HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

MASTERARBEIT

Frau B.Sc.
Mahsa Khabbazi

Spezifikation, Implementierung und Test einer Kamera-basierten Hinderniserkennung für den JetBot unter Verwendung der Structure-From-Motion-Technik

Mittweida, 2025

Fakultät Ingenieurwissenschaften

MASTERARBEIT

Spezifikation, Implementierung und Test einer Kamera-basierten Hinderniserkennung für den JetBot unter Verwendung der Structure-From-Motion-Technik

Autor:
Frau B.Sc.

Mahsa Khabbazi

Studiengang:
Elektrotechnik/Automation

Seminargruppe:
EA22wA-M

Erstprüfer:
Prof. Dr.-Ing. Jan Thomanek

Zweitprüfer:
Prof. Dr.-Ing. Daniel Kriesten

Einreichung:
Mittweida, 10.03.2025

Bibliografische Beschreibung:

Khabbazi, Mahsa:
Spezifikation, Implementierung und Test einer Kamera-basierten Hinderniserkennung für den JetBot unter Verwendung der Structure-from-Motion-Technik. - 2025. - IX, 77, S.
Mittweida, Hochschule Mittweida, Fakultät Ingenieurwissenschaften, Masterarbeit, 2025

Referat:

Das vorliegende Masterarbeit befasst sich mit der Anwendung der Structure-from-Motion-Methode zur Hinderniserkennung mithilfe einer Fischaugenkamera. Ziel war es, eine 3D-Rekonstruktion der Umgebung zu erstellen und Hindernisse basierend auf der generierten Punktwolke zu identifizieren.

Inhalt

Abbildungsverzeichnis

Tabellenverzeichnis

Mathematischen Verzeichnis

Symbol	Beschreibung
$\hat{\mathbf{x}}_{\mathbf{C}}, \hat{\mathbf{y}}_{\mathbf{C}}$	Verzerrten Koordinaten
$\bar{\boldsymbol{x}}_{\mathbf{1}}$	2D-Punkt im Bild der Kamera C1
$\bar{\boldsymbol{x}}_{\mathbf{2}}$	2D-Punkt im Bild der Kamera C2
B	Die Basislinie (Abstand zwischen den beiden Kameras).
C1,C2	Kamerazentrum
$\mathbf{c_x}$,$\mathbf{c_y}$	Pixelkoordinaten des Hauptpunkts (optisches Zentrum)
D	Die Disparität
E	Essential Matrix
$\mathbf{e_1}$,$\mathbf{e_2}$	Epipol
f	Die reale Brennweite der Kamera ist.
F	Fundamentalmatrix
$\mathbf{f_x}$, $\mathbf{f_y}$	Brennweiten in x- und y-Richtung, die unterschiedliche Pixel-Seitenverhältnisse ermöglichen
K	Kalibrierungsmatrix
$\mathbf{k_1}$, $\mathbf{k_2}$	Die radialen Verzerrungskoeffizienten
$\mathbf{l_1}$,$\mathbf{l_2}$	Epipolare Line
P	Projektionsmatrix
$\mathbf{R}_{\mathbf{glob}}$	ist die globale Rotationsmatrix der Kamera im Weltkoordinatensystem.
R	Die Rotationsmatrix ist, die die Orientierung der Kamera beschreibt.
$\mathbf{R_{est}}$	ist die geschätzte Rotationsmatrix aus der Essentiellen Matrix.
s	Schiefewert, der auftritt, wenn der Bildsensor nicht exakt orthogonal zur optischen Achse montiert ist

s_w, s_h	Die Sensorgröße in mm darstellen.
t_{glob}	ist der globale Translationsvektor der Kamera im Weltkoordinatensystem.
T	Der Translationsvektor ist, der die Verschiebung der Kamera angibt.
t	Translationsmatrix
t_{est}	ist der geschätzte Translationsvektor aus der Essentiellen Matrix.
w, h	Die Auflösung des Bildsensors in Pixeln sind.
x_c	Die Koordinaten des Punktes im Weltkoordinatensystem sind.
X_L	3D-Koordinaten eines Punktes im Koordinatensystem der Kamera C1
X_R	3D-Koordinaten eines Punktes im Koordinatensystem der Kamera C2
x_s	Die Koordinaten des Punktes im Kamerakoordinatensystem sind
Z	Die Tiefe des Punktes in der realen Welt.

Abkürzungsverzeichnis

BFMatcher	Brute-Force Matcher
BRIEF	Binary Robust Independent Elementary Features
CLAHE	Contrast Limited Adaptive Histogram Equalization
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
FAST	Features from Accelerated Segment Test
FTS	Fahrerlose Transportsysteme
FTS	Fahrerlose Transportsysteme
KNN Match	K-Nearest Neighbors Matching
LiDAR	Light Detection and Ranging
MVG	Mehransichten-Geometrie (Multiple View Geometry)
ORB	Oriented FAST and Rotated BRIEF
RANSAC	Random sample consensus
SfM	Struktur-aus-Bewegung (Structure from Motion)
SIFT	Scale-Invariant Feature Transform
SURF	Speeded-Up Robust Features
SVD	Singulärwertzerlegung (Singular value decomposition)

1 Einleitung

Die Bildverarbeitungstechnologie ist heute zu einem leistungsfähigen Werkzeug für die 3D[1]-Modellierung und Rekonstruktion realer Umgebungen geworden. Eine der fortschrittlichsten Methoden in diesem Bereich ist Structure from Motion (SfM), die die Möglichkeit bietet, 3D-Informationen aus 2D[2]-Bildern zu extrahieren. Diese Methode basiert auf der Analyse von Bildern, die aus verschiedenen Blickwinkeln aufgenommen wurden, und dem Einsatz fortschrittlicher Computertechniken, um gleichzeitig die Position der Kamera und die 3D-Struktur der Szene zu schätzen. Diese Methode kann in verschiedenen Bereichen eingesetzt werden, z. B. bei Autonomes Fahren, Fahrerlose Transportsysteme(FTS) Roboter, Kartierungen und sogar in der Architektur und Kulturerbe.

