt und es sollen vor allem odometrische Fehler korrigiert werden. Der Referenzrahmen des position Tracking ist lokal und muss in einem weiteren Schritt in das globalen (Welt) Koordinatensystem transformiert werden.

Für das Position Tracking wird die Pose inkrementell, von Frame zu Frame, ermittelt. Daher führen zu grüße Änderungen zwischen zwei Zuständen zum Ausfall

und die Schätzung muss abgebrochen werden.

Visuelle Odometrie zählt zu den Position Tracking Verfahren. Die Startposition wird als Koordinatenursprung des lokalen Bezugssystems gesetzt. Die Odometrieschätzung der aktuellen Pose bezieht sich immer auf die vorherige Pose.

2.2 3-D Strukturen und Kamerapose aus 2-D Bildern

In den vergangenen 30 Jahren wurde viel Forschung im Bereich der Computer Vision als ein Teilgebiet der autonomen Robotik betrieben. Werden die Umgebungsinformationen teilweise oder ausschließlich aus Kameratasensoren erhalten, können geometrische Modelle aufgestellt werden um Informationen aus den Bildern zu gewinnen. Grundsätzlich soll die Kamera Pose aus Pixelinformationen hergeleitet werden. Eine Grundlage hierfür bildet die Korrespondenzanalyse, für die Merkmale aus den Bildern extrahiert werden müssen. Durch Bildkorrespondenzen kann eine Beziehung zwischen Bildern und zwischen Kamera und Umwelt hergestellt werden. Beim Umgang mit Bildkorrespondenzen werden Methoden der Projektiven- und Epipolaregeometrie herangezogen.

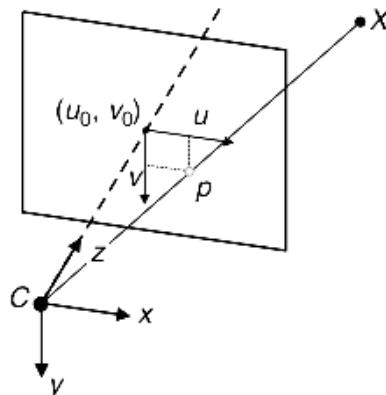


Abbildung 2.1: Abbildung einer 3-D Punktes X auf den 2-D Punkt p der Bildebene. (Scaramuzza, 2011 [2])

Transformationen aus der projektiven Geometrie können im Alltag regelmäßig auf Fotos beobachtet werden. Beispielsweise erscheint ein Kreis je nach Perspektive, aus der er abgebildet wurde, in einem Bild als Ellipse. Der Kreis wurde transformiert.

Durch Kameratasensoren werden Objekte aus dem 3-Dimensionalen Raum, der 'Welt', auf die 2-Dimensionalen Bildebene des Kamerabildes abgebildet. Dabei bleiben fast keine Eigenschaften der transformierten Objekte erhalten. Wie beim bereits aufgeführten Beispiel Kreis als Ellipse werden Objekte deformiert. Die einzige Eigenschaft, die erhalten bleibt, ist die Geradentreue. D.h. gerade Linien bleiben auch im Projektiven Raum immer gerade Linien.

2.2.1 Koordinatensysteme

Bekannt ist der Euklidische Raum und die Euklidische Geometrie als Beschreibung von Winkeln und Formen von Objekten. Meistens werden Punkte des euklidischen Raumes im kartesischen Koordinatensystem dargestellt. Ein Problem im Modell der euklidischer Geometrie sind parallele Linien. Als Sonderfall wird angenommen, das sich parallele Linien im Unendlichen schneiden. Obwohl es tatsächlich im Euklidischen Raum keine Punkte im Unendlichen gibt.

Der Projektive Raum ist die Transformation des Euklidischen Raums durch Einführung der Punkte im Unendlichen. Die Punkte werden dort als ideale Punkte oder Fernpunkte bezeichnet. Auch Räume sind in der geometrischen Betrachtung Objekte, die rotiert, verschoben oder gestreckt werden können.

$$T_{rotation}(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \quad (2.1)$$

Transformation Rotation für 2-D

$$T_{scale}(S) = \begin{pmatrix} S_x & 0 \\ 0 & S_y \end{pmatrix} \quad (2.2)$$

Transformation Verschiebung für 2-D

Transformationen sind Operationen die auf alle Objekte angewendet werden können, also auf den Raum ebenso wie auf den Punkt. Verschiebung und Rotation zählen zu den Lineartransformationen und lassen sich daher durch Matrizen beschreiben.

Homogene Koordinaten

Die Translation, eine wichtige Transformation die für Lokalisierung benötigt wird, ist nicht linear. Um Translation in Matritzenschreibweise darstellen zu können, muss von kartesischen Koordinaten (x, y) zu homogenen Koordinaten $(x, y, 1)$ übergegangen werden. Dabei wird ein Vektor um eine weitere Komponente $w = 1$ erweitert. Es handelt sich bei homogenen Koordinaten um eine andere Schreibweise, nicht um eine Veränderung der Dimension. Alle Punkte (kx, ky, k) sind gleich, für $k \neq 0$.

Eine Rücktransformation von homogenen Koordinaten zu kartesischen erfolgt durch division von (x, y) durch k .

Es wurde bereits erwähnt, dass im projektiven Raum sog. Fernpunkte eingeführt werden um Unendlichkeit einzuführen. Im inhomogenen Modell treffen sich zwei parallele Geraden im Fernpunkt inf der als m bezeichnet werden soll. m ist Teil der Geradengleichung aller inhomogenen Geraden $y = mx + d$.

In gleicher Weise kann m auch allen Ursprungsgeraden aller Ursprungsebenen zugewiesen werden. Daraus folgt

$$\begin{aligned} (x, y) &\rightarrow (x : y : 1) \\ (m) &\rightarrow (1 : m : 0) \\ (\text{inf}) &\rightarrow (0 : 1 : 0) \end{aligned} \quad (2.3)$$

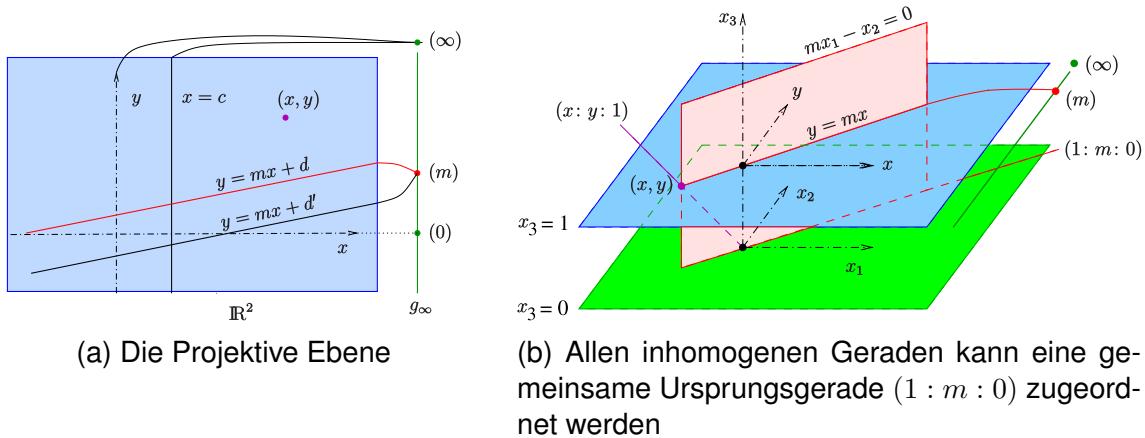


Abbildung 2.3: Beziehung zwischen inhomogenen und homogenen Koordinaten, (Ag2gaeh Wikimedia Creative Commons)

Durch dieses Schreibweise der Koordinaten wurde erreicht, dass es keine Ausnahmefälle mehr gibt. Bisherige Transformationen werden in homogenen Koordinaten geschrieben. Hiermit kann auch die Translation als lineare Abbildung beschrieben werden.

$$\begin{pmatrix} x' \\ y' \\ w \end{pmatrix} = \begin{pmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (2.4)$$

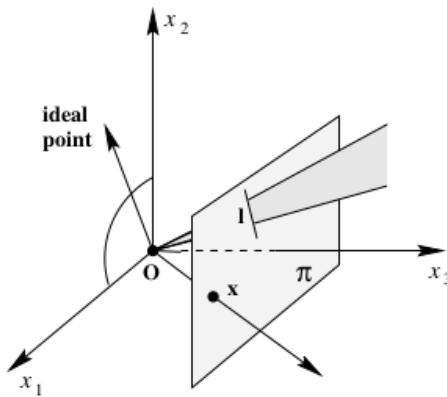
Verschiebung in Homogenen Koordinaten

$$\begin{pmatrix} x' \\ y' \\ w \end{pmatrix} = \begin{pmatrix} x\cos\theta & -y\sin\theta & 0 \\ x\sin\theta & y\cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (2.5)$$

Rotation in Homogenen Koordinaten

$$\begin{pmatrix} x' \\ y' \\ w \end{pmatrix} = \begin{pmatrix} 0 & 0 & a \\ 0 & 0 & b \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (2.6)$$

Translation in Homogenen Koordinaten

Abbildung 2.4: Punkte und Linien in \mathbb{P}^2 aus \mathbb{P}^3 (Hartley, 2003 [3])

Kameraprojektionen

Das Abbilden von 3-D auf 2-D ist eine Projektion die durch eine Zentralprojektion beschrieben wird. Die Strahlen von allen Punkten laufen durch das Projektionszentrum, in diesem Fall der Brennpunkt. Der Brennpunkt ist gleichzeitig das Kamerazentrum. Dabei schneiden die Strahlen die Bildebene, die vor der Kamera aufgespannt wird. Die Schnittpunkte sind die Bildpunkte. Dieses Modell entspricht dem Modell der Lochbildkamera. D.h. Fokus und Linsenstärke werden vernachlässigt.

In der projektiven Geometrie ist die Zentralprojektion eine Abbildung von \mathbb{P}^3 zu \mathbb{P}^2 . Ein Punkt in \mathbb{P}^3 kann als $(X, Y, Z, T)^T$ in homogenen Koordinaten geschrieben werden und das Projektionszentrum als $(0, 0, 0, 1)^T$. Um \mathbb{P}^3 auf \mathbb{P}^2 ab zu bilden kann die sogenannte Kameramatrix (oder Projektionsmatrix) P verwendet werden.[3] Es ergibt sich eine lineare Abbildung zwischen Kameraprojektion und einem Punkt im Raum als

$$\begin{pmatrix} x \\ y \\ w \end{pmatrix} = P_{3 \times 4} \begin{pmatrix} X \\ Y \\ Z \\ T \end{pmatrix} \quad (2.7)$$

2.2.2 Korrespondenzanalyse

Bildkorrespondenzen sind Merkmale in Bildern die das selbe Objekt in zwei unterschiedlichen Bildern beschreiben und als Ankerpunkte zwischen den Kamerapositionen dienen. Wenn eine Rekonstruktion von 3-D Punkten durchgeführt werden soll muss immer die Aufgabe der Bestimmung von Korrespondenzen gelöst werden.

Bei der Korrespondenzanalyse geht es ganz allgemein darum übereinstimmende Bildmerkmale in zwei Bildern zu finden. Handelt es sich um zwei Bilder einer Stereokamera, spezifiziert sich die Aufgabenstellung zur Stereoanalyse.

Die Korrespondenzanalyse ist aus dem Anwendungsgebiet der Bewegungsschätzung für Videostreaming hervorgegangen. Korrespondenzanalyse wird in der Videokodierung genutzt um zu Schätzen, in welchen Bereichen eines Bildes Bewegung stattgefunden hat. Für die Bereiche werden Bewegungsvektoren erstellt.

Bilder werden anschließend mit den Bewegungsvektoren kodiert und so müssen nur Teile des Bildes übertragen und aktualisiert werden. [4]

Bei SfM und Visueller Odometrie soll im Grunde genau das Gegenteil erreicht werden. Die Szene ist überwiegend statisch und die Bewegung der Kamera muss geschätzt werden. Schwierigkeiten, die bei der Korrespondenzanalyse auftreten, sind beispielsweise Regionen die nur im Bildfeld einer Kamera vorhanden sind, aufgrund von Geometrie oder Verdeckung, wiederkehrende Muster oder Oberflächen mit schwacher Textur.

Die Korrespondenzanalyse kann in pixelbasierte und merkmalsbasierte Verfahren unterschieden werden. Für das erstellen der Disparitätenkarte (ein Schritt zur Ermittlung der Tiefenwerte von Pixeln) wird eine pixelbasierte Stereoanalyse verwendet. Zum einen, da eine hohe Informationsdichte benötigt wird. Zum anderen kann aus der Kenntnis über die Geometrie des Stereoaufbaus die Suche auf einen eindimensionalen Suchraum beschränkt werden. Diese Aussage wird in der folgenden Betrachtung der Epipolargeometrie noch ersichtlich.

Für die Korrespondenzanalyse der Bewegungsschätzung werden merkmalsbasierte Verfahren angewendet.

2.2.3 Verbinden von zwei Ansichten durch Epipolargeometrie

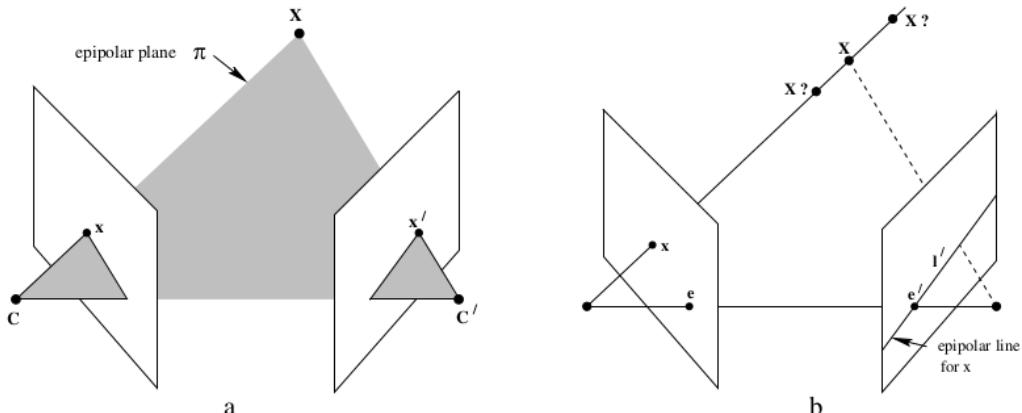


Abbildung 2.5: (a) Die Kameras sind gekennzeichnet als C und C' , vor Ihnen spannen sich die jeweiligen Bildebenen auf. (b) X muss auf seiner Epipolarlinie liegen. (Hartley, 2003 [3])

Die Epipolargeometrie besagt, dass die Kamerazentren C und C' , die Bildpunkte x und x' und die Raumpunkte koplanar sein müssen.

$$Cx \cdot (CC' \times Cx') = 0 \quad (2.8)$$

Diese Forderung wird durch die Fundamentalmatrix F abgebildet. Die Fundamentalmatrix wird als algebraische Repräsentation der Epipolargeometrie bezeichnet. Sie ist eine homogene 3×3 Matrix mit Rang(2), die nachfolgende Gleichung erfüllt [3]

$$x'^T F x = 0 \quad (2.9)$$

In Matrixschreibweise mit Homogenen Koordinaten lässt sich 2.9 schreiben als

$$\begin{bmatrix} x_{Ri} & y_{Ri} & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = 0 \quad (2.10)$$

Zwischen diesen koplanaren Punkten lässt sich eine Ebene aufspannen. Diese Ebene wird als Epipolarebene bezeichnet. Jede Kamera spannt außerdem eine Bildebene in Blickrichtung auf. Die Epipolarebene schneidet die Bildebenen. Aus Sicht der ersten Kameraposition befindet sich X auf einem Strahl der vom Projektionszentrum durch den Bildpunkt x der Bildebene läuft. Dieser Strahl entspricht der Epipolarlinie im zweiten Bild, die durch die Epipolarebene geschnitten wird. Um einen Punkt eindeutig bestimmen zu können, lässt sich folgende Beschränkung einführen: x' muss auf seiner Epipolarlinie l' liegen um eine Projektion des Gleichen Punktes X zu sein wie x .

$$x \mapsto l' \quad (2.11)$$

Die Epipolarlinie ist die Überschneidung zwischen Epipolar- und Bildebene. Der Epipol ist der Punkt in der Bildebene in welchem diese die Basislinie schneidet. Die Baseline ist die Gerade die zwischen den Kamerazentren C und C' gezogen werden kann.

2.2.4 Sonderfall Stereo Vision

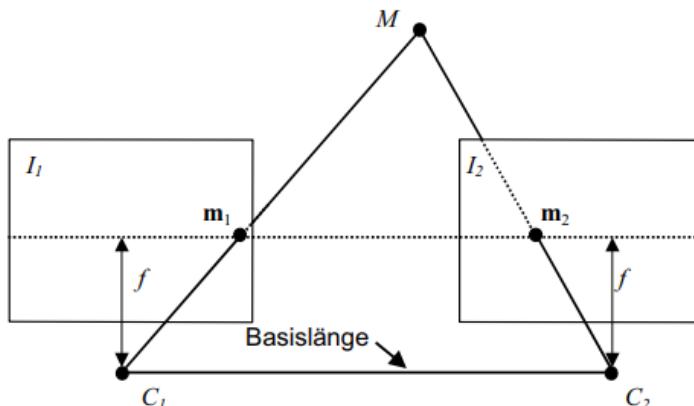


Abbildung 2.6: Sind die Kamerazentren exakt parallel, verlaufen die Epipolarlinien parallel zur Baseline, (Schreer, 2005 [4])

Bei einem Stereo Kamera Aufbau wird von einem Sonderfall der Epipolargeometrie nutzen gemacht. Die Kamerazentren sind dabei exakt parallel zueinander ausgerichtet. Das führt dazu, dass die Epipole ins Unendlichen verschoben werden. Sind C und C' parallel, verlaufen die Epipolarlinien l und l' parallel zur Baseline. Das besondere daran ist, das nun die Epipolargeometrie für jeden Punkt

x' sofort bekannt ist. Die Forderung für x' ist $x' \mapsto l'$ und genauso gilt $x \mapsto l$. l und l' liegen nun auf gleicher Höhe. Damit ist l' aus l bekannt.

Die Bestimmung von x' wird auf diese Art wesentlich beschleunigt. Dieser Umstand lässt sich nicht für aufeinanderfolgende Bilder I und I' ausnutzen. Stattdessen wird die Geometrie beim erstellen der Tiefenkarte verwendet. Dazu wird das aktuelle Bildpaar, welches zeitgleich aus der gleichen Fahrzeugpose aufgenommen untersucht.

Prinzipiell wird dabei für jedes Pixel wie folgt vorgegangen:

Zuerst wird die Epipolarlinie gesucht.

Anschließend wird die Linie nach der besten Übereinstimmung abgesucht.