In diesem Projekt wird die Implementierung von SfM unter Verwendung einer Fischaugenkamera zur Hinderniserkennung untersucht.

1.1 Problemstellung

Mobile Roboter müssen ihre Umgebung richtig verstehen, um in komplexen Umgebungen effektiv arbeiten zu können. Eine der größten Herausforderungen in diesem Bereich besteht darin, Hindernisse genau zu erkennen und ihre Entfernung zum Roboter abzuschätzen, um Kollisionen mit Hindernissen zu vermeiden. Derzeit werden zu diesem Zweck verschiedene Instrumente wie Lasersensoren (LiDAR[3]) oder Ultraschallsensoren (Ultrasonic) eingesetzt. Diese Instrumente sind jedoch oft teuer, haben eine begrenzte Genauigkeit oder sind empfindlich gegenüber den Umgebungsbedingungen.

Andererseits ist, wie Johnson in seinem Artikel sagte, der Einsatz von Kameras aufgrund ihrer geringeren Kosten, der einfachen Installation und der Bereitstellung umfassender visueller Informationen zu einer attraktiven Lösung geworden. Kameras allein sind jedoch nicht in der Lage, die Umgebung in 3D zu erfassen, und es müssen fortschrittliche Bildverarbeitungsalgorithmen eingesetzt werden. Eine der herausragenden Methoden in diesem

[1] 3D: Drei dimensional

[2] 2D: Zwei dimensional

[3] LiDAR nutzt Laserimpulse, um Entfernungen zu messen und detaillierte 3D-Modelle der Umgebung zu erstellen.

Bereich ist Structure from Motion, die durch die Analyse von 2D-Bildern das 3D-Modell der Umgebung rekonstruieren kann. [Johnson, et al., 2014]

Das Hauptproblem besteht darin, dass die Implementierung von SfM aufgrund der Komplexität der Bildverarbeitung spezielle Kenntnisse und eine Feinabstimmung der Kamera erfordert. Darüber hinaus bringt der Einsatz von Fischaugenkameras zur Verbesserung der Umgebungserfassung zusätzliche Herausforderungen mit sich, wie z. B. die komplexe Kalibrierung und das Management von Bildverzerrungen. Dieses Projekt wird versuchen, diese Probleme zu überwinden.

1.2 Aufgabenbeschreibung und Ziel der Arbeit

Diese Arbeit hat das Ziel, ein kamerabasiertes Hinderniserkennungssystem für den JetBot zu entwickeln und zu evaluieren. Konkret sollen folgende Inhalte bearbeitet werden:

1. **Einarbeitung in die Bildverarbeitung und SfM-Technik:**
    - Verständnis der Grundlagen der Bildverarbeitung und der SfM-Methode
    - Untersuchung von Algorithmen zur 3D-Rekonstruktion aus 2D-Bildern
2. **Softwareentwicklung mit C++ und OpenCV:**
    - Einarbeitung in die Bibliothek OpenCV
    - Nutzung der OpenCV-Bibliothek für Bildverarbeitungsaufgaben
    - Implementierung der Hinderniserkennungssoftware
3. **Experimente zur Validierung:**
    - Durchführung von Experimenten, um die Wirksamkeit der Hinderniserkennung zu testen
    - Untersuchung der Genauigkeit und Zuverlässigkeit der erzeugten 3D-Modelle

1.3 Gliederung der Arbeit

Diese Arbeit besteht aus sieben Kapiteln. **Kapitel 1** führt in das Thema ein, beschreibt die Problemstellung und das Ziel der Arbeit. Zudem wird ein Überblick über den Aufbau der Arbeit gegeben.

Im **Kapitel 2** werden die theoretischen Grundlagen erläutert, die für das Verständnis der Methodik notwendig sind. Dazu gehören die Kamerakalibrierung, die Bildverarbeitung

sowie die Structure from Motion (SfM)-Methode. Diese Grundlagen sind essenziell, um die spätere Implementierung der Hinderniserkennung nachvollziehen zu können.

Aufbauend auf diesen Grundlagen wird in **Kapitel 3** das Konzept des Hinderniserkennungssystems vorgestellt. Es werden die eingesetzten Algorithmen und die verwendete Hardware, insbesondere der JetBot und die Fischaugenkamera, erläutert. Zudem werden Herausforderungen bei der Umsetzung diskutiert. Ein Blockschaltbild fasst die wesentlichen Schritte des Projekts zusammen und dient als Leitfaden für das nachfolgende Kapitel.