Schließlich wird die Tiefe aus der Verschiebung berechnet

$$Z = \frac{bf}{d} \quad (2.12)$$

2.2.5 Structure From Motion - SfM

Um die Pose einer Kamera bestimmen zu können, muss die Positionen der Bildpunkte in der Umgebung bekannt sein. Ein Verfahren zur Rekonstruktion von 3-D Objekten aus Bildern ist als Structure from Motion (SfM) bekannt. Visuelle Odometrie zählt als Sonderfall von SfM.

SfM ist rechenaufwendig und wird überwiegend offline betrieben. Anders als Visuelle Odometrie, die immer online betrieben wird. Offline bedeutet hier, dass eine Menge an Bildern (geordnet oder ungeordnet) bereits vollständig vorliegt. Viele Algorithmen die heute für Visuelle Odometrie oder SLAM eingesetzt wurden sind bereits seit den 80er Jahren bekannt wo sie für SfM entwickelt wurden. [5]

SfM hat viele interessante Anwendungsgebiete, wie in der Geoinformatik und Informationsgewinnung aus offline Bilddaten. Beispielsweise veröffentlichten Sameer Agarwal et. al. 2009 das Paper Building Rome in a Day indem sie zeigten wie eine 3-D-Rekonstruktion der Stadt Rom in 21 Stunden angefertigt wurde. Als Eingangsdaten verwendeten sie ausschließlich die Bilder, die bei der Suche für das Schlagwort Rom von der Photo-Community Plattform Flickr ausgegeben wurden.[6]

2.2.6 Simultaneous Localization and Mapping - SLAM

Eine andere Kategorie von SfM ist Simultaneous Localization and Mapping (SLAM). Hier wird neben der Lokalisierung gleichzeitig eine Umgebungskarte erstellt. Die Karte ergibt sich aus den 3-D Punkten, die aus den Merkmalen der Bildpaare rekonstruiert wurden. Während diese Punkt für die Visuelle Odometrie nach der Schätzung verworfen werden, fasst SLAM sie zu 3-D Punktwolken zusammen.

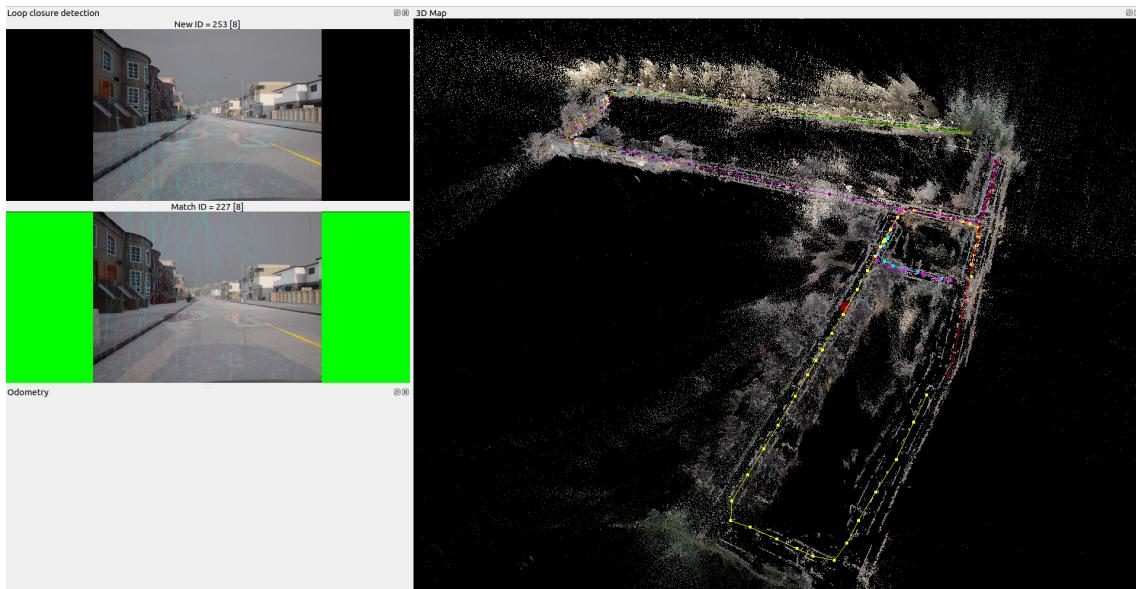


Abbildung 2.7: Screenshot einer SLAM Karte einer CARLA Testfahrt. Die Karte wurde mit dem Visual Odometrie System der Abschlussarbeit durch Erweiterung mit RTAB SLAM erzeugt.

Diese Punktwolken bilden eine Karte der Umgebung.

Im Gegensatz zu SfM wird SLAM i.d.R. online also in Real-Time betrieben. Bei VO und SLAM werden die Eingangsbilder sequentiell akquiriert und die Pose von einem Zeitschritt zum nächsten geschätzt.

Auch SLAM schätzt die zurückgelegte Trajektorie. Die erstellte Umgebungskarte wird genutzt um ein sog. Loop-Closure zu entdecken. Loop-Closure bedeutet, dass zuvor detektierte Landmarken wiedererkannt werden. Der Agent kann also feststellen, dass er sich zuvor bereits in der Nähe der aktuellen Position befunden hat.

Die bisherigen Schätzungen werden bei SLAM als Graph gespeichert. Die Posen sind Knoten. So entsteht ein Pose Graph. Nach erneutem Besuchen einer Landmarke kann eine Pose Graph Optimierung ausgeführt werden, bei dem die Positionen aller Knoten rekursiv an die korrigierte Schätzung angepasst werden.

2.2.7 Verschiedene Ansätze für Visuelle Odometrie

Das Konzept der Visuellen Odometrie wurde zuerst 1980 als Lokalisierung für Mars Rover eingeführt und erfolgreich umgesetzt. Ein Großteil der Anfänglichen Forschung kam daher aus der Raumfahrt. Die Motivation Visuelle Odometrie für Marsrover einzusetzen liegt vor allem in der Abwesenheit von GNSS und dem extrem unebenen Untergrund. Anwendungen für VO und SLAM Implementierungen lassen sich heute in den unterschiedlichsten Bereichen finden und haben einen festen Platz in der Erforschung autonomer Fahrzeuge.

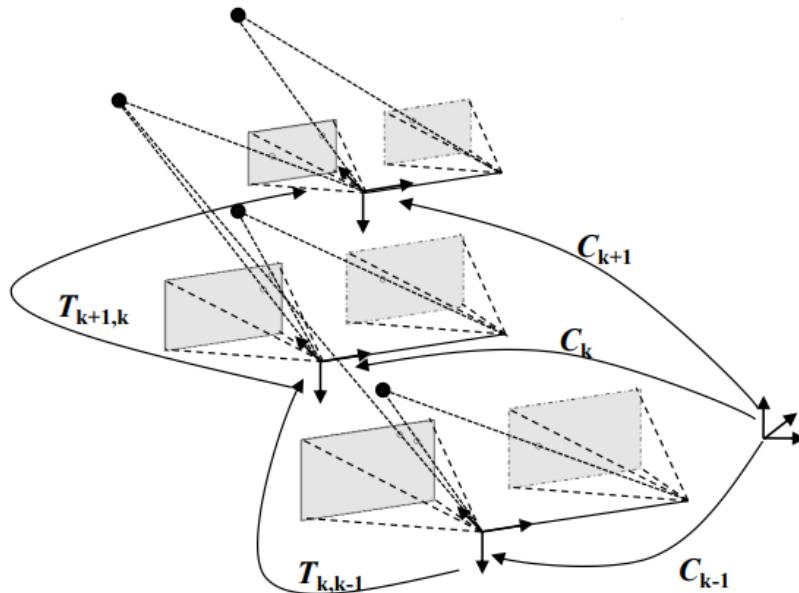


Abbildung 2.8: Illustration der Visuelle Odometrie Problemstellung. Ein Kamerakoordinatensystem C_k bewegt sich in Bezug zum Weltkoordinatensystem. Zu diskreten Zeitpunkten k soll die Transformationsmatrix T_k zwischen den Kameraposen C_k geschätzt werden. (Scaramuzza, 2011 [2])

Um die Problemstellung der Visuellen Odometrie zu erläutern wird angenommen, dass sich zwei Kameras als Stereokamerasystem auf einem autonomen Fahrzeug befinden. Es werden zwei Koordinatensysteme betrachtet; das lokale Kamerakoordinatensystem C und das globale Weltkoordinatensystem. Dabei ist C der gemeinsame Koordinatenursprung beider Kameras und befindet sich im Brennpunkt der linken Kamera.

Um die Kameraposen über einen Verlauf, mit diskreten Zeitabständen k , zu schätzen, wird zunächst die Transformation $T_{k,k-1}$ zwischen der vorherigen Position C_{k-1} und der aktuellen Pose C_k geschätzt.

Durch addieren von Transformation $T_{k,k-1}$ auf die vorherige Pose C_{k-1} ergibt sich so die aktuelle geschätzte Kameraposition C_k .

Daraus folgt, dass ein Aneinanderreihen aller Transformationen aus der Initialpose $k = 0$, heraus die geschätzte Trajektorie der Kamera beschreibt.

$$T_{k,k-1} = \begin{pmatrix} R_{k,k-1} & t_{k,-1} \\ 0 & 1 \end{pmatrix} \quad (2.13)$$

Wie die Gleichung 5.4 zeigt, setzt sich die Transformationsmatrix $T \in \mathbb{R}^{4 \times 4}$ aus einer Rotationsmatrix $R \in \mathbb{R}^{3 \times 3}$ und einem Translationsvektor $t \in \mathbb{R}^{3 \times 1}$ zusammen. T ist also der Ergebnisvektor einer einzelnen Odometrieschätzung.

Der zurückgelegte Pfad wird inkrementell geschätzt. Anders als bei SLAM gibt es keine Korrektur über einen längeren Zeitraum, weshalb die Drift über die Zeit akkumuliert. Wird VO als Front-End für (V)-SLAM implementiert kann die Drift durch Loop-Closure korrigiert werden. Eine andere Möglichkeit der Korrektur, ist über Sensorfusion mit anderen Sensoren. Im Laufe des letzten Jahrzehnts sind dabei überwiegend Forschungsergebnisse zum Einsatz von Visual-Inertial-Odometry

(VIO) veröffentlicht worden.

Im Allgemeinen zeigt VO weniger Drift als klassische odometrische Messverfahren durch Radsensoren, Drehgeber oder Gyroskop.[2]

Für einen VO Algorithmus kann zwischen Feature Based, Appearance Based und Lernenden Verfahren unterschieden werden.

Feature-based Algorithmen

Beim Feature based oder merkmalsbasierten Ansatz werden markante Merkmale gefunden und von Frame zu Frame verfolgt. Feature Based Algorithmen durchlaufen immer zwei Phasen: Zuerst werden Merkmale in aufeinanderfolgenden Bildern gefunden. Anschließend werden die Merkmale abgeglichen um übereinstimmende Merkmale zu bestimmen.

Jeder der Phasen hat seine Herausforderungen. Für eine zuverlässige Merkmalsdetektion werden Merkmalsdetektoren berechnet.

Der zweite Schritt ist für das Problem der Korrespondenzfindung bekannt. Veränderungen in den Bildern wie Bewegung, Verdeckung oder Veränderung der Skalierung machen die Korrespondenzanalyse zu einer anspruchsvollen Aufgabe.

Wurden Korrespondenzen zwischen den Bildern bestimmt, wird die Pose aus der Minimierung des Reprojektionsfehlers zwischen den Bildern bestimmt.

Appearance-based Algorithmen

Appearance-based Methoden kommen gänzlich ohne Merkmalsdetektion und Matching aus. Stattdessen werden die Intensitätswerte aller Pixel analysiert und zwischen zwei Bildern auf das gesamte Bild untersucht. Anstelle der Minimierung des Reprojektionsfehlers wird ein sog. Photometrische Fehler minimiert.

Appearance-based Methoden liefern aufgrund der großen Informationsmenge die aus den Bildern genutzt wird sehr gute Ergebnisse, sind aber extrem rechenaufwendig. Daher wurden sie für mobile Systeme erst vermehrt im vergangenen Jahren, aufgrund leistungsfähiger Hardware, untersucht.

Lernende Verfahren

Die Erforschung von Visuelle Odometrie mit lernenden Verfahren ist vor allem für monokulare Visuelle Odometrie von Bedeutung. Hier fehlt die Skaleninformation für die Berechnung der 3-D Punkte bzw der Tiefeninformation und muss zusätzlich geschätzt werden. Ein Möglicher Deep Learning Ansatz wäre Beispielsweise, ein Neuronales Netz zuerst mit Videos eines Stereo VO Systems zu trainieren um es dann in Mono Setups einzusetzen. Versucht wurde dies bei D3VO der Technischen Universität München. [7]

Bisher erzielen traditionelle Methoden nach wie vor bessere Ergebnisse, als Deep Learning Ansätze. Es wird aktuell viel auf diesem Gebiet geforscht. Sobald die richtige Methodik gefunden wird könnten die monokularen VO Systeme mit neuronalen Netzen die traditionellen Stereo Systeme ablösen, da sie weniger Hardware benötigten.

3. Extraktion von Bildmerkmalen

Das Extrahieren von Bildmerkmalen ist zentraler Bestandteil aller Feature-based Algorithmen. Es muss festgelegt werden, welcher Typ von Merkmal gefunden werden soll, ein Suchalgorithmus ausgewählt werden und schließlich Überlegt werden durch welchen Deskriptor die Merkmale beschrieben werden sollen.

3.1 Auffinden von Merkmalen

3.1.1 Merkmale Beschreiben durch Nachbarschaftsoperationen

Merkmale sind lokale Punkte im Bild die durch die Pixel in ihrer Nachbarschaft charakterisiert sind. Gute Merkmale sind möglichst eindeutig bestimmbar und lassen sich nicht mit anderen Stellen im Bild verwechseln. Anwendungen wie Structure from Motion setzen Merkmale voraus, die sich auch nach Veränderungen des Bilds wie beispielsweise durch Translation, Rotation oder Lichtverhältnisse wiedererkennen lassen. Um diesen Anforderungen gerecht zu werden, eignen sich die Interest-Operatoren (auch Interest-Points, Corner).

Es sollen zunächst einige allgemeine Grundlagen zur Merkmalsdetektion betrachtet werden. Es wird einführend auf Kantendetektion eingegangen um von dort zu den Interest-Operatoren zu kommen.

Eine einfache Operation durch die Merkmale gefunden werden können, ist die Faltung.

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t - \tau)d\tau = \int_{-\infty}^{\infty} f(t - \tau)g(\tau)d\tau \quad (3.1)$$

Dabei ist g der Faltungskern (auch Kernel oder Filter) und bestimmt welche Operation auf das Bild angewandt wird. Durch den Kern kann ein Bild beispielsweise geglättet, schärfer gemacht werden oder Kanten extrahiert werden.

Faltungskerne lassen sich außerdem als Approximation verwenden, wenn eine Ableitung des Bildes gebildet werden soll. Die Ableitung eines Bildes ist essentiell für die Bildverarbeitung, da sie den Gradient bildet. Durch die Bestimmung der Gradienten können Kanten und ihre Eigenschaften beschrieben werden. Der Gradientenvektor ist definiert als

$$\nabla f = \left[\frac{\delta f}{\delta x}, \frac{\delta f}{\delta y} \right] \quad (3.2)$$

Der Gradient eines Bildes zeigt in die Richtung, in der es die größte Veränderung der Intensitätswerte der Pixel gab. Daher lässt sich die Richtung der größten Veränderung bestimmen durch

$$\theta = \tan^{-1} \left(\frac{\delta f}{\delta y} / \frac{\delta f}{\delta x} \right) \quad (3.3)$$

Die Richtung verläuft immer Senkrecht zur Kante. Die Stärke des Anstiegs und damit die Stärke der Kante und ist gegeben durch den Betrag des Gradienten

$$||\nabla f|| = \sqrt{\left(\frac{\delta f}{\delta x}\right)^2 + \left(\frac{\delta f}{\delta y}\right)^2} \quad (3.4)$$

Für die Bildverarbeitung können die Partiellen Ableitungen praktisch berechnet werden als

$$\begin{aligned} \delta x &= f(x+1, y) - f(x, y) \\ \delta y &= f(x, y+1) - f(x, y) \end{aligned} \quad (3.5)$$

Beispiele für bekannte Kanten Operatoren sind der Sobel Operator, Prewitt, Roberts, Laplacian of Gaussian (LoG) und Canny.

$$\rho(\tau) = K \int_{-\infty}^{\infty} x(t)m(t+\tau)dt \quad (3.6)$$

Die gesuchten Interest-Operatoren entstehen durch das aufeinandertreffen zweier Kanten. Das bedeutet in der Nachbarschaft dieses Punktes existieren zwei starke und unterschiedliche Gradientenansteige. Die einfachste Art Interest-Points zu detektieren ist durch Korrelation.

3.1.2 Algorithmen zum Auffinden von Interest-Operatoren

Für die Lokalisierung werden Detektoren verwendet, die nach Interest-Operatoren im Bild suchen. Interest-Punkte haben gegenüber Punkten und Kanten die entscheidende Eigenschaft invariant gegenüber Veränderungen in Rotation, Skalierung und Lichtverhältnissen zu sein.

Moravec-Operator

Der Moravec-Operator wurde 1977 veröffentlicht. Berechnet die mittlere quadratische Gradientensumme und bestimmt Ecken anhand eines Schwellwertes. Der Moravec-Operator eignet sich nicht im Kontext von Lokalisierung, da er nicht Rotationsinvariant ist.