Das darauf folgende **Kapitel 4** widmet sich der Implementierung des Systems. Hier werden die technische Umsetzung, die Softwarearchitektur sowie die Verarbeitung der Videodaten detailliert beschrieben. Wichtige Schritte wie die Feature-Extraktion, die Bewegungsberechnung der Kamera und die 3D-Rekonstruktion werden im Detail erklärt.

Im Anschluss daran werden in **Kapitel 5** die durchgeführten Experimente und die Validierung der Methode präsentiert. Es wird untersucht, wie genau die Hinderniserkennung funktioniert und welche Faktoren die Ergebnisse beeinflussen. Zudem werden die Herausforderungen der realen Anwendung analysiert und Optimierungsmöglichkeiten aufgezeigt.

Kapitel 6 fasst die wichtigsten Erkenntnisse der Arbeit zusammen und diskutiert mögliche Weiterentwicklungen, um die Präzision und Stabilität des Systems zu verbessern. Abschließend bietet **Kapitel 7** eine kompakte Zusammenfassung der gesamten Arbeit und einen Ausblick auf zukünftige Forschungsansätze.

2 Theoretische Grundlagen

Die Fähigkeit, aus zweidimensionalen Bildern eine dreidimensionale Rekonstruktion zu erstellen, spielt eine zentrale Rolle in der Bildverarbeitung und Robotik. In diesem Kapitel werden die theoretischen Grundlagen behandelt, die für das Verständnis der in dieser Arbeit eingesetzten Methoden notwendig sind. Es wird erläutert, wie eine Kamera funktioniert, wie 3D-Objekte aus der realen Welt in 2D-Bilder umgewandelt werden und wie die ursprüngliche 3D-Struktur mithilfe von SfM rekonstruiert werden kann.

Durch die Betrachtung eines Objekts aus mindestens zwei Perspektiven kann dessen Abstand zur Kamera berechnet werden – ein Prinzip, das auch beim menschlichen Sehen genutzt wird. Die entstehenden Bilder basieren auf einer Projektion, bei der die dreidimensionale Welt durch die Linse der Kamera auf den zweidimensionalen Kamerasensor abgebildet wird. Dabei gehen die Tiefeninformationen verloren, was eine nachträgliche Rekonstruktion erforderlich macht.

2.1 Kamera Modell und Kalibrierung

Um zuverlässige 3D-Informationen aus Bildern zu extrahieren, ist ein präzises Verständnis des Kameramodells erforderlich. Jede Kamera besitzt eine bestimmte geometrische Struktur, die bei der Abbildung der Umgebung auf den Bildsensor berücksichtigt werden muss. Die Kalibrierung spielt dabei eine entscheidende Rolle, da sie es ermöglicht, die internen und externen Parameter der Kamera zu bestimmen und Verzerrungen zu korrigieren.

2.1.1 Lochkamera

Herkömmliche Kameras basieren auf dem Lochkameramodell, das häufig als perspektivisches Modell in der Bildverarbeitung genutzt wird. Dieses Modell ist besonders praktisch für die mathematische Beschreibung von Projektionen und den Entwurf von Algorithmen, da es nur geringe Verzerrungen aufweist und somit die aufgenommenen Bilder leicht interpretierbar sind. In einer physikalischen Lochkamera (Physical Camera Model) tritt das Licht durch eine kleine Öffnung in eine lichtdichte Box ein und projiziert das Bild der Außenwelt auf die gegenüberliegende Seite der Kamera. Da das Licht sich in geraden Linien bewegt, erscheint das Bild auf der Bildebene seitenverkehrt und kopfüber.

In der mathematischen Modellierung (Mathematical Camera Model) wird die Bildebene oft vor den Brennpunkt verschoben, um die Berechnungen zu vereinfachen. Beide Ansätze sind jedoch äquivalent, solange die Bildkoordinaten entsprechend transformiert werden. (Abbildung 2-1) veranschaulicht dieses Konzept, indem sie die grundlegende Struktur einer

Lochkamera zeigt. Hierbei wird die Abbildung durch die Größe der Öffnung, die Position der Bildebene und den Abstand zum Objekt bestimmt.

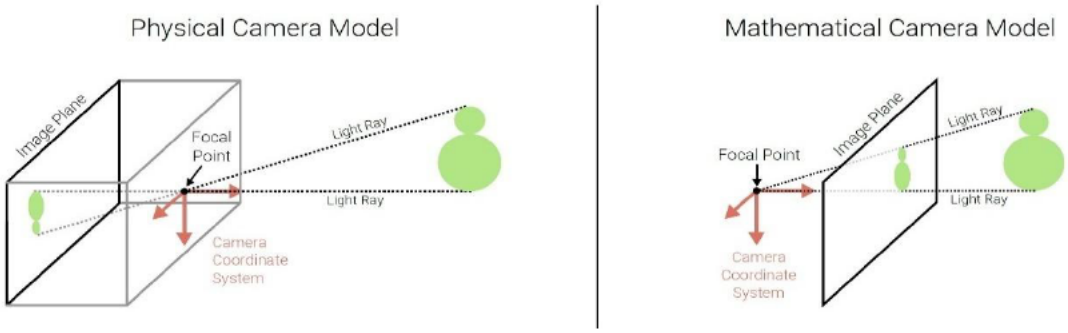

Abbildung 2-1: Lochkameras [Geiger, 2021]

Neben der grundlegenden Funktionsweise der Lochkamera gibt es zwei zentrale Methoden, wie dreidimensionale Objekte auf eine zweidimensionale Bildebene projiziert werden können. Diese Methoden bestimmen, wie die Größenverhältnisse und Perspektiven im Bild dargestellt werden.