Harris Corner Detektor

Der Harris Corner Detektor wurde 1988 basierend auf dem Moravec-Operator veröffentlicht. Der wesentliche Unterschied ist, dass Harris eine Rotations- und

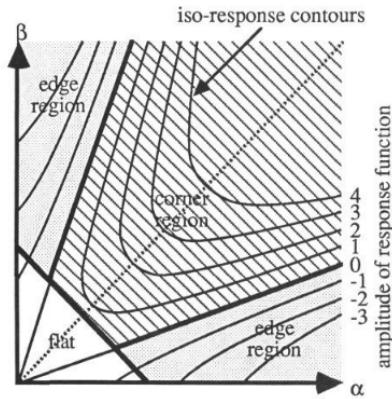


Abbildung 3.1: Klassifikation der Autokorrelation in Edge, Flat und Corner Region. α und β entsprechen λ_1 und λ_2 (Harris, 1988 [8])

Skalierungsinvarianz durch Integration von Autokorrelation erreicht. Auf einen Bildpunkt angewandt kann durch eine Funktion $E(u, v)$ die Veränderung der Intensitätswerte für eine Verschiebung eines Punktes in jede Richtungen gefunden werden.

$$E(u, v) = \sum_{x,y} \underbrace{w(x, y)}_{\text{Window Fkt}} \frac{\underbrace{[I(x + u, y + v) - I(x, y)]^2}_{\text{verschobene Intensität}}}{\underbrace{[I(x, y)]^2}_{\text{Original Intensität}}} \quad (3.7)$$

3.1: Formel des Harris-Corner Detektors

Für die Detektion von Interest-Operatoren muss die Funktion maximiert werden. Verwendung der Taylor Approximation 1. Grades führt zu nachfolgenden Schritten.

$$[!ht] f(x + u, y + v) \approx f(x, y) + u f_x(x, y) + v f_y(x, y) \quad (3.8)$$

$$\begin{aligned} E(u, v) &= \sum [I(x + u, y + v) - I(x, y)]^2 \\ &= \sum [I(x, y) + u I_x + v I_y - I(x, y)]^2 \\ &= \sum u^2 I_x^2 + 2uv I_x I_y - v^2 I_y^2 \\ &= \sum [u \ v] \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \\ &= [u \ v] \left(\sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \right) \begin{bmatrix} u \\ v \end{bmatrix} \end{aligned} \quad (3.9)$$

Für kleine Verschiebungen ergibt sich die bilineare Approximation

$$E(u, v) \cong [u, v] M \begin{bmatrix} u \\ v \end{bmatrix} \quad (3.10)$$

Mit M als

$$M = \sum_{x,y} w(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \quad (3.11)$$

Der zweite Part des Harris Operators ist die Score Funktion.

$$R = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2 = \det(M) - k(\text{trace}(M))^2 \quad (3.12)$$

Durch den Score wird überprüft, ob das Fenster einen Interest-Operator enthält.

Shi-Tomasi Detektor (GFTT)

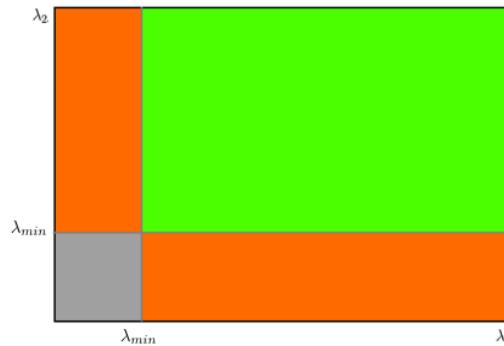


Abbildung 3.2: Ein Merkmal wird bei GFTT durch eine Min-Funktion bestimmt. Nur wenn λ_1 und λ_2 über dem Grenzwert λ_{min} liegen, handelt es sich um einen Interest-Operator (grüner Bereich) (OpenCV Docs GFTT, 2022)

Das erste mal würde dieser Detektor 1994 unter dem Titel *Good Features To Track* in einem Paper veröffentlicht. Daher wird er sowohl als Shi-Tomasi als auch GFTT Detektor bezeichnet. Dieser Detektor ist gleich dem Harris Detektor mit Ausnahme einer Änderung im Scoring. Statt das Scoring über die Eigenwerten zu berechnen, wurde vorgeschlagen, die Score direkt aus einer Min-Funktion der Eigenwerte zu setzen.

$$R = \min(\lambda_1, \lambda_2) \quad (3.13)$$

Dadurch kann die ursprünglich eingeführte Funktion weggelassen werden.

FAST

Features from Accelerated Segment Test (FAST) wurde 2006 veröffentlicht. FAST ist für seine Geschwindigkeit bekannt. Anwendungen wie SLAM haben sehr hohe Anforderungen an die Laufzeit der Merkmalsdetektoren. FAST hat eine kürzere Rechenzeit als die vorangegangenen Detektoren und wurde speziell für den Einsatz in zeitkritischen Anwendungen entwickelt.

FAST legt einen Kreis von 16 Pixel um ein interessantes Pixel q . Der Kreis wird durchnummeriert und der Intensitätswert I_q von q erfasst. Es wird verglichen ob im Kreis N zusammenhängende Pixel heller als $I_q + thresh$ oder dunkler als $I_q - thresh$ sind. Wobei $thresh$ ein Grenzwert ist.

Nun werden nicht alle 16 Pixel untersucht. FAST schlägt einen High-Speed-Test vor bei dem nur 4 Pixel getestet werden. Zuerst werden die Intensitäten von 1 und 9 gegen den Grenzwert getestet, dann 5 und 13. Ist p ein Interest-Punkt müssen

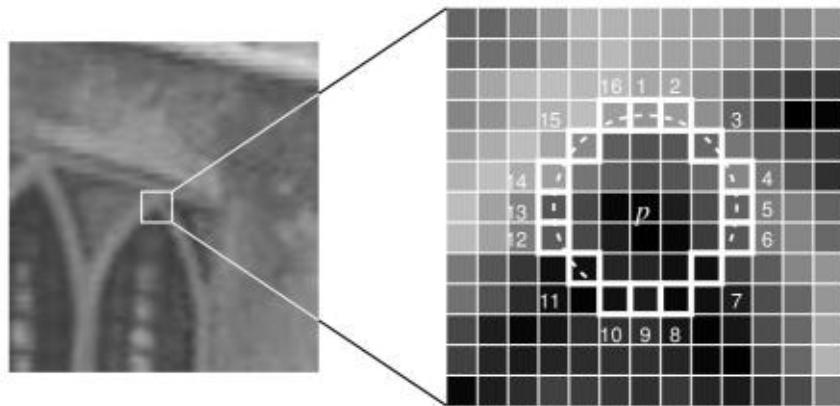


Abbildung 3.3: FAST Detektor Performance Test, (E. Rosten and R. Porter and T. Drummond, 2010 [9])

mindestens 3 dieser Pixel heller als $I_q + \text{thresh}$ oder dunkler als $I_q - \text{thresh}$ sein.

Das Verfahren ist tatsächlich sehr schnell, weißt aber einige Mängel auf. Die Empfehlung ist eine Kombination mit einem Machine Learning Ansatz, dadurch können die Nachteile gut behoben werden. [10] Für den Systementwurf wurden generell nur direkte Verfahren ohne Machine Learning ausgewählt und anstatt des FAST Detektors daher der GFTT (Shi-Tomasi Detektor) Detektor eingesetzt.

3.2 Matching

Beim Feature Matching wird festgestellt, ob es sich bei Merkmalen die in zwei verschiedenen Bildern gefunden wurden um das selbe Merkmal handelt. Hierzu wird für jedes Merkmal des ersten Bildes in dessen ursprünglichen Position im zweiten Bild gesucht. Wie groß die Suchumgebung um die ursprüngliche Position herum ist, hängt von der gewählten Abstandsmetrik des Suchalgorithmus ab.

Generell hängt die Leistung der Matchingverfahren von den Eigenschaften der zugrunde liegenden Merkmale als auch von der Wahl der zugeordneten Bilddeskriptoren ab. Es ist daher wichtig die geeigneten Detektoren und Deskriptoren auszuwählen. Beispielsweise wäre für ein Mikroskop Bild einer organischen Struktur ein Blob-Detektor wahrscheinlich am besten geeignet, während in einer Urbanen Umgebung mit menschengemachten, geometrischen Strukturen ein Interest-Operator wesentlich erfolgversprechender ist.

3.2.1 Übereinstimmungen finden

Zwischen zwei Bildern sollen gemeinsame Merkmale gefunden werden. Aus gefundenen Merkmalen besitzt jedes Bild eine eigene Menge von Merkmalsdeskriptoren. Jeder Deskriptor befindet sich an einer bestimmten Stelle im Bild. Für das Matching wird ein Deskriptor des ersten Bildes nicht mit der gesamten Menge an Deskriptoren des zweiten Bildes verglichen. Stattdessen wird um die Position, an der sich der Deskriptors im ersten Bild befindet, eine Region of Interest bzw Suchfenster gebildet. Es werden dann nur Deskriptoren des zweiten Bildes für den Abgleich herangezogen, die innerhalb dieser Region auftreten. Selbst beim, von OpenCV implementierten, Brute Force Matching wird in einem bestimmten Abstand zum gesuchten Deskriptor gesucht.

$$\begin{aligned}
 d_{euclidean}(p, q) &= \|q - p\|_2 \\
 &= \sqrt{(q_1 - p_1)^2 + \dots + (q_n - p_n)^2} \\
 &= \sqrt{\sum_{i=1}^n (q_i - p_i)^2}
 \end{aligned} \tag{3.14}$$

Die Größe des Suchfensters wird über den Abstand angegeben. Bei klassischen Deskriptoren wie SIFT und SURF wird im Euklidischen Abstand gesucht. Bei Binär-Deskriptoren wie ORB und BRISK sollte der Hamming-Abstand verwendet werden.

$$d_{hamming}(q, p) = \sum_{i=0}^{n-1} (q_i \oplus p_i) \tag{3.15}$$

Dei Begründung liegt in der unterschiedlichen Beschaffenheit der Deskriptoren. Bei SIFT repräsentieren sie Histogramme von gerichteten Gradienten. Bei ORB handelt es sich um Binärwörter die durch den XOR-Operator verglichen werden können wie in Gleichung 3.15 zu sehen.

3.2.2 Matching Ergebnisse Filtern

Die Matches die durch die Suche bestimmt wurden sind teilweise nur sich stark ähnelnde Merkmale und keine echten Übereinstimmungen. Die Ergebnisse sind also von unterschiedlicher Qualität. Sie müssen daher noch gefiltert werden. Zum einen wird so bei den folgenden Schritten Zeit gespart. Zum anderen handelt es sich im Grunde um Ausreißer die ohne Filterung zu Fehlern in nachfolgenden Schätzungen führen werden.

Um die Ergebnisse zu Filtern gibt es drei Unterschiedliche Ansätze:

Lowes Distance Ratio Test Der Test überprüft ob es in der direkten Nachbarschaft, weitere Matches gibt. Falls ja und das Match hat einen Ratio nahe eins, sind das Match und sein Nachbar von gleicher Qualität. Bei dem Ausgewählten Match handelt es sich dann mit 50% Wahrscheinlichkeit um einen Ausreißer. Daher werden beide Matches verworfen. Der Ratio unter dem die Merkmale liegen müssen ist variabel und wird über einen threshold and die Funktion übergeben. [11]

Cross Check Beim Cross-Check werden die Resultate der beiden Bilder darauf verglichen, ob das beste Match in Set A auch das beste Match in Set B war und umgekehrt. Cross Checking kann beim Brute Force Matcher verwendet werden.

Geometrischer Test Ausgleichsverfahren basierend auf der Methode der kleinsten Quadrate wie beispielsweise RANSAC.

3.2.3 Stereo Matching

Der Abgleich von Stereokamera Bildpaaren unterscheidet sich vom Matching in normalen Bildsequenzen. Stereo Bildpaare wurden zum gleichen Zeitpunkt aufgenommen und sind so ausgerichtet, dass sich die Merkmale bei beiden Bildern in der gleichen Reihe befinden.

Prinzipiell wird beim Stereo Matching für jedes Pixel wie folgt vorgegangen:

Zuerst wird die Epipolarlinie gesucht.

Anschließend wird die Linie nach der besten Übereinstimmung abgesucht.

Schließlich wird die Tiefe aus der Verschiebung berechnet

$$Z = \frac{bf}{d} \quad (3.16)$$

3.2.4 Semi Global Block Matching

Beim Stereo Block Matching wird nicht pixelweise verglichen, sondern kleine Regionen. Ein Fenster wird die Epipolarlinie des rechten Bildes entlang geschoben und der Inhalt mit der Ausgewählten Region im linken Bild verglichen.

Die Fenstergröße wirkt sich direkt auf die Disparitätenkarte aus. Sie beschreibt die Verschiebung zwischen den beiden Bildern und enthält noch nicht die gesamte Tiefeninformation. Tiefe wird mit Hilfe der Disparität berechnet, wie gezeigt in Gl. 3.16. Für kleine Fenstergrößen enthält die Disparitätenkarte mehr Details aber

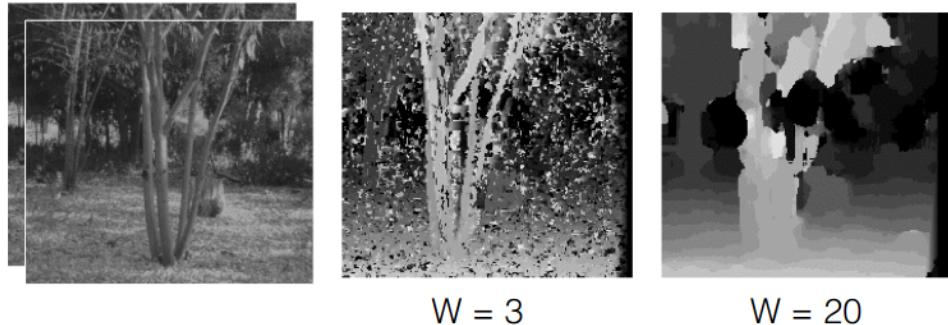


Abbildung 3.4: Effekt der Fenstergröße, (Kris Kitani - Carnegie Mellon University, 2020)

auch mehr Rauschen. Für große Fenster ist die Karte glatter und hat weniger Details.

Generell beschleunigt Block Matching die Erstellung der Disparitätenkarte da weniger Vergleiche vorgenommen werden müssen. In Regionen ohne Texturen oder bei Bildern in denen sich Muster oft wiederholen kann das Verfahren fehlschlagen. Teile der Karte sind dann Schwarz und ohne Information.

3.3 Deskriptoren

Für gefundene Merkmale werden Deskriptoren berechnen. Sie sind sozusagen der Fingerabdruck von jedem Merkmal. Mithilfe des Deskriptors sollen Merkmale eindeutig und zuverlässig wiedererkannt werden können. Vor allem auch dann, wenn sich die Abgebildete Szene zwischen zwei Bildern verändert hat. Ein Interest-Operator bleibt als solcher erkennbar, auch wenn die Region gedreht wird. Schwieriger verhält es sich mit der Skalierung.

Mit dem Harris-Detektor können bei einer großen Änderung der Skalierung der Interest-Punkt nicht mehr wiedergefunden werden. Die ersten skalierungsinvarianten Detektoren kamen mit SURF (2006) und SIFT (1999). SIFT (Scale-Invariant Feature Transform) liefert sehr gut Resultate in Verbindung mit Lokalisierungsaufgaben. Allerdings braucht der Deskriptor viel Rechenleistung und ist zu langsam für Echtzeitanforderungen. ORB ist eine weit verbreitete, wesentlich weniger Rechenaufwendige Alternative. Für den Systementwurf wurden zu Beginn beide Algorithmen betrachtet. Zusammen mit der Simulationsumgebung verbrauchten die Visuelle Odometrie Software mit SURF zu viele Ressourcen. Der Wechsel zu ORB brachte eine wesentliche Entlastung der CPU.

Diese Beobachtung lässt sich einfach durch eine Kategorisierung der Berechnungsmethoden der Algorithmen erklären.

SIFT gehört zu der Klasse der *Histogram of Oriented Gradients* (HOG) Deskriptoren während ORB zu den *Binären* Deskriptoren zählt. Ein Deskriptor hat immer eine Beschreibung für einen interessanten Pixel und seiner unmittelbare Nachbarschaft sowie eine Distanzfunktion mit der die Ähnlichkeit zwischen zwei Deskriptoren (des gleichen Typs) bestimmt wird.

3.3.1 Histogram Deskriptoren

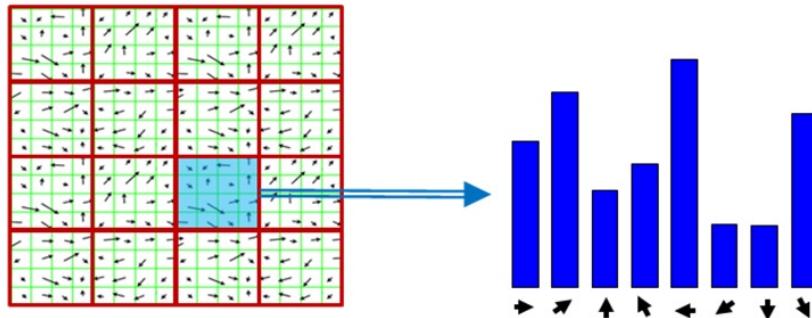


Abbildung 3.5: Berechnen der Histogramme für die Ausrichtung der Gradienten (Lowe, 1999 [12])

Histograms of Oriented Gradients (HOG) Deskriptoren die in der Praxis häufig eingesetzt werden sind SIFT [12] und SURF[13]. Im folgenden werden die Rechenschritte für SIFT beispielhaft erläutert.

Bei dieser Art von Deskriptor wird ein Region um das Merkmal, beispielsweise 16x16 Pixel, genommen. Die Region wird wiederum in kleinere Subregionen unterteilt, beispielsweise 4x4. Für alle Pixel wird die Orientierung berechnet und anschließend der Gradient für die Subregion. Daraufhin kann ein Histogram für die Subregionen erstellt werden.

Durch Aneinanderreihen der Histogramme ergibt sich ein Merkmalsvektor. Für das Beispiel waren es 16 Subregionen mit 8 Histogrammen per Subregion, also ein Merkmalsvektor der mit 128 (16×8) Elementen.

3.3.2 Binäre Deskriptoren

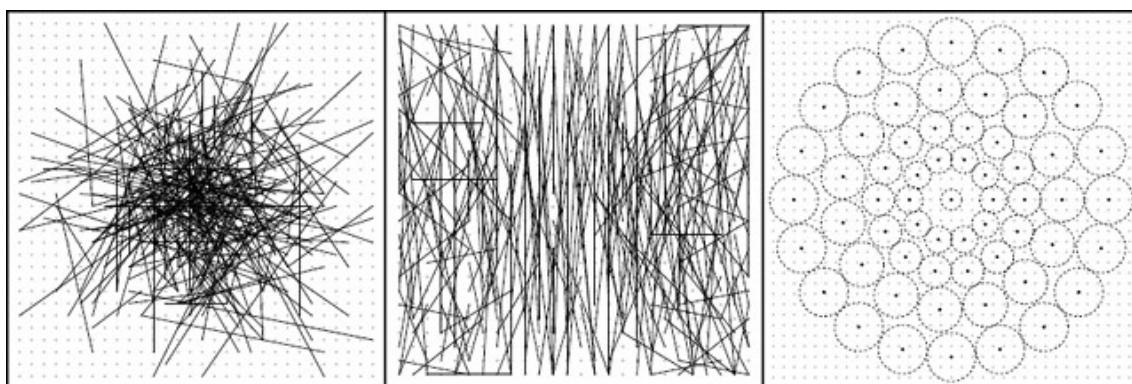


Abbildung 3.6: Deskriptor Muster, von Links nach Rechts: BRIEF, ORB, BRISK. (Heinly, 2012 [14])

Anstatt, wie bei HOG Deskriptoren, die Gradienten aller Pixel zu berechnen, werden bei binären Deskriptoren Intensitätswerte verglichen. Das Ergebnis des Vergleichs sind Binärwörter. Diese können direkt durch den XOR-Operator abgeglichen werden, was das Matching deutlich beschleunigt.

Beispiele für bekannte binäre Deskriptoren sind BRIEF [15], ORB [16] und BRISK[17]. Binärdeskriptoren können anhand von drei Attributen verglichen werden.

Das Abtastmuster Ein Muster das auf die Region um den Deskriptor gelegt wird. Es werden nur Punkte die auf dem Muster liegen untersucht.

Ausrichtungskompensation Eine Methode um die Orientierung des Merkmals zum bestimmen und es zu drehen um Rotation des Merkmals zwischen den Bildern auszugleichen.

Stichprobenpaare Auswahl von Punkten die verglichen wurden welche im Binärwort eingetragen werden sollen.

Es müssen nicht alle diese Attribute in binären Deskriptor vorhanden sein, sie helfen lediglich bei der Klassifizierung. Wie auch zuvor bei HOG Deskriptoren wird eine Region ausgewählt. Auf die Region wird ein Muster gelegt. Je nach Einstellung werden N Punktpaar auf dem Muster ausgewählt und deren Intensitätswerte verglichen. Ist Punkt A heller als Punkt B ist das Resultat eine 1, sonst 0. Dies wird für die N Punktpaare wiederholt. Anschließend wird ein Stichprobenmenge aus N entnommen und als Binärwort abgelegt. Für das spätere Matching mit einem anderen Deskriptor muss lediglich die Summe über die XOR Verknüpften Binärwörter gebildet werden um die Ähnlichkeit zu bestimmen. $\text{Sum}(\text{xor}(\text{string1}, \text{string2}))$ ist die sog. Abstandsfunktion. Aus der Abstandsfunktion lässt sich schließen, dass die Reihenfolge der gespeicherten Werte keine Rolle spielt.

3.3.3 ORB - Oriented FAST and Rotated BRIEF

ORB ist eine Erweiterung des FAST Detektors und BRIEF Deskriptors. BRIEF war 2011 der erste veröffentlichte binäre Deskriptor. Er hat noch kein festes Muster. Stattdessen wird ein zufälliges Muster erstellt. Der Deskriptor erzielt damit eine hohe Wiedererkennungsrate, solange es keine Rotation im Bild gibt, denn es wurde auch noch keine Ausrichtungskompensation implementiert. ORB erweitert BRIEF mit einer entsprechenden Methode und ist damit rotationsinvariant. Um die Orientierung eines Interest-Operators zu messen verwendet ORB einen *intensity centroid* [18], bestimmt also den Schwerpunkt der Intensitätswerte. Die Momente der Region errechnen sich zu

$$m_{pq} = \sum_{x,y} x^p y^q I(x, y) \quad (3.17)$$

mit $I(x,y)$ als Intensitätsfunktion und p,q als Grad des Moments. Der Schwerpunkt wird berechnet durch

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right) \quad (3.18)$$

und schließlich die Richtung durch

$$\theta = \text{atan2}(m_{01}m_{10}) \quad (3.19)$$

4. Kamerasysteme

4.1 Kamerasensoren

Um eine Schätzung der Pose eines Fahrzeugs in seiner Umgebung vornehmen zu können, muss ein Modell der Umgebung erstellt werden. Kamerasensoren sind externe Sensoren die Eigenschaften und Informationen der Umwelt erfassen.

Kamerasensoren sind mittlerweile preisgünstig, während die Qualität und Auflösung der Bilder hoch ist. Dadurch sind sie einer breiten Masse von Forschern und Studenten zugänglich, auch ein Grund ist warum visuelle Bildverarbeitungssysteme in der Robotik so verbreitet sind.

Um die Kamerapose durch Visuelle Odometrie zu schätzen muss in jedem Fall ein Weg gefunden werden Informationen über die Distanz zwischen der Kamera und Objekten aus der Umgebung zu gewinnen. Die Tiefe kann mit einer einzelnen Kamera rein algorithmisch geschätzt werden. Mit zwei Kameras lässt sich durch Stereoskopie ein Sonderfall der Epipolare Geometrie ausnutzen und Punkte können effizient trianguliert werden. Spezielle Tiefenbildkameras mit Infrarot (IR) Projektion oder Time-of-Flight (ToF) Messungen lösen das Problem vollständig in Hardware. Im folgenden soll auf die wichtigsten Sensoren kurz eingegangen werden.

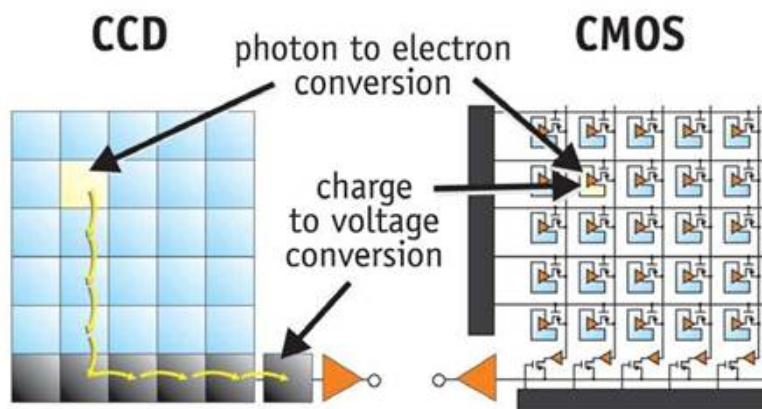


Abbildung 4.1: Aufbau CCD vs CMOS (Merolini, 2022 [19])

4.1.1 RGB Kamera mit CCD- oder CMOS-Sensoren

CCD Sensoren

Die CCD Technologie wurde Mitte der 70er Jahre erfunden. CCD war damals der Durchbruch der Halbleitersensoren. Die Abkürzung CCD steht dabei für "Charge-Coupled Device".

Aufgrund der Funktionsweise des Ladungsträgertransports werden sie oft mit Eimerketten verglichen. Eine Elektronenschicht liegt auf einer Fotozellen Silizium-Trägerschicht. Auf die Fotozelle einfallendes Licht löst negativ geladene Elektronen aus den Silizium Atomen. Je intensiver das Licht, desto mehr Elektronen lösen sich. Jede Fotozelle ist mit einem Gate verbunden. An das Gate wird eine Spannung angelegt, wodurch die Ladung angezogen werden. Durch gezieltes heben und senken des Potentials werden die Ladungsträger durch die Speicherbereiche geschoben. Die Ladung wandert solange durch das Array, bis der Rand des Sensors erreicht ist. Dort wird das Ladungspaket von einem externen Verstärker in eine Spannung konvertiert.

CCD ist zeichnet sich durch eine hohe Bildqualität aus. Die Frequenz, mit der der Ladungsträgertransport durchgeführt werden kann, ist jedoch begrenzt und niedriger als bei CMOS. Bei zu hohen Frequenzen würde im CCD zu viel Ladung verloren gehen. Das Verschieben der Ladungsträger in CCD Sensoren kann außerdem zum sog. Blooming Effekt führen, der bei CMOS Sensoren nicht auftritt.

CMOS Sensoren

Complementary Metal-Oxide Semiconductor, kurz CMOS, Sensoren sind ebenfalls bereits seit den 1970er Jahren auf dem Markt. Es dauerte jedoch noch einige Jahrzehnte bis die CMOS Technologie die heutige Bildqualität erreichte. Da CMOS-Sensoren neben der hohen Bildraten mittlerweile auch eine gute Bildqualität vorweisen, verdrängen sie CCD-Sensoren aus den meisten Anwendungen.

Während CCD die Ladungspakete extern in Spannung konvertiert, wird die Ladung bei CMOS bereits im Pixel umgewandelt. CMOS Sensoren die selbst Ladung konvertieren werden deshalb auch als Aktiver Pixel Sensor (APS) bezeichnet.

Ein einfaches CMOS-Sensor Pixel besteht aus drei MOSFETs und einer Photodiode wie in 4.2 gezeigt. M_{sf} agiert als Verstärker, M_{rst} als Reset der Photodiode und mit M_{sel} kann die Reihe ausgewählt werden.

Wird der Reset geöffnet, erzeugt die Photodiode je nach einfallender Lichtintensität Spannung am Verstärker. Die Spannung wird über den SEL Transistor ausgelesen.

Wird ein Bild von einem Objekt aufgenommen, das sich in Bewegung befindet, kann es zum sog. Rolling-Shutter Effekt kommen. Durch eine minimale zeitliche Verzögerung bei der Belichtung der Reihen tritt eine Verzerrung im Bild auf. Der Effekt wird Behoben indem zwei weitere Transistoren und ein Kondensator in die

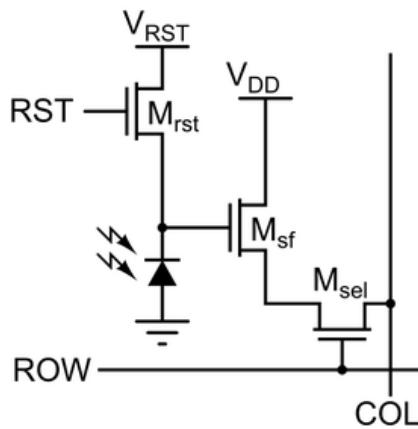


Abbildung 4.2: Schaltung eines einfachen 3 Transistor aktiven Pixels (Wikimedia Creative Commons [20])

Pixelzelle hinzugefügt werden. Dadurch können alle Photodioden zur selben Zeit zurückgesetzt werden und die Belichtungszeit wird synchronisiert.

4.1.2 Tiefenbildkameras auf Basis von Projektionen

Das Bekannteste Beispiel dieses Kameratyps ist wohl die ursprünglich für die XBox Spielekonsole eingeführt Kinect 1 Kamera. Ein IR-Transmitter sendet ein bekanntes Muster aus, das von einem Receiver wieder empfangen wird. Objekte auf die das Muster trifft reflektieren das Muster. Je nach Beschaffenheit und Entfernung des Objekts ist das Empfangende Muster verzerrt und anders skaliert.

Das Erfassen von Entfernung durch Musterprojektionen ist sehr empfindlich gegenüber Fremdlicht und hat nur eine begrenzte Genauigkeit. Es gibt daher auch den Ansatz Projektion mit Stereoskopie zu kombinieren wie beispielsweise bei Intels D400 Serie. Neben einer höheren Reichweite ist die Motivation für die IR Projektion hier vor allem ein Nachteil der Stereokamera auszugleichen. Finden sich keine markanten Strukturen im Bild, beispielsweise bei einer weißen Wand, kann Stereoskopie versagen. Für solche Fälle führt eine Fusion aus Stereoskopie und Projektion, zumindest Indoor (Reichweite 4m), zu wesentlich Robusteren Ergebnissen. [21]

4.1.3 Tiefenbildkameras auf Basis von Stereoskopie

Bei Stereokameras sind zwei Kameras exakt parallel zueinander ausgerichtet und um eine Baseline b verschoben. Aus der Epipolargeometrie folgt, dass in diesem Fall die Epipole im unendlichen liegen. D.h. die Epipolarlinien verlaufen auf gleicher Höhe durch die Bildebenen der beiden Kameras. Das bedeutet, der Suchraum eines Punktes aus dem linken Kamerabild ist im Rechten Kamerabild enorm beschränkt. Wurde die Abbildung des Punktes in beiden Bildebenen gefunden kann direkt trianguliert werden, da die Baseline bereits bekannt ist. Auf diese Weise wird ein Tiefenprofil erstellt. Stereokameras nutzen Kamasensoren und haben eine entsprechend große Reichweite.

4.1.4 Time-of-Flight Tiefenbild

Time-of-Flight Kameras verwenden Photonic Mixing Device (PMD) Sensoren um über ein Laufzeitverfahren die Distanz zwischen Objekten und Kamera zu messen. Zur Laufzeitmessung wird ein gepulster Infrarot Laserstrahl ausgesendet und von einem Receiver empfangen. Dabei wird durch die Phasenverschiebung des Pulses gemessen wie viel Zeit zwischen Senden und Empfangen vergangen ist. Da die Laufzeit von Licht bekannt ist, kann so auf die zurückgelegte Distanz des Laserpulses geschlossen werden.

$$t_D = 2 \cdot \frac{D}{c} \quad (4.1)$$

Mit t_D Zeit, D Distanz und c Lichtgeschwindigkeit $\approx 300\,000 \frac{\text{km}}{\text{s}}$.

4.2 Kamerakonfigurationen

Da die Visuelle Odometrie auf Kameradaten basiert stellt sich auch die Frage, wie viele Kameras benötigt werden. Grundsätzlich können eine, zwei oder mehrere Kameras beliebig am Fahrzeug montiert werden. Aber nicht jeder Aufbau ist effizient oder sinnvoll.

Durchgesetzt hat sich monokulare Visuelle Odometrie mit einer Kamera und stereo VO mit zwei Kameras. In dem Versuch die Robustheit von Visueller Odometrie zu erhöhen, vor allem in schlecht belichteten Umgebungen wurden auch Mehrkamerasysteme mit mehr als zwei Kameras untersucht. Liu et al. untersuchten 2019 ein Multi-Stereokamera Setup [22] und erzielten bessere Ergebnisse für Nachtfahrten mit einem Offline-Dataset. Mhiri et al. veröffentlichten bereits 2014 eine Methode in der sie Visuelle Odometrie mit klassischer SfM und Machine Learning kombinierten um die Odometrie aus einem Mehrkamerasystem mit zu schätzen, bei dem die Kameras nicht synchronisiert wurden [23].

Mehrkamerasysteme sind eine Seltenheit um sehr spezielle Problemstellungen zu lösen. Weitaus häufiger wird Visuelle Odometrie durch Sensorfusion mit anderen Sensoren kombiniert um die Robustheit zu erhöhen. Es ist außerdem zu beobachten, dass der Trend in die entgegengesetzte Richtung geht. Viele, in den letzten Jahren veröffentlichte Paper, untersuchen monokulare Visuelle Odometrie. Monokulare Visuelle Odometrie ist bereits so lange bekannt wie Stereo Visuelle Odometrie, mittlerweile ist die Technik jedoch weit genug fortgeschritten um monokulare Systeme mit Neuronale Netze (lernende Visuelle Odometrie Verfahren) zu implementieren, welche die Nachteile von monokularen Systemen behoben sollen.

4.2.1 Monokulare Kamerasyteme

Bei monokularen Kamerasytemen wird die Odometrie anhand aufeinanderfolgender Bilder einer einzelnen Kamera geschätzt. Der Vorteil dabei ist, das weniger Hardware benötigt wird. Das größte Problem ist die Tiefenschätzung. Ohne

eine bekannte Basislinie, wie bei Stereokameras, muss die Skalierung zwischen den Bildern ebenfalls geschätzt werden. Dadurch erhöht sich die Drift des Systems. Monokulare und Stereo Visuelle Odometrie haben sich über die Zeit als zwei unterschiedliche Forschungsgebiete entwickelt. Dennoch kann monokulare Visuelle Odometrie auch für stereo Visuelle Odometrie interessant werden, wenn der Abstand zu Szene wesentlich größer als die Basislinie ist. In dem Fall wird aus der stereo eine mono Visuelle Odometrie. Für autonome Fahrzeuge trifft dieser Fall i.d.R. nicht ein.

4.2.2 Stereo Kamerasysteme

Stereokameras setzen sich aus zwei Kameras zusammen, die exakt parallel ausgerichtet sind und eine feste Basislinie haben. Der Grund dafür sind die besonderen Eigenschaften der Epipolargeometrie solch eines Aufbaus. Dadurch können Pixel schnell Trianguliert werden und so eine Tiefenkarte aus der Geometrie geschätzt werden. Stereokameras sind günstig, liefern qualitativ hochwertige Bilder und sind seit Markteinführung der Kinect Kamera auch bei Robotik Enthusiasten verbreitet. Auch wenn die Triangulation ein schnelles und Zuverlässiges Verfah-



Abbildung 4.3: Szenen mit wenig Textur sind auch bei ausreichender Beleuchtung eine Herausforderung für Stereokamera Triangulation (Shutterstock ID 1248440977, 2022)

ren ist, muss die abgebildete Szene bestimmte Voraussetzungen Erfüllen. Ohne ausreichende Beleuchtung oder Textur arme Umgebungen führen dazu, dass Pixel nicht abgeglichen werden könne. Ohne ein erfolgreiches Matching kann die Triangulation nicht stattfinden.

4.2.3 RGB mit Tiefenbildkamera

Für ein Kamerasystem mit RGB-D Kamera wird der gleiche Odometrie Algorithmus implementiert wie für Stereokameras. Software seitig gibt es lediglich den Unterschied, dass das Stereo-Matching und die Triangulation entfällt. Falls diese Schritte nicht ohnehin bereits im Kamerasystem abgearbeitet wurden.

Theoretisch kann eine RGB-D Kamera dort Datenpunkte liefern, wo eine reguläre Stereokamera versagt. Schlechte Lichtverhältnisse und wenig Struktur im Bild haben keine negativen Auswirkungen. Genauso wie sich wiederholende Muster und Asphalt kein Problem darstellen. Dafür wird RGB-D aufgrund des Infrarotlasers durch Sonneneinstrahlung und Reflexionen von Spiegelnden Oberflächen gestört. Dazu kommt, dass die meisten RGB-D Kameras nur über eine geringe Reichweite von wenigen Metern verfügen. In der Praxis werden sie daher überwiegend in geschlossenen Räumen verwendet. In den letzten Jahren kamen jedoch bereits ein paar Kameras auf den Markt, die speziell für den SLAM Einsatz entwickelt wurden und größere Reichweiten versprechen [24].

4.2.4 Rektifizierung

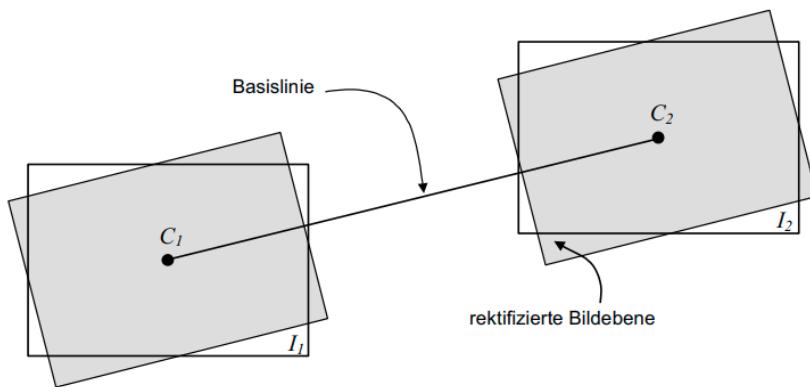


Abbildung 4.4: Bildebenen einer Stereokamera werden Rektifiziert. Dadurch verlaufen die Epipolarlinien parallel zu den Bildzeilen. (Scheer, 2005 [4])

Das simulierte Stereokamerasystem das in CARLA am Fahrzeug angebracht wird ist ideal. Alle Kameraparameter sind aus der Konfiguration bekannt. Zwischen der linken und rechten Kamera gibt es keine ungewollte Verschiebung. In realen Systemen sind Stereokameraaufbauten i.d.R. nur näherungsweise parallel.