- Die orthografische Projektion
- Die perspektivische Projektion

(Abbildung 2-2) stellt diese beiden Projektionsarten gegenüber. Während die orthografische Projektion oft für technische Zeichnungen verwendet wird, da sie Längenverhältnisse exakt erhält, sorgt die perspektivische Projektion für eine realistischere Darstellung der Szene, da Objekte in größerer Entfernung kleiner erscheinen.

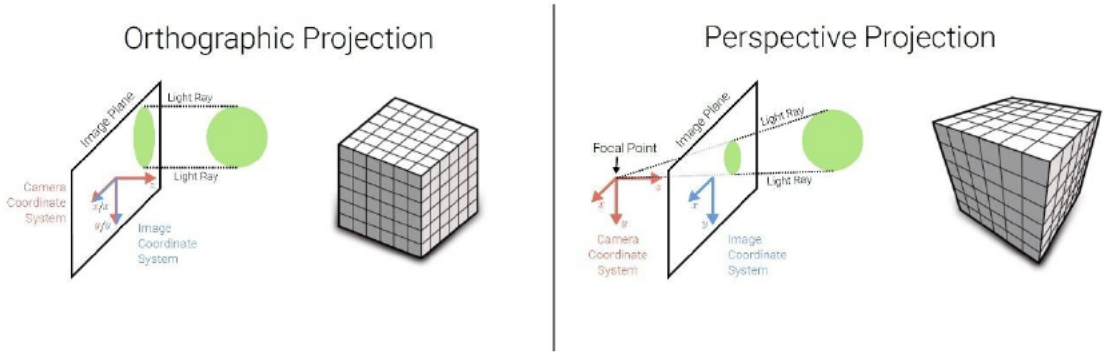

Abbildung 2-2: Projektionsmodelle [Geiger, 2021]

1. **Orthographische Projektion:** Bei der orthografischen Projektion wird ein 3D-Punkt $\boldsymbol{x_c} \in \mathbb{R}^3$ auf Pixelkoordinaten $\boldsymbol{x_s} \in \mathbb{R}^2$ verlaufen die Lichtstrahlen parallel zur z-Achse des Kamerakoordinatensystems, sodass keine perspektivische Verzerrung auftritt. Infolgedessen bleiben die **x**- und **y**-Koordinaten unverändert, während die **z**-Koordinate weggelassen wird.
   Dieses Modell wird häufig in technischen Anwendungen verwendet, da es die tatsächlichen Längenverhältnisse eines Objekts beibehält (Abbildung 2-3) veranschaulicht dieses Konzept, indem die parallele Projektion der Lichtstrahlen auf die Bildebene dargestellt wird.

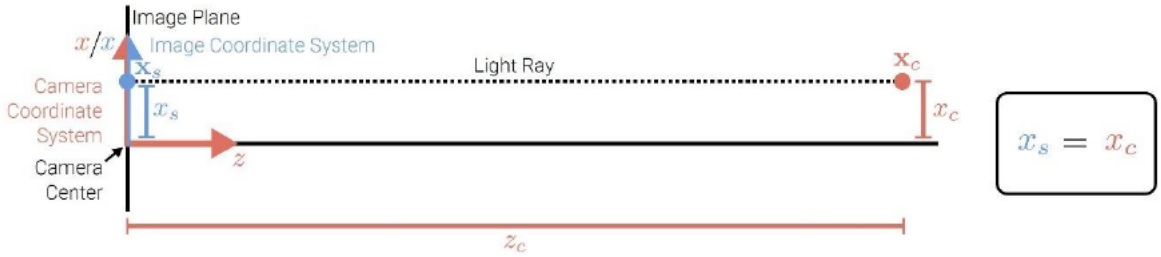

Abbildung 2-3: Orthographische Projektion [Geiger, 2021]

2. **Perspektivische Projektion**: Die perspektivische Projektion unterscheidet sich von der orthografischen Projektion dadurch, dass die Lichtstrahlen nicht parallel verlaufen, sondern durch das Kamera-Zentrum, den Pixel $\boldsymbol{x_s}$ und den 3D-Punkt $\boldsymbol{x_c}$geführt werden. Dadurch entsteht eine Abbildung, bei der Objekte in größerer Entfernung kleiner erscheinen, während näher gelegene Objekte vergrößert dargestellt werden. Dies entspricht der natürlichen Wahrnehmung des menschlichen Auges und ermöglicht eine realistische Darstellung von Szenen in der Bildverarbeitung. (Abbildung 2-4) zeigt dieses Konzept, indem die Richtung der Lichtstrahlen und die daraus resultierende Projektion auf die Bildebene visualisiert werden [Geiger, 2021].