Deshalb werden die Bilder durch eine softwareseitige Drehung in ein tatsächlich achsenparalleles System transformiert. Dieses virtuelle Drehen wird als Rektifizierung bezeichnet und die Kamerabilder als rektifiziert. Die Transformation ist linear und bei erfolgreich transformierten Stereobildern liegen die Epipole im unendlichen.

Für eine Rektifikation müssen die internen Kameraparameter bekannt sein. Diese sog. intrinsischen Parameter liefert entweder der Kamerahersteller oder werden durch Kamerakalibrierung ermittelt. Wichtig ist zudem, dass sich die Kamerazentren (C_1, C_2) durch die Rektifizierung nicht verschieben.

5. Systementwurf

5.1 Erzeugen von Sensoroutput

5.1.1 Kameradaten

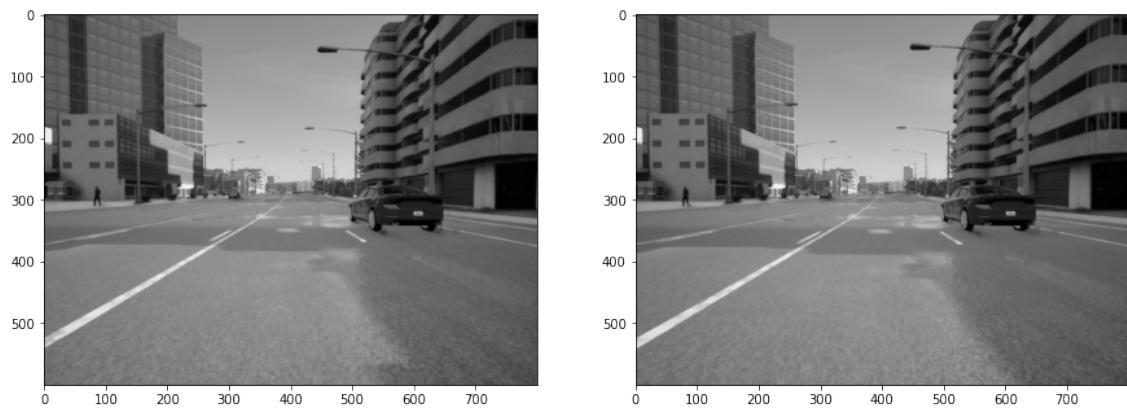


Abbildung 5.1: Grayscale konvertierte RGB Bilder aus Stereo Setup auf Fahrzeug in CARLA Simulator montiert.

Alle Sensoroutputs werden im Simulator generiert.

Zu Beginn kommen Kameradaten in das System. Entweder in Form von zwei RGB Bildern, die um eine Baseline $b = 50\text{cm}$ verschoben aufgenommen wurden. Oder als RGB und Depth Image Paar.

5.1.2 Node für Kamerasynchronisation

Für die Visuelle Odometrie wird ausschließlich der RGB Kamasensor und der Pseudosensor Depth Map verwendet. Für die Auswertung werden zusätzlich die Ground Truth Daten des Pseudosensors aufgenommen. Die Bridge konvertiert die Sensordaten aus der Simulation in den sensor_msg Datentyp und veröffentlicht die Nachrichten auf ein entsprechendes Topic im Netzwerk.

Eine CameraInfo.msg Nachricht enthält:

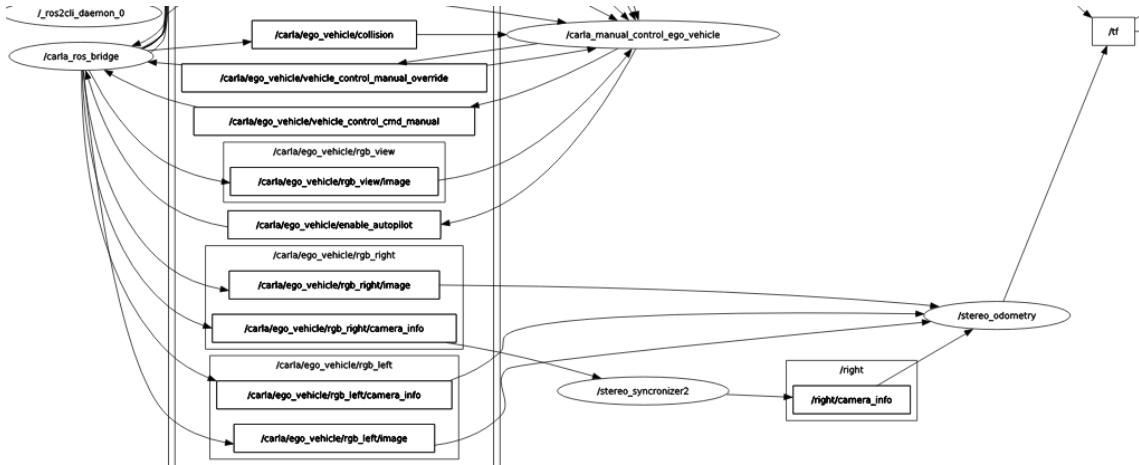


Abbildung 5.2: Graph zeigt im unteren Teil die Verknüpfungen der Topics aus der Simulator-Bridge bis zur gesuchten Transformation.

Die 3x3 Kamera-Matrix K

K enthält die intrinsischen Parametern des Rohbildes (enthält ggf. Verzerrungen). K projiziert 3D Punkte der Kamera-Koordinaten-Ebene auf 2D Pixel-Koordinaten mit $f_{x/y}$ Brennweite und Bildmittelpunkt $c_{x/y}$

$$K = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

Die 3x3 Rektifizierungsmatrix R

R ist eine 3x3 Rotationsmatrix, die das Kamera Koordinatensystem an der idealen Bildebene ausrichtet, damit die Epipolarlinien beider Bilder parallel sind.

Die 3x4 Projektionsmatrix P

Die Projektionsmatrix enthält die intrinsischen Parameter nach der Rektifizierung. Mögliche Verzerrungen wurden durch einen Kamera-Kalibrierungsschritt korrigiert. Unter Verwendung der Simulierten CARLA Sensoren existiert keine Verzerrung und die Parameter gleichen der Kameramatrix K .

$$P = \begin{pmatrix} f'_x & 0 & c'_x & T_x \\ 0 & f'_y & c'_y & T_y \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Für Stereokameras entspricht der vierte Zeilenvektor $(T_x \ T_y \ 0)'$ der Position des optischen Zentrums der zweiten Kamera im Frame der ersten. Die erste Kamera ist dem zu folge immer $T_x = T_y = 0$. Ohne Korrekturschritt wäre das auch für die zweite Kamera der Fall. Stattdessen berechnen sich die Parameter für die zweite Kamera zu:

$$\begin{aligned} T_y &= 0 \\ T_x &= -f_x * B \end{aligned}$$

Für ein horizontal ausgerichtetes Setup, mit B für die Baseline zwischen den Kameras.

CARLA bietet keinen Stereokamera Sensortyp an. Es müssen zwei einzelne Kamerasensoren mit der korrekten Ausrichtung am Fahrzeug initialisiert werden. Über die Bridge werden die Kameras allerdings auch als voneinander unabhängige Kameras veröffentlicht.

Anstatt einen neuen Sensortyp in CARLA zu implementieren wurde ein ROS2 Node zum ändern der Nachricht geschrieben. Der Node abonniert das Topic der rechten Kamera. Die Information zu den Kameraparametern werden entsprechend der Stereokamera angepasst und als neue CameraInfo.msg veröffentlicht. Andere Nodes können jetzt einfach das geänderte Kamera Topic abonnieren anstelle des Topics von der Bridge.

5.2 Stereo Visuelle Odometrie Ablauf

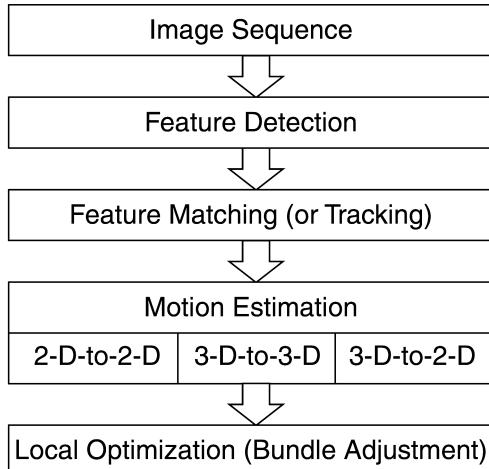


Abbildung 5.3: Klassische Pipeline die bei der visuellen Odometrie durchlaufen wird. (In Anlehnung an Scaramuzza, 2011 [2])

Die Visuelle Odometrie ist ein Verfahren, bei dem die Trajektorie inkrementell geschätzt wird. Der Algorithmus bzw die sog. Visual Odometrie Pipeline wird für jede Schätzung durchlaufen.

5.2.1 Merkmalsdetektion

Es werden zwei aufeinanderfolgende Bilder Img_{t-1} und Img_{t+0} auf Merkmale untersucht. Die Bilder stammen aus der linken Kamera bei Stereo RGB Setup bzw aus der RGB Kamera beim RGB-D Setup.

Die Suche nach Merkmalen erfolgt über einen Detektor. Untersucht werden ORB und GFTT. GFTT (Good Features To Track) ist die Implementierung des Shi-

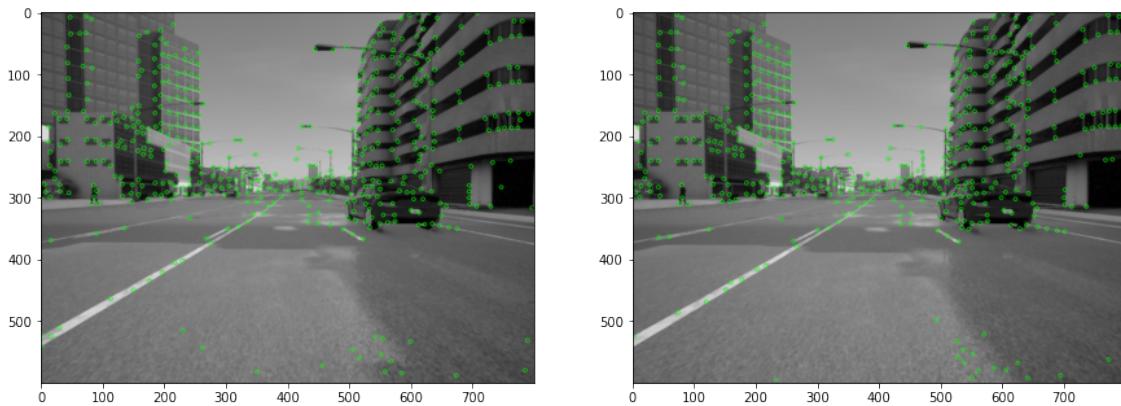


Abbildung 5.4: Merkmalsdetektion in zwei aufeinanderfolgenden Bildern einer CARLA Szene

Tomasi Interest-Operators und detektiert Kanten im Bild. GFTT ist ein reiner Detektor.

Alternativ wird die Extraktion mittels ORB Detektor betrachtet. Für diesen Fall werden die Merkmale durch einen oriented FAST-9 Detektor gesucht. In anderen Arbeiten wurde gezeigt, dass der ORB Detektor 64 mal schneller Merkmale extrahiert als GFTT, bei großen Veränderungen der Skalierung oder des Kamerawinkels allerdings weniger Merkmale liefert.[25]

Die Detektoren liefern Keypoints für die gefundenen Merkmale. Diese werden an Detektor-Algorithmen übergeben. Ein Detektor ist vergleichbar mit einem Fingerabdruck die eindeutige Beschreibung eines Merkmals. Ein durch einen Detektor beschriebener Keypoint lässt sich auch nach Veränderung im Bild wiederfinden. Diese Robustheit gegenüber Veränderungen wird als Invarianz bezeichnet und ist eine wesentliche Voraussetzung für das Matching von Merkmalen zwischen zwei Zeitschritten.

Als Deskriptor wird in jedem Fall ORB verwendet.

5.2.2 Matching von Merkmalen

Beim Matching von Merkmalen wird bestimmt welche Deskriptoren in beiden Bildern auftreten. Durch die Bewegung der Kamera können Merkmale aus der Szene das Bildfeld verlassen bzw hinzugekommen sein. Außerdem können Merkmale zB verdeckt worden sein.

Matches sind eine Menge von Deskriptoren, die für beide Bilder Img_{t-1} und Img_{t+0} gültig ist. Durch diese Menge wurde eine Beziehung zwischen den Bildern hergestellt.

Der simpelste Algorithmus für das Matching ist die Implementierung von Brute Force. Als erfolgreiches Match gilt ein Vergleich innerhalb einer Suchregion mit der kleinsten Distanz. Brute Force Matcher liefern gute Ergebnisse, verglichen allerdings sehr viele Merkmale. Daher ist Brute Force die langsamste Methode für den Abgleich.

Eine Methode die für große Mengen an Deskriptoren (>1000) [25] schneller sein kann ist der FLANN (Fast Library for Approximate Nearest Neighbors) basierte Matcher. FLANN baut einen KD-Baum und gleicht Deskriptoren nur in einer Nachbarschaft ab. Dadurch werden gute Matches schnell gefunden, das Resultat sind aber nicht zwangsläufig das best-mögliche. Eine Erhöhung der Qualität der Ergebnisse kann über Parameter erfolgen, führt aber zu höheren Kosten in der Laufzeit und erhöht die Gesamtausführungszeit.

Um ein möglichst robustes Ergebnis zu erzielen wird der Brute Force Matcher

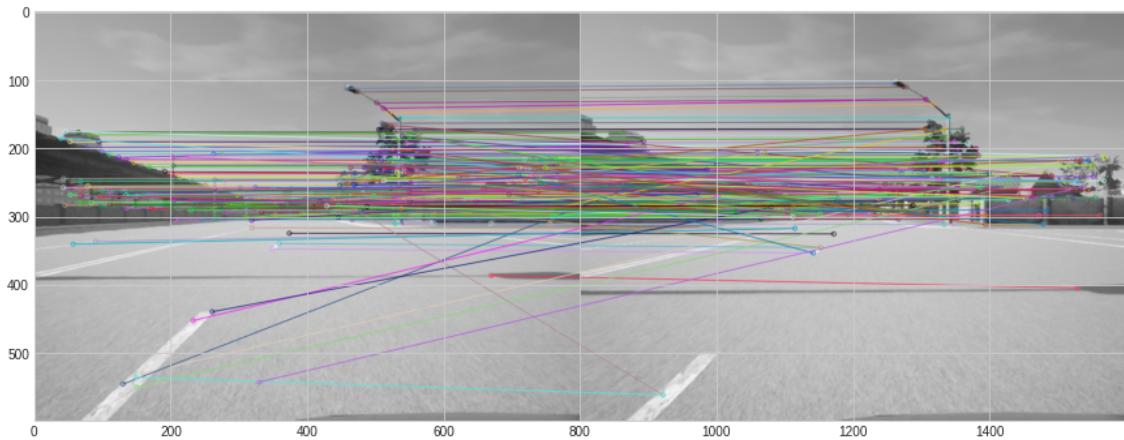


Abbildung 5.5: Brute Force Merkmalsabgleich ohne Cross Checking in zwei aufeinanderfolgenden Bildern einer CARLA Szene.

mit Cross-Check verwendet. Beim Cross-Check werden die Resultate der beiden Bilder darauf Verglichen, ob das beste Match in Set A auch das beste Match in Set B war und umgekehrt. Cross-Checking liefert konsistente Resultate und ist eine Alternative zum sonst verwendeten Lowes Ratio Test für das Filtern von Ausreißern. [26] In der OpenCV Implementierung des Brute Force Matcher kann

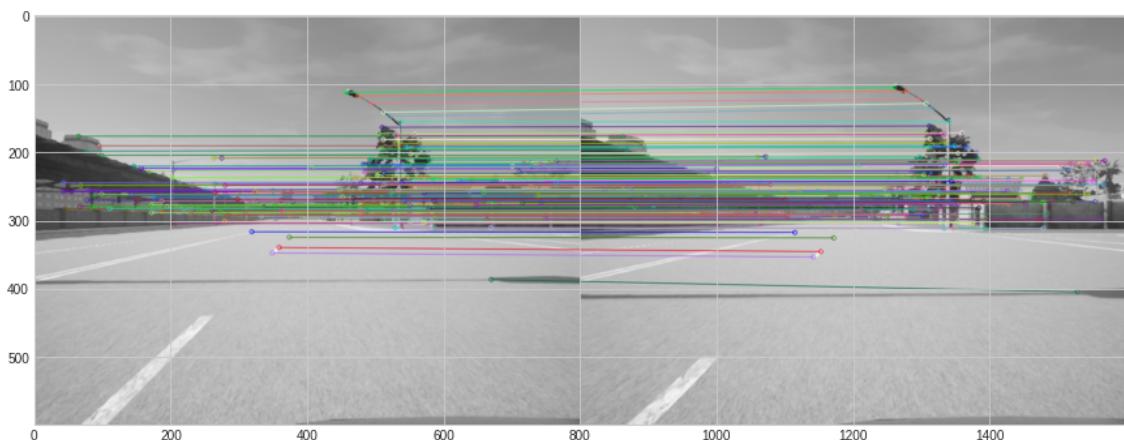


Abbildung 5.6: Brute Force Merkmalsabgleich mit Cross Checking in zwei aufeinanderfolgenden Bildern einer CARLA Szene.

außerdem ebenfalls ein Nachbarschafts-beschränkung mitgegeben werden. Für

Binäre Deskriptoren wie ORB wird hier am besten die Hamming Distanz anstatt des Standardmäßigen Euklidischen Abstands gewählt.

5.2.3 Tiefeninformation

Die zwei simulierten Setups beziehen ihre Tiefeninformation aus unterschiedlichen Weise. In jedem Fall kann auf Pixelebene die Tiefe als ein zusätzlicher Kanal betrachtet werden. Beim Stereo RGB Kamera Setup liegen die Tiefeninformatio-

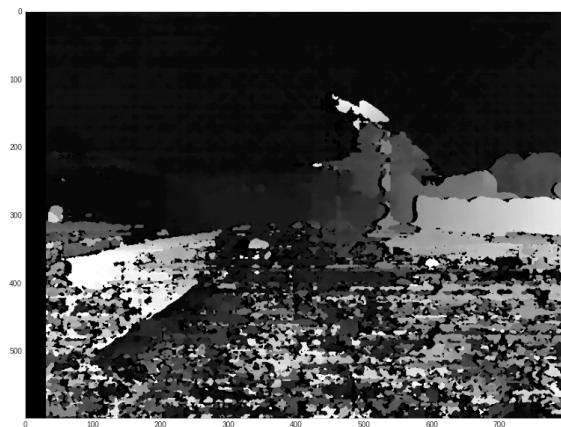


Abbildung 5.7: Disparitätenkarte als Ergebnis eines Stereo Matchings einer CARLA Szene.

nen nicht direkt vor sondern werden über Triangulation von Stereomatches unter Zuhilfenahme der Epipolargeometrie berechnet. Daraus entsteht die sog. Disparitätenkarte. Bei der Programmierung wird hier ein StereoMatcher verwendet. StereoMatcher sind Matching Algorithmen die die Epipolaren Beschränkungen gezielt zur Beschleunigung der Suche einsetzen. Im vorgestellten Systementwurf wird ein in OpenCV implementierter Semi-Global Matching Algorithmus basierend auf einem 2008 von Hirschmüller veröffentlichten Paper verwendet. [27] RGB-D liefert das Tiefenbild direkt, weshalb keine zweite Kamera und keine Berechnung der Verschiebung zwischen den Bildern nötig ist. Real wird die Tiefe über eine Time-of-Light Messung bestimmt. In der Simulation werden die Tiefenbilder direkt vom Simulator ausgegeben mit einer maximalen Tiefe von einem Kilometer, was reale ToF Kameras bei weitem übertrifft.