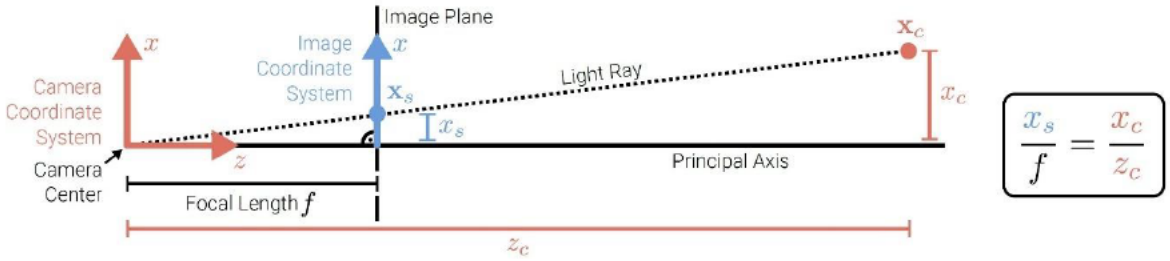

Abbildung 2-4: Perspektivische Projektion [Geiger, 2021]

In der Praxis müssen Weltkoordinaten, die häufig in physikalischen Einheiten wie Metern angegeben werden, so skaliert werden, dass sie auf den Bildsensor der Kamera passen, der in Pixeln misst. Ein Skalierungsfaktor wird angewendet, um die Größenverhältnisse der realen Welt präzise auf die Bildkoordinaten zu übertragen. Dadurch werden die Weltkoordinaten in eine kompaktere, pixelbasierte Darstellung umgerechnet, sodass reale Objekte im Bild korrekt dargestellt werden können. Dies ist entscheidend für Anwendungen wie die 3D-Rekonstruktion oder Bildverarbeitung.

Zusätzlich wird während dieses Prozesses der Koordinatenursprung der realen Welt in das Kamerakoordinatensystem transformiert. Für die Darstellung auf der Bildebene wird der Koordinatenursprung jedoch, wie in der gezeigten (Abbildung 2-5) in die obere linke Ecke des Bildschirms verschoben. Laut dem Modell der Lochkamera-Projektion besteht die Beziehung zwischen einem 3D-Punkt und seinem entsprechenden 2D-Bildpunkt aus Zwei Komponenten, die wie folgt beschrieben werden:

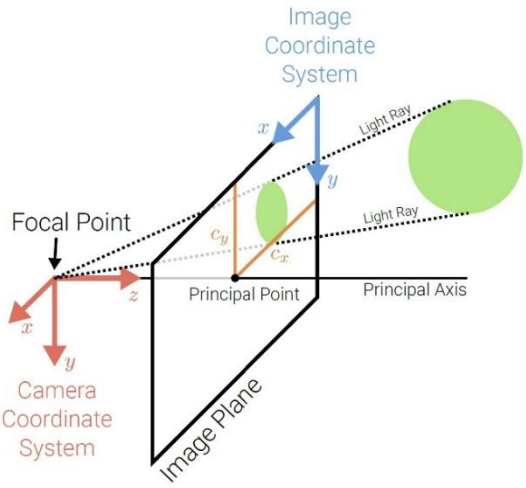

Abbildung 2-5: Koordinatenursprung [Geiger, 2021]

2.1.1.1 Extrinsische Parameter:

Während die perspektivische Projektion die Abbildung eines 3D-Punktes auf die Bildebene beschreibt, wird für eine vollständige Modellierung der Kamera auch die räumliche Position und Orientierung der Kamera im Weltkoordinatensystem benötigt. Diese werden durch die extrinsischen Kameraparameter definiert.

Die erste Komponente ist die starre Körpertransformation in homogene Koordinaten, die Punkte $\mathbf{x_c}$ im Weltkoordinatensystem mit Punkten $\mathbf{x_s}$ im Kamerakoordinatensystem verknüpft. Dabei beschreibt die 3×3-Rotationsmatrix **R** die Ausrichtung der Kamera während der Translationsvektor **T** mit drei Elementen die Position des Weltursprungs im Kamerakoordinatensystem repräsentiert.

Die mathematische Darstellung dieser Transformation wird durch Formel (2-1) ausgedrückt:

$$\begin{bmatrix} \boldsymbol{x_s} \\ \boldsymbol{y_s} \\ \mathbf{1} \end{bmatrix} \sim [\boldsymbol{R} \quad \boldsymbol{T}]. \begin{bmatrix} \boldsymbol{x_c} \\ \boldsymbol{y_c} \\ \boldsymbol{z_c} \\ \mathbf{1} \end{bmatrix} \tag{2-1}$$

wobei:

- X_s die Koordinaten des Punktes im Kamerakoordinatensystem sind,
- X_c die Koordinaten des Punktes im Weltkoordinatensystem sind,
- R die Rotationsmatrix ist, die die Orientierung der Kamera beschreibt,
- T der Translationsvektor ist, der die Verschiebung der Kamera angibt.

Zusammen bilden diese Parameter die extrinsische Kameramatrix, die als 3×4-Matrix definiert ist. Sie spielt eine zentrale Rolle in der Kamerakalibrierung und 3D-Rekonstruktion, da sie bestimmt, wie ein Punkt aus der realen Welt in das Kamerakoordinatensystem transformiert wird.

(Abbildung 2-6) veranschaulicht diese Transformation, indem sie die Beziehung zwischen den Weltkoordinaten, dem Kamerakoordinatensystem und den extrinsischen Parametern grafisch darstellt.