5.2.4 Bewegungsschätzung

Die Bewegungsschätzung erfolgt von Frame zu Frame (F2F). Für jedes Match werden die Pixelkoordinaten aus den Keypoints ausgelesen. Dadurch liegen zwei 2D Punktmengen für die beiden Bilder vor. Für die Matches aus dem vorherigen Bild Img_{t-1} werden außerdem die Tiefeninformation der Pixel zu den Punkten hinzugefügt. Die Information lässt sich bei Stereo Kamera aus der Disparitätenkarte lesen bzw aus dem Tiefenbild bei einer RGB-D Kamera.

Dadurch liegen schließlich eine 3D Punktmenge der Matches für Img_{t-1} und eine 2D Punktmenge für Img_{t+0} vor.

Die 3D Punkte aus dem vorherigen Zeitschritt dienen als Objekte im Weltkoordinatensystem zu denen die aktuelle Punkte in Bezug gesetzt werden. Es wird die Transformation gefunden für die der Re-Projektionsfehler am geringsten ist. Die Reprojektion die hier betrachtet wird läuft von den 3D Punkten durch die 2D Bildpunkte zum Kamerabrennpunkt. (Anders als beim Bundle Adjustment wo die Reprojektion auf die 2D Bildpunkte betrachtet wird).

Das Problem des Schätzens der Pose aus Korrespondenzen aus 3D Referenzpunkten und ihrem 2D Abbild in einer kalibrierten Kamera wird als Perspective-from-N-Points Problem (PnP) bezeichnet und ist fundamentaler Bestandteil vieler Computer Vision Anwendungen. Erste Lösungen der Problemstellung existieren bereits seit 1841. Heute wird i.d.R. eine Kombination aus PnP mit RANSAC verwendet, da PnP empfindlich gegenüber Ausreißern ist. RANSAC filtert Ausreißer. Mathematisch wird bei PnP Algorithmen versucht eine eindeutige Lösung für ein überbestimmtes Gleichungssystem aus Polynomgleichungen zu bestimmen. Ohne weitere Information zu den Punkten werden mindestens 4 Punkte für eine eindeutige Lösung benötigt. Die Berechnung für P4P ist jedoch wesentlich komplexer als für $p \geq 5$ weshalb wenn möglich, wie bei der Visuellen Odometrie oder allgemein SLAM Verfahren mit mindestens 5 Punkten berechnet wird.

Unter Zuhilfenahme des Lochkamera-Modells und der Kameramatrix K werden die 3D Punkte aus der Bildebene (Pixelkoordinaten) in das Kamerakoordinatensystem transformiert.

$$sP_c = K[R|T]p_w \quad (5.1)$$

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (5.2)$$

Umformung zu

$$P_c = K^{-1} \cdot (s \cdot [u, v, 1]) \quad (5.3)$$

Bei PnP wird also die Bewegung aus einer 3D zu 2D Transformation geschätzt.

$$\begin{aligned} T_k &= \begin{pmatrix} R_{k,k-1} & t_{k,-1} \\ 0 & 1 \end{pmatrix} \\ &= \arg \min_{T_k} \sum_{i=1}^n \|p_k^i - \hat{p}_{k-1}^i\|^2, \end{aligned} \quad (5.4)$$

5.2.5 Pose Update und Verlorene Odometrie

Ausgabe der Schätzung ist ein Translationsvektor t und eine Rotationsmatrix R für die aktuelle Position. Durch Multiplikation der vorherigen Pose mit $R|t$ wird die aktuelle Pose berechnet. Addition der Posen gibt die Trajektorie.

An dieser Stelle wird ein Update der Output Odometrie vorgenommen und die ROS Transformation von $tf \text{ odom} \rightarrow base_link$. Die ausgegebene Kovarianz

wird berechnet durch die Median Absolute Deviation (MAD) zwischen den 3D Merkmalskorrespondenzen.

$$MAD = \text{median}(\|X_i - \tilde{X}_i\|) \quad (5.5)$$

Unter Umständen konnte keine Schätzung durchgeführt werden, Beispielsweise weil nicht genügend Matches gefunden werden konnten. In diesem Fall wird die aktuelle Kameraposition auf die letzte gültige Kameraposition gesetzt.

6. Implementierung

Um das System zu implementieren und zu testen wurde bereits existierende Software und Bibliotheken verwendet die in ihrer Funktion und Anwendung im Projekt kurz erläutert werden sollen.

6.1 Simulationsumgebung

6.1.1 Simulator CARLA

Der gesamte Systeminput wird durch den CARLA Simulator[28] erzeugt. Nach eigener Aussage ist die Motivation hinter dem Projekt die Demokratisierung in der Forschung und Entwicklung autonomen Fahrens. CARLA (Car learning to act) ist ein Open-Source Simulator basierend auf der Unreal Engine 4. Die Engine ist kostenlos für nicht-kommerzielle Nutzung und bietet eine Rendering, physikalische Modelle (PhysXVehicles) und NPC Logik für Fußgänger und andere Verkehrsteilnehmer. CARLA startet als Server mit einer Python Client API. Über die API können Parameter der Simulation wie z.B. die Map oder das Wetter gesetzt werden. Ein gespaßtes Fahrzeug ist ein Client der über Sockets mit dem Server kommuniziert. Die echte Position des Fahrzeugs wird über einen Ground Truth Pseudo-Sensor ausgegeben.

6.1.2 ROS2 Framework für Prozesskommunikation

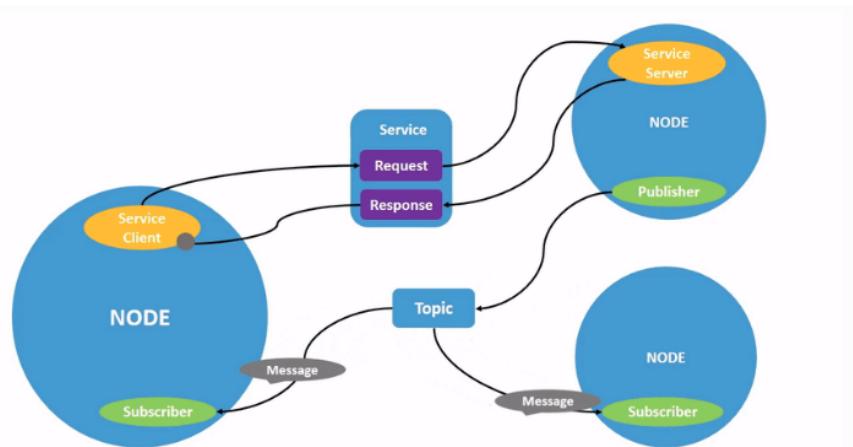


Abbildung 6.1: Ein ROS2 System besteht aus vielen Nodes die Nachrichten über eine Publish-Subscriber Architektur austauschen. (ROS2 Dokumentation, 2022)

Das Robot Operating System ROS2 wird seit 2007 entwickelt und besteht aus einer Sammlung von Bibliotheken und Tools für Robotik Anwendungen. Anders als der Name vermuten lässt, ist ROS2 kein in sich geschlossenes Betriebssystem sondern ein Framework das unter Linux betrieben werden kann. ROS2 und darin zur Verfügung stehenden Software sind Open Source.

In einem ROS2 System läuft jeder Prozess in einem Node. Ein Programm kann aus mehreren Nodes bestehen aber ein Node sollte immer nur einen Zweck erfüllen. Dadurch werden die Systeme Modular und ein Node kann in unterschiedlichen Systemen eingesetzt werden. Bei der Initialisierung eines Nodes können Einstellungen durch sog. Parameter übergeben werden. Jeder Node verwaltet seine eigenen Parameter. Beispielsweise werden für die Stereo und RGB-D Odometrie zwei Nodes vom selben Modul mit unterschiedlicher Parametrierung instanziert.

Alle Daten im ROS2 Netzwerk werden in ROS2 Message Datentypen gepackt. Die Nachrichten werden über ein Publish-Subscriber-Architektur im Netzwerk verteilt. Jeder Node kann beliebig viele Topics veröffentlichen oder abonnieren. Das Topic agiert als Datenbus des Systems.

Der große Vorteil dieser Architektur ist die Einführung eines Standards für die Schnittstellen der Programme. Ohne Veränderung an dem Node für die Visuelle Odometrie vornehmen zu müssen, kann die Datenquelle geändert werden und es ist auch keine Kenntnis über die Empfänger der Daten nötig. Die Eingangsdaten müssen nur vom Typ *sensor_msgs Image* und *CameraInfo* sein und ein Empfänger der Odometrie Daten vom Typ *nav_msgs Odometry* erwarten.

6.1.3 Kameradaten

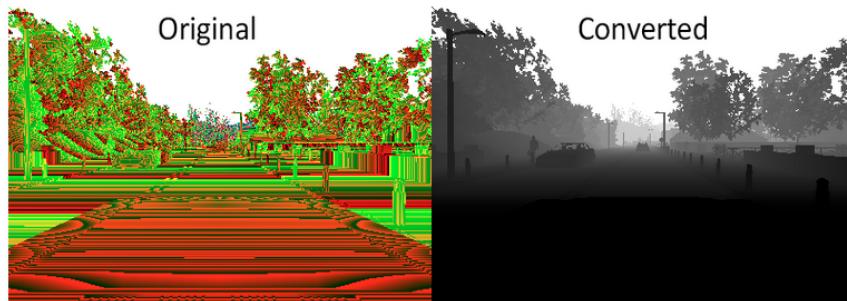


Abbildung 6.2: Depth Map Pseudosensor des Simulators in Original und User freundlichen Konvertierung. (CARLA Dokumentation, 2022)

Der Simulator verfügt viele, für autonomes Fahren relevante, Sensoren wie Kamera, Lidar, IMU, GNSS ect.. Die Sensoren werden über eine JSON-Konfigurationsdatei instanziert und stehen immer in Relation zum Fahrzeug. Dabei sind alle Distanzen in Metern zu lesen.

Für die vorliegende Arbeit wurden zwei RGB Kamerasensoren zu einer Stereo-Kamera konfiguriert und eine RGB Kamera mit Depth Map Sensor als RGB-D

Kamera verwendet. Die Kameras müssen nicht rektifiziert werden und die intrinsischen (Kamera internen) Parameter sind direkt bekannt. Die Auswahl und Positionierung erfolgt über die CARLA API. Die Position wird dabei in Metern und immer in Bezug zum Fahrzeug definiert. Bilddaten werden vom Server als BGRA Bytearray gesendet.

Folgende Effekte aus dem Simulator haben Einfluss auf das Kamerabild:

Grain jitter erzeugt ein künstliches Rauschen. *Blooming* erzeugt eine helle Verzeichnung in der Umgebung einer Überbelichtung. *Auto exposure* ändert den Bild-Gammawert, um Anpassung an dunklere oder hellere Bereiche zu simulieren. *Lens flares* simuliert die Reflexion heller Objekte auf der Linse. *Depth of field* lässt Objekte in der Nähe oder sehr weit von der Kamera unscharf werden.

Unglücklicherweise werden diese Effekte über die Unreal Engine erzeugt und nicht in der Kamera in CARLA. Ein Blooming Effekt würde bei modernen CMOS beispielsweise i.d.R. nicht auftreten sondern ist ein CCD Phänomen. Auch die Implementierung des Auto-Exposure Effekts ist bei CARLA dem Menschlichen Auge nachempfunden um beim User einen besseren Optischen Erlebnis zu erzielen anstatt die Kamera korrekt zu modellieren.

6.1.4 Transformationsbaum

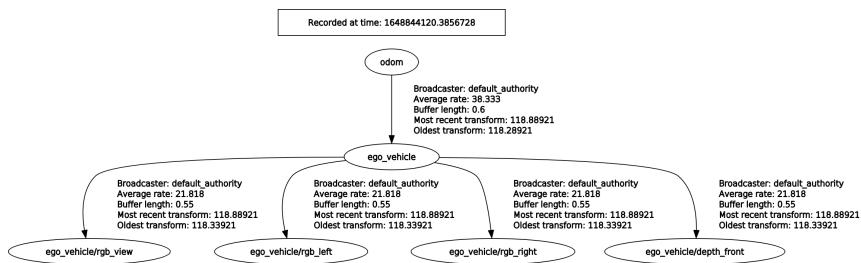


Abbildung 6.3: ROS2 Transformationsbaum für den Versuchsaufbau.

Im Netzwerk kann die räumliche Beziehung zwischen Nodes über einen Transformationsbaum abgebildet werden. Jeder Node hat sein eigenes Koordinatensystem. Über die Transformationen wird bekannt gegeben wie die Koordinatensysteme zu einander ausgerichtet sind. So kann beispielsweise definiert werden wo sich die Kameras in Bezug zum Mittelpunkt des Fahrzeugs befinden. Wird jetzt die Pose des Fahrzeugs aktualisiert, können auch alle Sensor Posen aktualisiert werden.

Abbildung 6.3 zeigt den Transformationsbaum des System. Neben den statischen Transformationen zwischen Sensoren und Fahrzeug (`ego_vehicle`) ist auch die Transformation des Odometrie Frames zum Fahrzeug zu sehen. Wird das System in ein anderes Netzwerk eingebunden kann also über diesen Frame auf das Fahrzeug und alles was mit dem Fahrzeug verbunden ist lokalisiert werden.

6.2 Bibliotheken

6.2.1 Computer Vision Bibliothek OpenCV

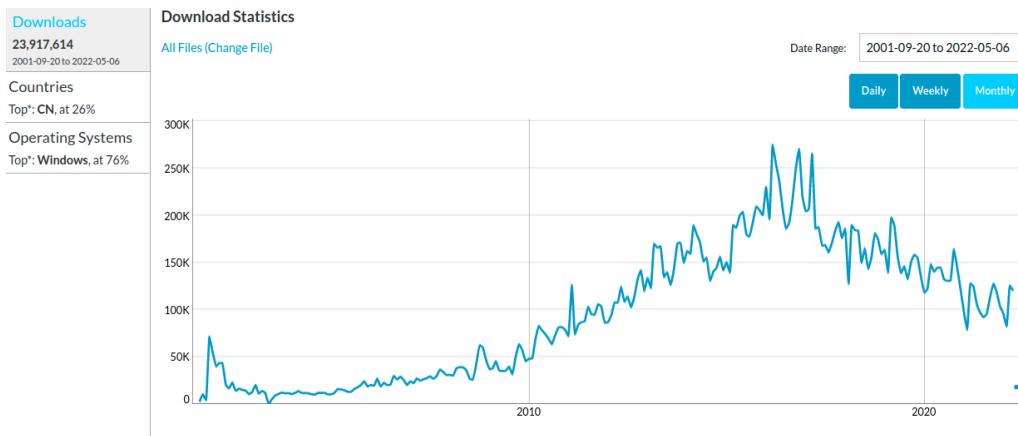


Abbildung 6.4: SourceForge OpenCV Download Statistik zeigt über 23,9 Millionen Downloads seit dem 20.09.2001

Lokalisierung mittels Kameradaten ist ein bekanntes Problem der Bildverarbeitung. Über die Jahrzehnte sind viele Algorithmen und Funktionen entwickelt worden um die verschiedenen Aspekte der Problemstellung zu lösen. Über 2500 davon lassen sich in der Open Source Bibliothek OpenCV (Open Computer Vision) finden. OpenCV wurde ursprünglich von Intel entwickelt und hatte seine Erstveröffentlichung 2000. Der Fokus der Bibliothek liegt auf Echtzeit Bildverarbeitung. Viele Probleme der Bildverarbeitung sind äußerst Rechenintensiv. Durch OpenCV sollen optimierte Implementierungen geschrieben werden und Robotik Ingenieure davon befreit 'das Rad neu erfinden zu müssen'.

OpenCV ist in C++ geschrieben bietet Interfaces in Python, Java, MATLAB und Mobile Device Ports. Durch die hohe Qualität der Bibliothek und die Fülle an Anwendungen der Bildverarbeitung wird OpenCV von sehr vielen Firmen und Forschungseinrichtungen benutzt wie beispielsweise Google, Microsoft, Sony, MIT, Intel, Siemens usw.. Die Zahl der Downloads liegt aktuell bei fast 24 Millionen.

Für den Systementwurf wurde OpenCV als Grundlage für das Prototyping verwendet. Die Bibliothek hat es ermöglicht einen schnellen praktischen Zugang zu der Theorie aus wissenschaftlichen Veröffentlichungen hinter der Bildverarbeitung zu erlangen. Stellenweise konnten Parameter für die VO heuristisch ermittelt werden und es erleichterte den Vergleich verschiedener Algorithmen.

Neben der optimalen Implementierung von Algorithmen wird bei OpenCV auch an eigene Forschung zu neuen Methoden betrieben. Der im System verwendete ORB Algorithmus ist eine Eigenentwicklung der OpenCV Researcher.

Alternativ zu OpenCV kann für das Prototyping beispielsweise MATLAB verwendet werden falls der Entwickler mit dieser Umgebung besser vertraut ist. Die Per-

formance von MATLAB gegenüber OpenCV ist jedoch bis zu 80 mal langsamer [29]. Code der mit OpenCV geschrieben wurde kann zudem direkt auf dem Ziel-system laufen.

6.2.2 V-SLAM Bibliothek RTAB-Map

RTAB-Map ist eine standalone C++ Open Source Bibliothek für Real-Time Appearance Based Mapping. Unter der Verwendung von OpenCV werden RGB-D, Stereo and Lidar Graph-Based SLAM Algorithmen implementiert [30].

RTAB-Map wird auch als ROS und ROS2 Package veröffentlicht. Der ROS Philosophie folgend sind verschiedene Teile der SLAM Algorithmen als Nodes implementiert, aus denen sich ein vollständiges SLAM zusammensetzt. Für Visual SLAM ist die visuelle Odometrie das Front-End zur Bildverarbeitung. Es gibt daher einen Stereo Odometrie Node aus RTAB-Map der zur Realisierung der visuellen Odometrie für die Fahrzeuge des CARS Projekt verwendet werden kann.