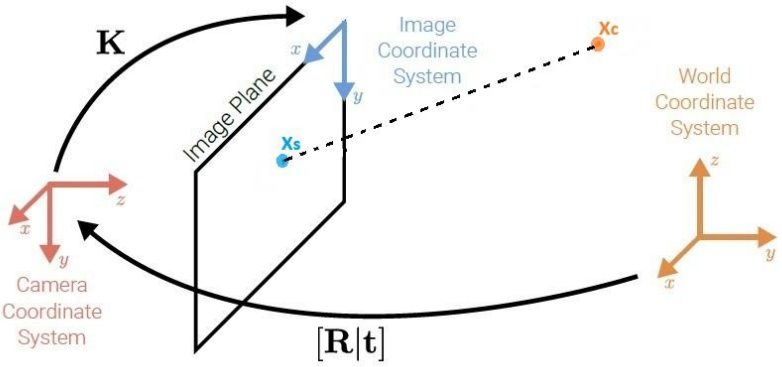

Abbildung 2-6: Extrinsische Parameter-(modifiziert) [Geiger, 2021]

2.1.1.2 Intrinsische Parameter

Nachdem die extrinsischen Parameter die räumliche Position der Kamera definiert haben, werden nun die intrinsischen Parameter betrachtet, die die internen Eigenschaften der Kamera beschreiben. Die Hauptaufgabe dieser Parameter besteht darin, die Abbildung von 3D-Weltkoordinaten auf 2D-Pixelkoordinaten zu ermöglichen.

Die Transformation erfolgt in zwei Schritten:

1. Die Projektion von 3D-Koordinaten auf die Bildebene der Kamera
2. Die anschließende Umwandlung in Pixelkoordinaten

Durch die perspektivische Projektion entsteht eine 3×3-Matrix, die als Kalibrierungsmatrix **K** bezeichnet wird. Diese Matrix enthält die intrinsischen Parameter der Kamera und wird durch Formel (2-2) definiert:

$$\boldsymbol{K} = \begin{bmatrix} \boldsymbol{fx} & \boldsymbol{s} & \boldsymbol{c_x} \\ \boldsymbol{0} & \boldsymbol{f_y} & \boldsymbol{c_y} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} \end{bmatrix} \tag{2-2}$$

Hierbei:

f_x, f_y: Brennweiten in x- und y-Richtung, die unterschiedliche Pixel-Seitenverhältnisse ermöglichen

s: Schiefewert, der auftritt, wenn der Bildsensor nicht exakt orthogonal zur optischen Achse montiert ist

c_x,c_y: Pixelkoordinaten des Hauptpunkts (optisches Zentrum)

Brennweite und Sensorauflösung: Die Brennweiten $\boldsymbol{f_x}$ und $\boldsymbol{f_y}$, hängen von der tatsächlichen Brennweite der Kamera sowie von der Pixelgröße ab. Sie werden mit Formel (2-3) berechnet:

$$\boldsymbol{f_x = \frac{f \cdot w}{s_w}} \qquad \boldsymbol{f_y = \frac{f \cdot h}{s_h}} \tag{2-3}$$

wobei:

- f die reale Brennweite der Kamera ist
- w, h die Auflösung des Bildsensors in Pixeln sind
- s_w, s_h die Sensorgröße in mm darstellen

Hauptpunkt der Kamera: Der Hauptpunkt einer Kamera befindet sich normalerweise in der Bildmitte und wird durch Formel (2-4) berechnet:

$$\boldsymbol{c_x = \frac{w}{2}, \qquad c_y = \frac{h}{2}} \tag{2-4}$$

Hierbei gibt $\mathbf{c_x}$, $\mathbf{c_y}$ die Pixelkoordinaten des optischen Zentrums an.

Kombination von intrinsischen und extrinsischen Parametern: Nachdem sowohl die extrinsischen als auch die intrinsischen Parameter definiert wurden, kann die gesamte Transformation eines Punktes von Weltkoordinaten in das Pixelkoordinatensystem beschrieben werden. Diese vollständige Transformation erfolgt mit Formel (2-5):

$$\boldsymbol{X_c = K\,[R\ \ t]\,.X_s = PX_s} \tag{2-5}$$

wobei **P** die Projektionsmatrix ist, die durch die intrinsischen und extrinsischen Parameter definiert wird. Diese wird mit Formel (2-6) dargestellt:

$$\boldsymbol{P = K\,[R\ \ t]} \tag{2-6}$$

(Abbildung 2-6) stellt die Zusammenhänge zwischen den einzelnen Parametern grafisch dar und zeigt, wie die Punkte aus der realen Welt in das Kamerakoordinatensystem und schließlich in das Bildkoordinatensystem transformiert werden.

Die Projektionsmatrix **P** ist entscheidend für die 3D-Rekonstruktion und viele Anwendungen in der Bildverarbeitung. Eine detailliertere Erklärung dieser Matrix erfolgt später in Abschnitt 2.2.3.

Herkömmliche Kameras besitzen ein begrenztes Sichtfeld, was ihre Einsatzmöglichkeiten in bestimmten Anwendungen einschränken kann. Eine Kamera mit einem größeren Sichtfeld ist jedoch besonders nützlich für viele Roboteranwendungen und autonome Fahrzeuge, da sie eine umfassendere Erfassung der Umgebung ermöglicht [Jonathan Courbon, 2012].