Der Node hat eine Parameterliste von 242 Parametern die wie eine API für die unterliegenden Funktionen funktioniert. In den meisten Fällen sind die Parameter Übergabewerte an OpenCV Funktionen. Der Node bietet dadurch die Möglichkeit direkt eine VO Pipeline aufzurufen und sich vollständig auf die Parametrierung der CV Funktionen z.B. WindowSize der Merkmalsdetektoren zu konzentrieren.

7. Ergebnisse der Lokalisierung

7.1 Methodik

Für die Datenerhebung wurde im CARLA Simulator die Umgebung 'Town03' ausgewählt. Die CARLA Dokumentation beschreibt Town03 als abwechslungsreichste und komplexeste Karte des Simulators. Town03 beinhaltet eine große Kreuzung, Erhebungen, Kreisverkehr und einen Tunnel.

Es wurden Daten für 5 unterschiedliche Witterungsverhältnisse erhoben: sunset, clear night, default, hard rain, wet sunrise. Die Witterungsverhältnisse wurden über CARLAs Python API gesetzt.

Für jedes Witterungsverhältnis wurden 5 Testfahrten aufgezeichnet. Die Testfahrten sind zufällige Strecken. Außerdem wurden für alle Fahrten 50 NPCs auf der Karte gespawnt. NPC beinhalten andere Fahrzeuge sowie Fußgänger. Die Geschwindigkeit des Testfahrzeugs variiert zwischen 30 und 100 $\frac{km}{h}$, je nach Streckenabschnitt.

Bei jeder Testfahrt waren die Stereokamera und die RGB-D Kamera am Fahrzeug angebracht, das Bild der linken RGB Kamera wurde in beiden verwendet. Die ROS Nodes für beide Visuellen Odometrie Programme laufen bei allen Fahrten live mit und alle Ausgangsdaten wurden über das ROS Werkzeug rosbag in eine sql3 Datenbank geschrieben.

Ein Verarbeitungstool 'BagParser' wurde geschrieben um Daten aus den rosbag Datenbanken zu extrahieren und in .txt Dateien, OpenCV Matritzen oder Quaternion formatierte .txt Dateien abzulegen.

Die Bewertung von geschätzten Trajektorien bringt einige Herausforderungen mit sich.

Zum einen referenzieren Ground Truth und die Schätzung unterschiedliche Koordinatensysteme. Ground Truth referenziert das Fahrzeugkoordinatensystem während die VO Schätzung das Koordinatensystem der linken Kamera referenziert. Die Positionen müssen also zuerst aneinander angeglichen werden, bevor ein Vergleich durchgeführt werden kann.

Ein anderes Problem ist die Fehlerfortpflanzung. Es kann sein, dass an einer Stelle ein Winkel mit großer Abweichung zu Ground Truth geschätzt wurde. Folgende Schätzungen sind vielleicht mit wesentlich kleineren Fehlern behaftet, zeigen im

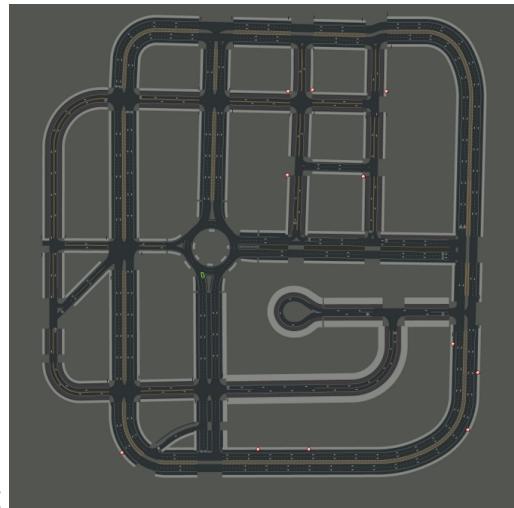


Abbildung 7.1: CARLA Simulator Karte Town03. (CARLA Dokumentation, 2022)

absoluten Vergleich aber noch den vorherigen Fehler.

Es handelt sich bei den Daten außerdem um hochdimensionale Datensätze; die Anzahl der Merkmale ist höher als die Anzahl der Beobachtungen.

Um mit diesen Herausforderungen umzugehen wurden die Daten basierend auf dem Paper *Rethinking Trajectory Evaluation for SLAM: a Probabilistic, Continuous-Time Approach*[31] und der Trajectory Evaluation Toolbox aus *A Tutorial on Quantitative Trajectory Evaluation for Visual(-Inertial) Odometry* [32] bewertet.

Voraussetzung für eine Evaluation ist zuerst die gleiche Ausrichtung der geschätzten Trajektorien mit Ground Truth. Die beiden sind durch eine euklidische Transformation im 3 Dimensionalen Raum miteinander verbunden. Die Transformationen bestehen aus Rotation und Translation und werden formal als Spezielle Euklidische Transformation SE(3) bezeichnet. Anstatt mit Rotationsmatrizen zu rechnen werden für die Auswertung mit rotations Quaternonen gerechnet. Quaternonen sind ein eigener Zahlenbereich und eine Erweiterung des Zahlenbereichs der reellen Zahlen. Als Vorbereitung der Auswertung müssen die Trajektorien daher als Messreihe von 7-Dimensionalen Vektoren abgelegt werden

$$X_i = (x \ y \ z \ q_w \ q_x \ q_y \ q_z)^\top \quad (7.1)$$

Die Ausrichtung wird durch eine Methode der Kleinsten Quadrate Implementation für zwei Punktmengen nach Umeyama [33] gelöst.

$$\arg \min_{R, T, s} \sum_{i=0}^N \|\hat{t}_i - sR_{ti} - T\|^2 \quad (7.2)$$

Mit Ground Truth Posen als \hat{t} und Schätzung durch Visuelle Odometrie als s .

In der Auswertung werden für jede Strecke der absolute Streckenfehler ATE (Absolute Trajectory Error) und der relative Fehler (RE) berechnet. Ground Truth sei X_{gt} und die, bereits ausgerichtete, geschätzte Trajektorie sei \hat{X}' . Die Trajektorie

ist der zeitlicher Verlauf der Zustände der VO Schätzung. Für einen einzelnen Zustand wird der Fehler zwischen \hat{x}'_i und Ground Truth x_i definiert als

$$\Delta x_i = \{\Delta R_i, \Delta p_i\} \quad (7.3)$$

für die gilt

$$R_i = \Delta R_i \hat{R}'_i, \quad p_i = \Delta R_i \hat{p}'_i$$

mit der Rotationsmatrix R und der Position p . Dadurch errechnet sich der absolute Fehler zu

$$\Delta R_i = R_i (\hat{R}'_i)^\top, \quad (7.4)$$

$$\Delta p_i = p_i - \Delta R_i \hat{p}'_i \quad (7.5)$$

Nun kann, um den die Qualität der Gesamtstrecke zu quantifizieren, die Quadratwurzel des mittleren quadratischen Fehlers (root mean square error (RMSE)) gebildet werden

$$ATE_{rot} = \left(\frac{1}{N} \sum_{i=0}^{N-1} \|\angle(\Delta R_i)\|^2 \right)^{\frac{1}{2}}, \quad (7.6)$$

$$ATE_{pos} = \left(\frac{1}{N} \sum_{i=0}^{N-1} \|\Delta p_i\|^2 \right)^{\frac{1}{2}} \quad (7.7)$$

wobei $\angle(\cdot)$ für die Konvertierung der Schreibweise von Matrixform zu Euler-Winkel.

Zusätzlich wird der relative Fehler berechnet. Der von Zhang et al. [32] vorgeschlagene Ansatz ist, die relative Beziehung zwischen den Zuständen zu verschiedenen Zeiten zu messen. Das gemeinsame Kriterium der Zustände ist die zurückgelegte Distanz. Es wird also die Gesamtstrecke und die Schätzung der Gesamten Strecke in Teilstrecken s und e für den Abschnitt \mathfrak{F} zerlegt.

Für jeden Teilabschnitt werden aus den Messwerten Gruppen mit K Zustandspaaren zusammengefasst. Die Menge der Paare d_k bilden jeweils eine Trajektorie für den Streckenabschnitt \mathfrak{F} .

$$\mathfrak{F} = \{d_k\}_{k=0}^{K-1}, \quad d_k = \{\hat{x}_s, \hat{x}_e\} \quad (7.8)$$

Der relative Fehler δd_k berechnet sich auf die gleiche Weise wie der absolute Fehler 7.5. Somit ergibt sich der Fehler δd_k für ein Wertepaar d_k aus

$$\delta \phi_k = \angle \delta R_k = \angle R_e (\hat{R}'_e)^\top, \quad (7.9)$$

$$\delta p_k = \|p_e - \delta R_k \hat{p}'_e\|^2. \quad (7.10)$$

Daraus ergibt sich eine Menge von Werten, jeweils einer pro Teilabschnitt. Die Menge wird formal beschrieben durch

$$RE_{rot} = \{\delta\phi_k\}_{k=0}^{K-1}, \quad (7.11)$$

$$RE_{pos} = \{\delta p_k\}_{k=0}^{K-1}, \quad (7.12)$$

$$(7.13)$$

Die Abbildung dieser Mengen ist in den Boxplots zu sehen. Der mittlere Kasten besteht aus zwei Quartilen und dem Median. Der Median ist der Strich, die Quartile beinhalten sämtliche Schätzfehler. Die Antennen bzw Whiskers bilden das obere und untere Quartil.

7.2 Ergebnisse

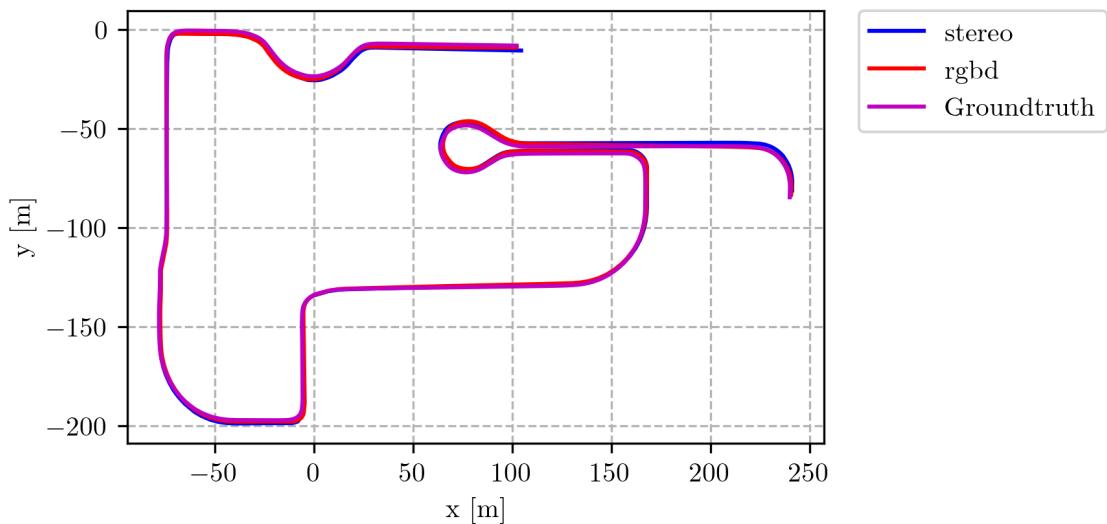


Abbildung 7.2: Trajektorien Ground Truth und Schätzung durch Visuelle Odometrie der Teststrecke 'Default 01'

Es wurden insgesamt Daten für 25 Testfahrten erhoben. Der Datensatz einer Testfahrt besteht immer aus den Ground Truth Odometrie Daten, der durch RGB-D geschätzten Odometrie und der durch Stereo geschätzten Odometrie. Zugunsten der Übersichtlichkeit werden nicht alle Strecken und Daten grafisch aufbereitet. Stattdessen wurden Daten einzelner Strecken als Stichproben abgebildet um einen intuitiven Einblick in die Ergebnisse zu vermitteln.

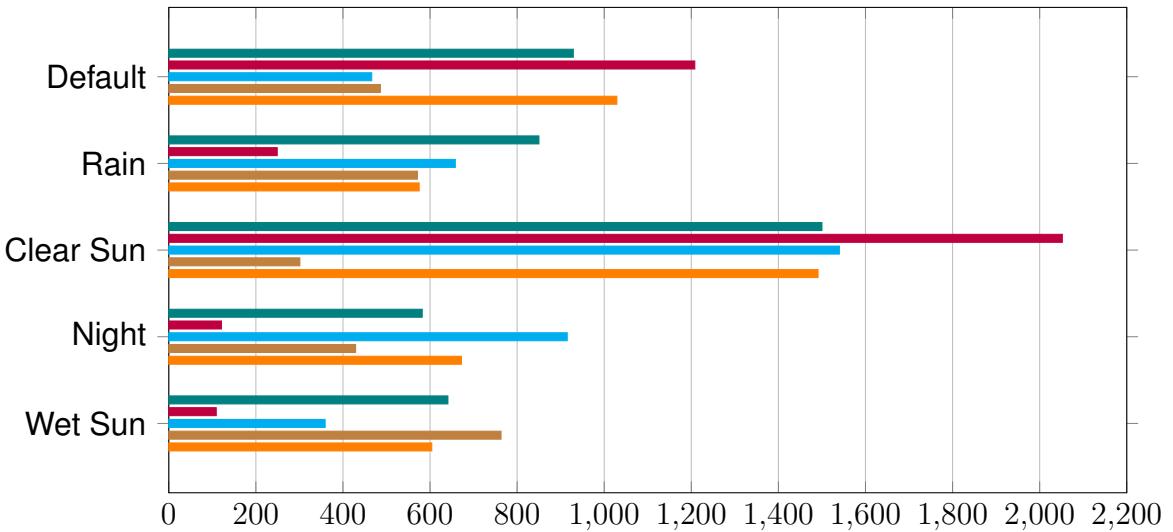


Abbildung 7.3: Streckenlängen aller 25 Testfahrten in m.

Die Datensätze Simulieren unterschiedliche Sichtverhältnisse. Es sei angemerkt das die Fahrzeuge im CARLA Simulator über keine eigenen Lichtquellen verfügen.

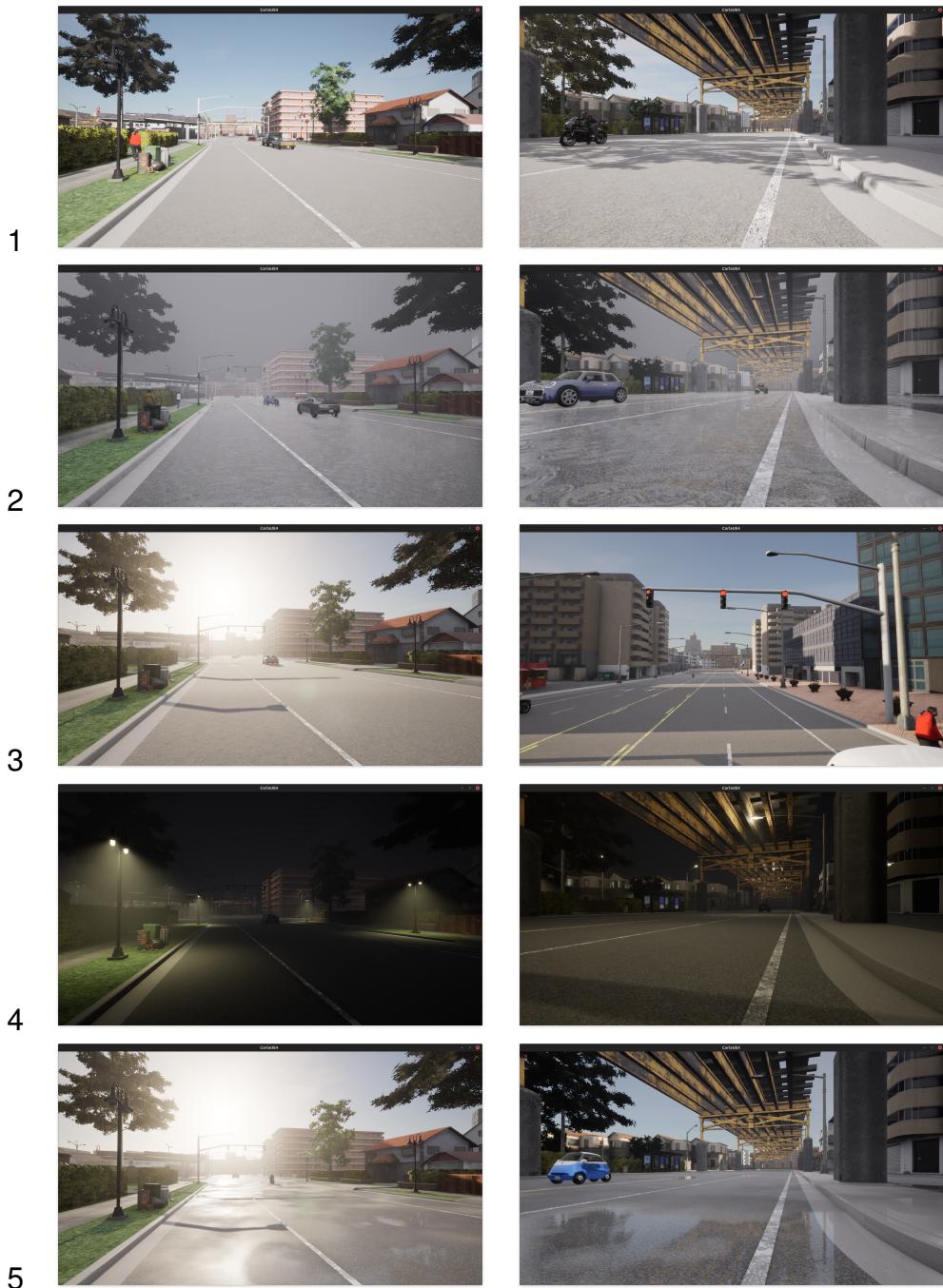


Abbildung 7.4: Witterungsverhältnisse der verschiedenen Datensätze: 1 Default, 2 hard rain, 3 clear sunset, 4 clear night, 5 wet sunset

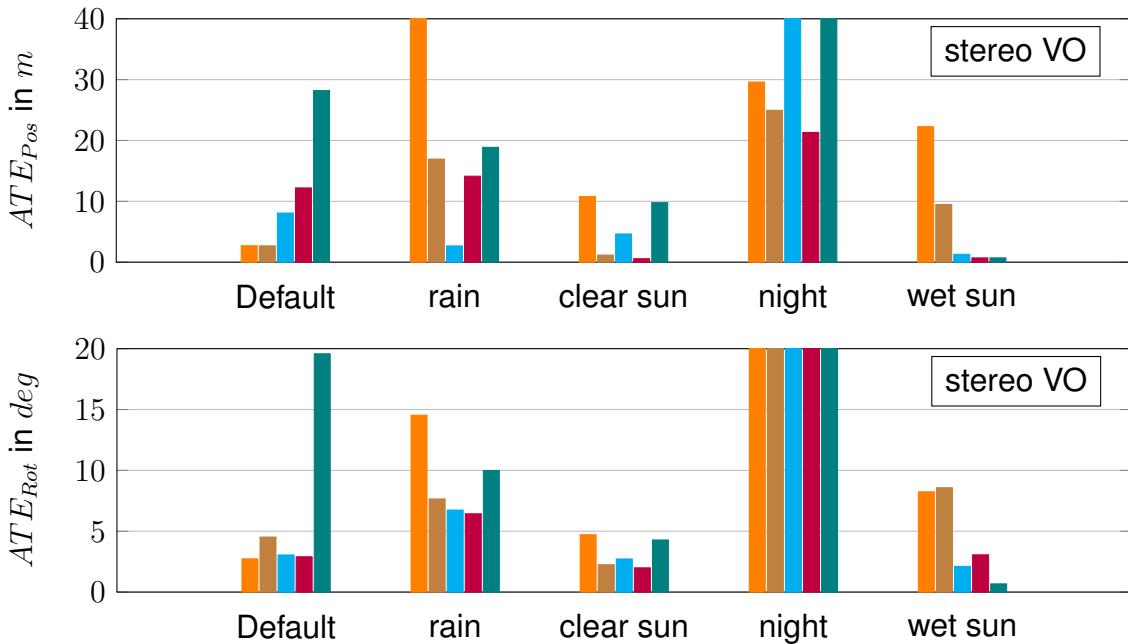


Abbildung 7.5: Absolute Trajectory Error (ATE) für Stereo VO der geschätzten Translation und des geschätzten Winkels in Grad für alle 25 Testfahrten.