2.1.2 Fischaugen Kamera

In vielen Anwendungen der Robotik und Bildverarbeitung ist es vorteilhaft, ein möglichst großes Sichtfeld zu erfassen, um eine umfassende Umgebungserkennung zu ermöglichen. Herkömmliche Kameras haben jedoch ein begrenztes Sichtfeld, was ihre Anwendbarkeit einschränkt.

Ein Fisheye-Objektiv ist ein Kameraobjektiv mit einer kurzen Brennweite und einem großen Sichtfeld, das Szenen mit einem Blickwinkel von fast oder sogar mehr als 180 Grad erfassen kann. Aufgrund dieser Eigenschaften entstehen bei Aufnahmen mit einer Fisheye-Kamera starke Verzerrungen, die durch die optischen Prinzipien bedingt sind. Daher erfordert die Verarbeitung solcher Bilder eine präzise geometrische Modellierung und Kalibrierung, um Verzerrungen zu korrigieren und die Bilder für weitere Anwendungen nutzbar zu machen [Yi Hou, 2020].

Fischaugenkameras werden aufgrund ihres großen Sichtfelds in verschiedenen Bereichen wie autonomen Fahrzeugen, Überwachungssystemen und Robotik eingesetzt. Sie ermöglichen eine umfassende Erfassung der Umgebung, allerdings führen die starken Verzerrungen dazu, dass spezielle Algorithmen zur Korrektur und Verarbeitung der Bilddaten erforderlich sind.

2.1.3 Verzerrungsarten

Kameras mit Fischaugenobjektiven ermöglichen eine breite Erfassung der Umgebung, jedoch führen ihre optischen Eigenschaften zu starken Verzerrungen, die in der Bildverarbeitung berücksichtigt werden müssen. Grundsätzlich unterscheidet man zwei Arten von Verzerrungen:

- **Radiale Verzerrungen**:
  Bei radialen Verzerrungen erscheinen gerade Linien im Bild gekrümmt. Diese Art der Verzerrung ist bei Fischaugenkameras besonders ausgeprägt, da die konvexe Form des Objektivs und der große Betrachtungswinkel eine deutliche Krümmung der Bildgeometrie verursachen. Während die Verzerrung in der Bildmitte meist gering ist, nimmt sie zu den Bildrändern hin deutlich zu.
- **Tangentiale Verzerrungen**:
  Diese Verzerrung tritt auf, wenn das Kameraobjektiv nicht exakt auf die Bildebene ausgerichtet ist. Falls das Objektiv leicht verkippt oder nicht perfekt zentriert ist, entstehen asymmetrische Verzerrungen, die sich über das gesamte Bild erstrecken.
  (Abbildung 2-7) stellt die verschiedenen Verzerrungsarten dar, einschließlich der radialen und tangentialen Verzerrung, die in Kameras auftreten können.

Abbildung 2-7: radiale- und Tangentiale Verzerrung [Geeksforgeeks, 2024]

Da in diesem Projekt eine Fischaugenkamera verwendet wird, bei der vor allem die radiale Verzerrung eine große Rolle spielt, wird ausschließlich diese Art der Verzerrung berücksichtigt. Tangentiale Verzerrungen werden in diesem Projekt nicht analysiert.

Mathematische Beschreibung der radialen Verzerrung:

Radiale Verzerrung bewirkt eine Verschiebung der Bildpunkte, entweder in Richtung des Bildzentrums (Barrel-Distortion) oder vom Bildzentrum weg (Pincushion-Distortion). Die Stärke dieser Verzerrung hängt von der Entfernung des jeweiligen Punktes zur Bildmitte ab. Die Verzerrung wird durch eine Polynombasierte Modellierung beschrieben, wobei die Verzerrungskoeffizienten eine zentrale Rolle spielen. Formel (2-7) beschreibt die mathematische Beziehung zwischen den verzerrten Koordinaten $\hat{x}_c, \hat{y}_c$ und den ursprünglichen Pixelkoordinaten ($\mathbf{x_c}$, $\mathbf{y_c}$):

$$\begin{aligned} \hat{x}_c &= x_c\left(1 + k_1 r_c^2 + k_2 r_c^4\right) \\ \hat{y}_c &= y_c\left(1 + k_1 r_c^2 + k_2 r_c^4\right) \end{aligned} \tag{2-7}$$

wobei:

- $r_c^2 = x_c^2 + y_c^2$. der quadratische Abstand zum Bildzentrum ist,
- k_1, k_2 die radialen Verzerrungskoeffizienten sind.

In (Abbildung 2-8) sind die Unterschiede zwischen Barrel-Distortion (negative radiale Verzerrung) und Pincushion-Distortion (positive radiale Verzerrung) dargestellt. Während bei der Barrel-Distortion die Bildpunkte zum Zentrum hin verschoben werden, erfolgt bei der Pincushion-Distortion die Verschiebung nach außen.