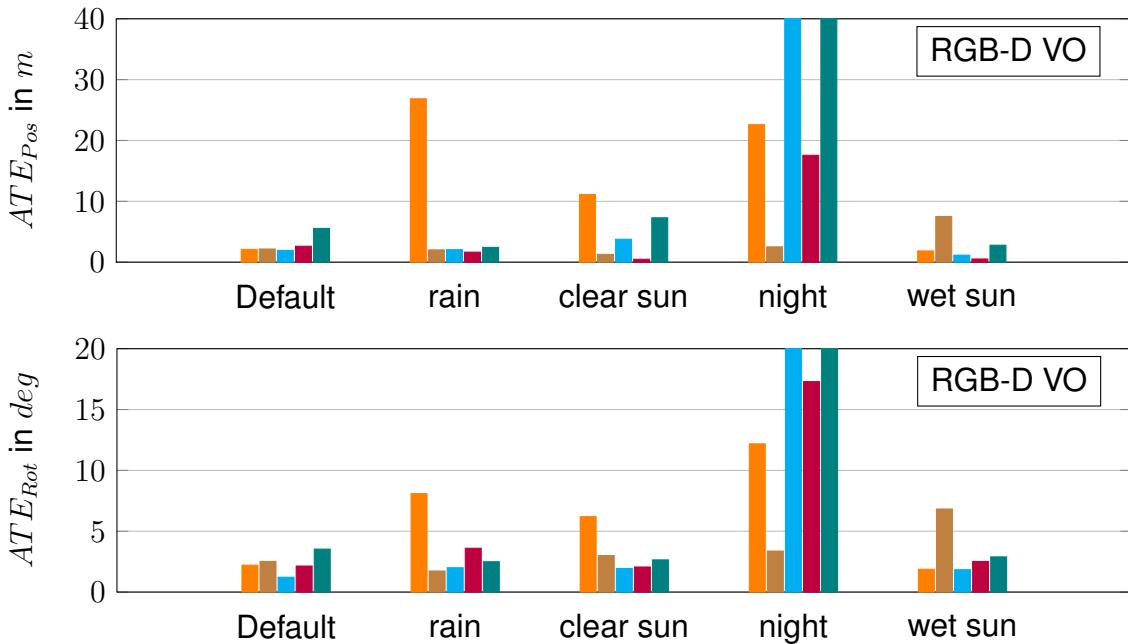


Abbildung 7.6: Absolute Trajectory Error (ATE) für RGB-D Visuelle Odometrie der geschätzten Translation und des geschätzten Winkels in Grad für alle 25 Testfahrten.

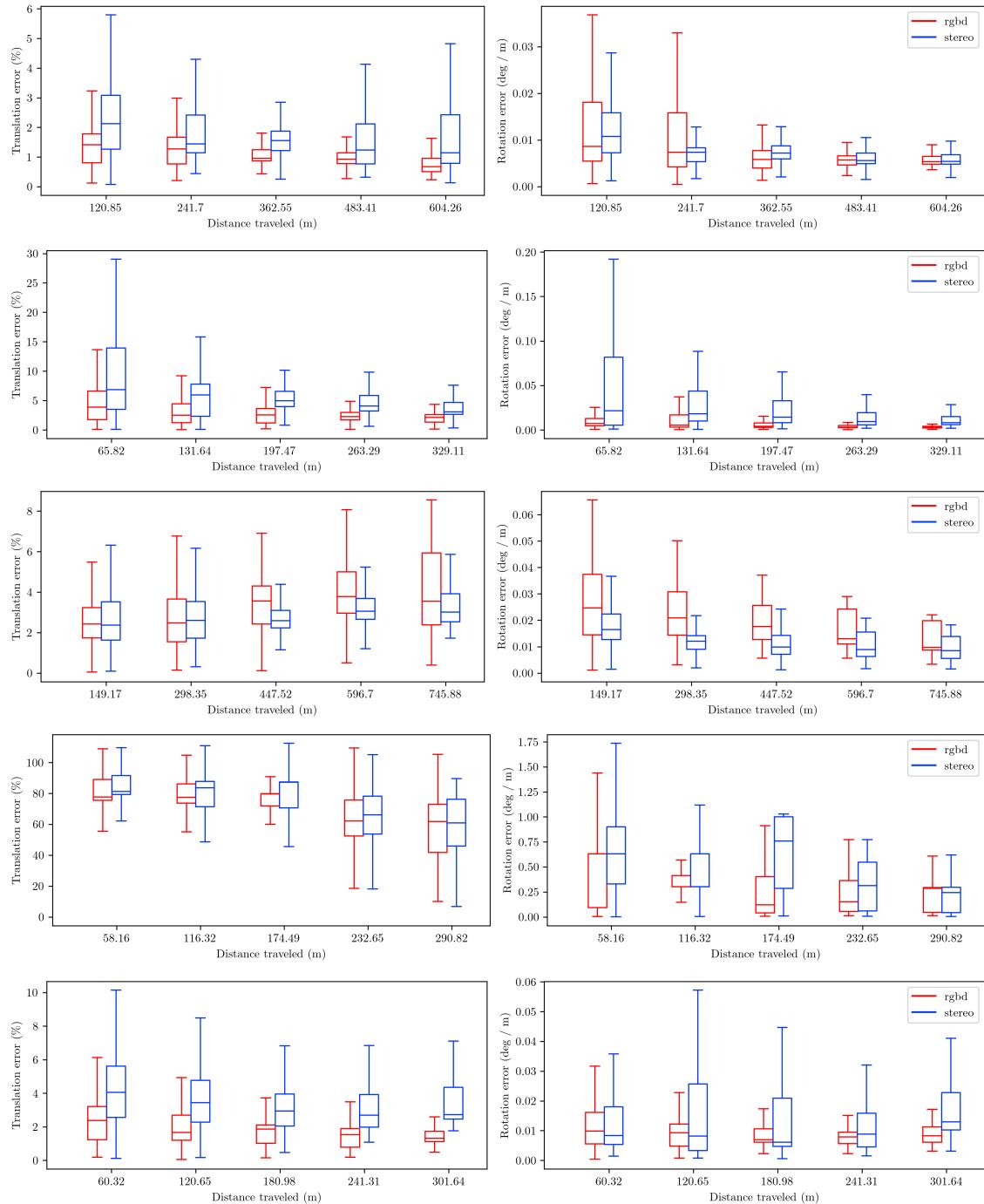


Abbildung 7.7: Um den Trend in jedem Datensatz zu veranschaulichen wurden die Absoluten Fehler für je eine Strecke aus jedem Datensatz geplottet. Von oben nach unten: 1 Default, 2 hard rain, 3 clear sunset, 4 clear night, 5 wet sunset

Median des relativen Rotations Fehlers in deg/m					
	Default	rain	clear sunset	night	wet sunset
(1) Stereo	0.9452	4.0144	3.4640	2.9886	2.1125
(1) RGBD	1.3338	3.1555	6.4847	0.8182	1.5687
(2) Stereo	1.2672	1.0465	1.3600	3.2212	2.1369
(2) RGBD	1.1546	1.0868	2.0310	2.5042	1.0776
(3) Stereo	0.7136	1.8220	2.3278	36.9868	0.7482
(3) RGBD	0.6619	1.6254	1.5180	38.5830	1.1404
(4) Stereo	2.1637	2.4429	1.7825	53.7278	2.4827
(4) RGBD	1.5119	3.3972	2.0200	11.9804	2.5498
(5) Stereo	1.5989	4.1391	1.5751	167.3668	1.2671
(5) RGBD	3.2175	2.1387	2.0686	34.6302	1.8053

Tabelle 7.1: Median des relativer Fehlers der Rotation

Median des Relativen Fehlers der Translation in %					
	Default	rain	clear sunset	night	wet sunset
(1) Stereo	2.2018	24.9510	6.8918	20.9637	3.6919
(1) RGBD	1.6438	4.76527	9.4637	4.7189	1.7361
(2) Stereo	2.4804	1.3636	0.9239	16.9190	2.6304
(2) RGBD	1.8756	1.7771	0.9684	2.2069	2.0401
(3) Stereo	1.4768	5.1681	4.1606	61.4019	0.8780
(3) RGBD	1.2665	1.6829	3.2752	63.2587	0.9144
(4) Stereo	2.6105	3.8872	0.0473	16.1462	0.1121
(4) RGBD	2.1176	1.3754	0.1263	9.0187	0.0566
(5) Stereo	17.3331	6.1409	3.4013	55.5880	0.9190
(5) RGBD	5.3631	1.4550	4.1606	48.8802	2.2763

Tabelle 7.2: Median des Relativen Fehlers der Translation

	RMSE der Rotation in deg/m				
	Default	rain	clear sunset	night	wet sunset
(1) Stereo	2.7334	14.5396	4.7107	29.6143	8.2440
(1) RGBD	2.2146	8.0921	6.1959	12.1838	1.8756
(2) Stereo	4.5297	7.6523	2.2452	20.4506	8.5746
(2) RGBD	2.5232	1.7378	3.0055	3.3728	6.8255
(3) Stereo	3.0501	6.7407	2.7294	52.7385	2.1195
(3) RGBD	1.2257	2.0066	1.9411	48.2168	1.8383
(4) Stereo	2.9097	6.4370	1.9908	58.5951	3.0643
(4) RGBD	2.1487	3.6022	2.0612	17.2936	2.5188
(5) Stereo	19.5840	9.9986	4.2870	168.5980	2.0029
(5) RGBD	3.2175	2.4937	2.6455	51.9567	2.8930

Tabelle 7.3: RMSE der Rotation

	RMSE der Translation in %				
	Default	rain	clear sunset	night	wet sunset
(1) Stereo	2.7093	66.6148	10.7996	44.9045	22.2712
(1) RGBD	2.1084	26.8571	11.1097	22.5937	1.8512
(2) Stereo	2.6822	16.9418	1.1682	24.9654	9.4863
(2) RGBD	2.1633	2.0260	1.2843	2.5186	7.5126
(3) Stereo	8.0751	26.6375	4.6390	86.9499	1.2902
(3) RGBD	1.9449	2.0570	3.76853	75.9970	1.1535
(4) Stereo	12.2844	14.1325	0.5691	21.3222	0.6728
(4) RGBD	2.6251	1.6480	0.4805	17.5666	0.5251
(5) Stereo	28.2357	18.8744	9.7872	91.4595	1.2994
(5) RGBD	5.5522	2.4321	7.2941	83.7440	2.7893

Tabelle 7.4: RMSE der Translation

7.3 Diskussion

Die RMSE Tabellen zeigen die Standardabweichung für die Schätzfehler der beiden Visuellen Odometrie Systeme RGB-D und Stereo Kamera. Die Standardabweichung zeigt wie sehr die Schätzungen um den Mittelwert Streuen. Die Median Tabellen zeigen die gemittelten Fehlerwerte mit geringeren Bewertung für Ausreißer.

Es ist sofort ersichtlich, dass die implementierte visuelle Odometrie unbrauchbare Ergebnisse für Nachtfahrten liefert. Es wäre interessant zu sehen, in wie fern sich die Ergebnisse für Fahrten mit Frontscheinwerfern ändern. Da Straßen, insbesondere Asphalt, eine Textur arme Oberflächentextur bieten, würden die Kamerabilder aus der RGB Kamera vermutlich dennoch nicht genug Merkmale beinhalten.

Der Schätzfehler mit Stereokamera streut stärker als für den RGB-D außer für 'Clear Sunset' wo ein sehr niedriger Sonnenstand vorherrscht.

In der Auswertung des Gesamtstreckenfehlers zeigt sich, dass der RGB-D Kameraaufbau insgesamt kleinere Schätzfehler hat. Mit weniger Streuung ist es auch das Robustere System.

Keines der Systeme zeigte Auffälligkeiten bei Ein- und Ausfahrten des in der Karte integrierten Tunnel. Bei hohen Geschwindigkeiten ($> 80 \text{ km/h}$) kam es vermehrt zum Verlust der Schätzung, vor allem auf Streckenabschnitten mit wenigen markanten Umgebungsmerkmalen.

Insgesamt konnte gezeigt werden, dass eine Lokalisierung des Fahrzeugs alleine durch Visuelle Odometrie erfolgreich durchgeführt werden konnte. Beide Systeme liefern i.d.R. bei Tageslicht weniger als 10 Grad Winkelfehler. Die Visuelle Odometrie mit RGB-D Kamera schafft in 17 von 25 Teststrecken einen absoluten Streckenfehler von weniger als 5 Metern.

7.4 Ausblick

Das System kann als Front-End für einen SLAM Algorithmus verwendet werden. Bei einigen Testfahrten hat sich die Odometrie von einer großen Fehlschätzung vor allem nie 'erholt', vor allem dann, wenn der Fehler zu Beginn auftrat. Durch SLAM könnten solche Verläufe stabilisiert werden und das gesamte Verfahren würde Robuster werden.

Eine weitere, naheliegende Möglichkeit wäre eine Sensorfusion mit einer IMU. Sogenannte Visual Inertial Odometry zeigt in aktuellen Forschungsergebnissen mehrere Paper gute Ergebnisse für eine Fehlerkorrektur und damit eine Reduktion der Drift.

Literaturverzeichnis

- [1] D. Fox. Introduction. <https://www.cs.cmu.edu/afs/cs/project/jair/pub/volume11/fox99a-html/node1.html> (accessed: 16.01.2022), 1999.
- [2] Fraundorfer F. Scaramuzza D. Visual odometry [tutorial]. *IEEE Robot. Automat. Mag.*, 18, 2011.
- [3] Zisserman A. Hartley R. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.
- [4] O. Schreer. *Stereoanalyse und Bildsynthese*. Springer Verlag Berlin Heidelberg, 2005.
- [5] H. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293, 1981.
- [6] A. et al. Agarwal. Building rome in a day. *2009 IEEE 12th International Conference on Computer Vision*, 2009.
- [7] Yang N. et al. Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. In *European Conference on Computer Vision (ECCV)*, September 2018.
- [8] Stephens M. Harris C. A combined corner and edge detector, 1988.
- [9] Rosten E. et al. Faster and better: A machine learning approach to corner detection. *Transactions on Pattern Analysis and Machine Intelligence*, 32, 2010.
- [10] OpenCV Documentation. Fast algorithm for corner detection. https://docs.opencv.org/4.x/d9/d0c/tutorial_py_fast.html (accessed: 08.05.2022).
- [11] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60, 2004.
- [12] D. Lowe. Object recognition from local scale-invariant features. *The proceedings of the seventh IEEE international conference on Computer Vision*, 2, 1999.
- [13] Bay H. et al. SURF: Speeded up robust features. *Computer Vision-ECCV*, 2, 2006.
- [14] Frahm J.-M. Heinly J. *Comparative Evaluation of Binary Features*. University of North Carolina, 2012.

- [15] Calonder M. et al. Brief: Binary robust independent elementary features. *Computer Vision-ECCV*, 2010.
 - [16] Rublee E. et al. Orb: an efficient alternative to sift or surf. *IEEE International Conference on. IEEE Computer Vision (ICCV)*, 2011.
 - [17] et. al. Leutenegger S. Brisk: Binary robust invariant scalable keypoints. *IEEE International Conference on. IEEE Computer Vision (ICCV)*, 2011.
 - [18] P. Rosin. Measuring corner properties. *Comput. Vis. Image Underst.*, 73, 1999.
 - [19] S. Merolini. Active pixel sensor vs ccd. who is the clear winner? https://meroli.web.cern.ch/lecture_cmos_vs_ccd_pixel_sensor.html (accessed: 02.05.2022).
 - [20] Gargan. Wikimedia commons. https://commons.wikimedia.org/wiki/File:Aps_pd_pixel_schematic.svg (accessed: 02.05.2022).
 - [21] Intel. Intel realsense™ product family d400 series datasheet. <https://www.intelrealsense.com/wp-content/uploads/2020/06/Intel-RealSense-D400-Series-Datasheet-June-2020.pdf> (accessed: 24.04.2022).
 - [22] Liu P. et al. Towards robust visual odometry with a multi-camera system. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.
 - [23] Mhiri R. et al. Visual odometry with unsynchronized multi-cameras setup for intelligent vehicle application. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, 2014.
 - [24] Eckstein. Vzense dcam710 tof rgbd long range depth camera module. <https://eckstein-shop.de/Vzense-DCAM710-TOF-RGBD-Long-Range-Depth-Camera-Module-EN> (accessed: 01.05.2022).
 - [25] Mouats T. et al. Performance evaluation of feature detectors and descriptors beyond the visible. *Journal of Intelligent and Robotic Systems*, 92, 2018.
 - [26] OpenCV Documentation. Feature matching - basics of brute force matcher. https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_matcher/py_matcher.html (accessed: 18.03.2022).
 - [27] H. Hirschmuller. Stereo processing by semiglobal matching and mutual information. *Transactions on Pattern Analysis and Machine Intelligence, IEEE*, 30, 2008.
 - [28] A. et al. Dosovitskiy. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, 2017.
-

- [29] Yousef W. Elsayed A. Matlab vs. opencv: A comparative study of different machine learning algorithms, 2019.
- [30] Michaud F. Labb   M. RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation, 2018.
- [31] Scaramuzza D. Zhang Z. Rethinking trajectory evaluation for slam: a probabilistic, continuous-time approach, 2019.
- [32] Scaramuzza D. Zhang Z. A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.
- [33] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13, 1991.