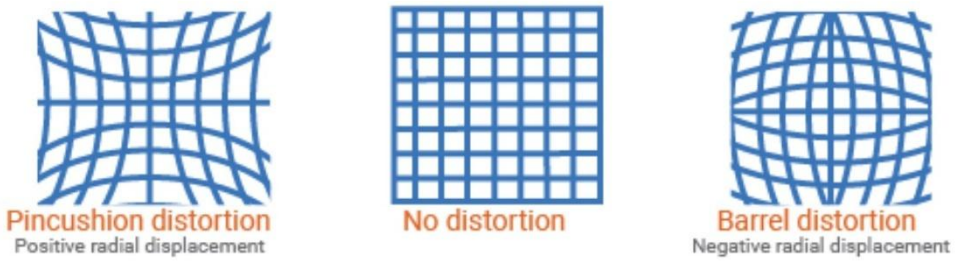

Abbildung 2-8: Radiale Verzerrungstyp [Mathworks, 2024]

Korrektur der radialen Verzerrung:

Nach der Korrektur der radialen Verzerrung werden die endgültigen Pixelkoordinaten mit Formel (2-8) berechnet [Szeliski, 2022]:

$$\begin{aligned} x_s &= f\hat{x}_c + c_x \\ y_s &= f\hat{y}_c + c_y \end{aligned} \tag{2-8}$$

Hierbei:

- x_s, y_s sind die korrigierten Pixelkoordinaten auf dem Bildsensor.
- f_x, f_y repräsentieren die skalierte Brennweite in x- und y-Richtung.
- c_x, c_y sind die Pixelkoordinaten des Hauptpunkts (optisches Zentrum der Kamera).

(Abbildung 2-9) zeigt die korrigierten Bildpunkte nach Anwendung dieser Gleichung, wobei die Verzerrung entfernt wurde und das Bild einer perspektivischen Kamera ähnlicher wird.

Bisher wurden die entscheidenden Faktoren für die Umwandlung von realen Koordinaten in Pixelkoordinaten erläutert. Im nächsten Abschnitt wird untersucht, wie die Kamera kalibriert wird, um diese Parameter exakt zu bestimmen.

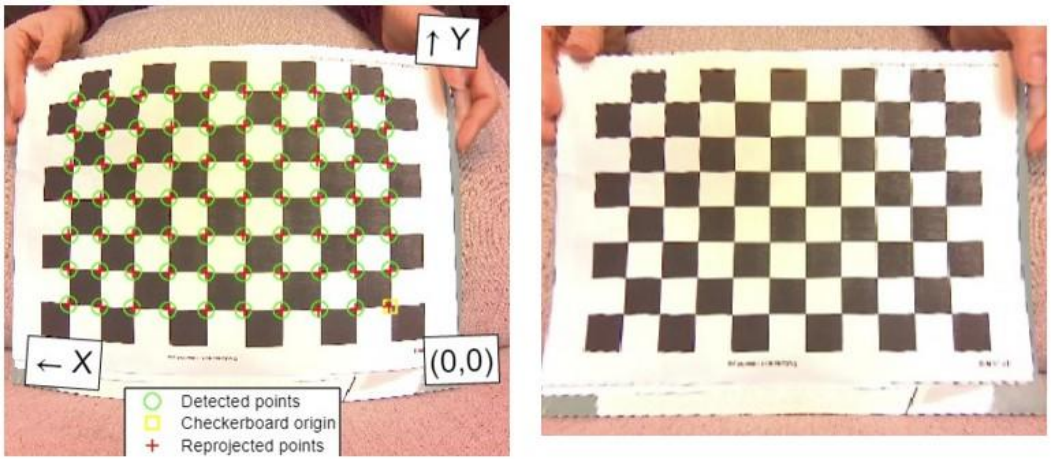

Abbildung 2-9: Entzerrung

2.1.4 Kalibrierung

Wie bereits in Abschnitt 2.1.1 erläutert, müssen bestimmte Parameter, insbesondere die intrinsischen Kameraparameter sowie die Verzerrungsparameter, genau bestimmt werden, um ein reales 3D-Objekt mit einer Kamera in ein korrektes 2D-Bild umzuwandeln. Die Kamerakalibrierung ermöglicht es, diese Parameter zu ermitteln und die Abbildungsgenauigkeit zu optimieren.

Zur Durchführung der Kalibrierung stehen verschiedene Methoden und Algorithmen zur Verfügung. In der Praxis wird dieser Prozess häufig mit Programmiersprachen wie Python (unter Verwendung von OpenCV), C++ oder spezialisierten Software-Tools wie MATLAB durchgeführt.

Die Schritte zur Kamerakalibrierung lauten wie folgt:

1. **Aufnahme eines Referenzmusters**
   Die Kamera erfasst Bilder eines Referenzmusters, typischerweise eines Schachbrettmusters, aus verschiedenen Winkeln und Entfernungen. Dadurch wird sichergestellt, dass ausreichend Referenzpunkte in den Bildern sichtbar sind.
   Das Hauptziel dieses Schrittes besteht darin, bestimmte markante Punkte, insbesondere die Ecken des Schachbrettmusters, in den aufgenommenen Bildern zu identifizieren. Hierzu werden Bildverarbeitungsalgorithmen eingesetzt, um diese Punkte genau zu extrahieren.
   (Abbildung 2-10) zeigt ein Beispiel für ein Kalibrierungsbild mit identifizierten Referenzpunkten. Zudem wird die Position jedes Fotos relativ zur Kamera sowie der Reprojektionsfehler dargestellt.
   Der Reprojektionsfehler ist der Unterschied zwischen der vorhergesagten und der tatsächlichen Position der Punkte im Bild. Eine geringe durchschnittliche Abweichung (unter 0,5 Pixel) sowie eine geringe Anzahl von Punkten mit größeren Abweichungen deuten auf eine präzise Kalibrierung hin.

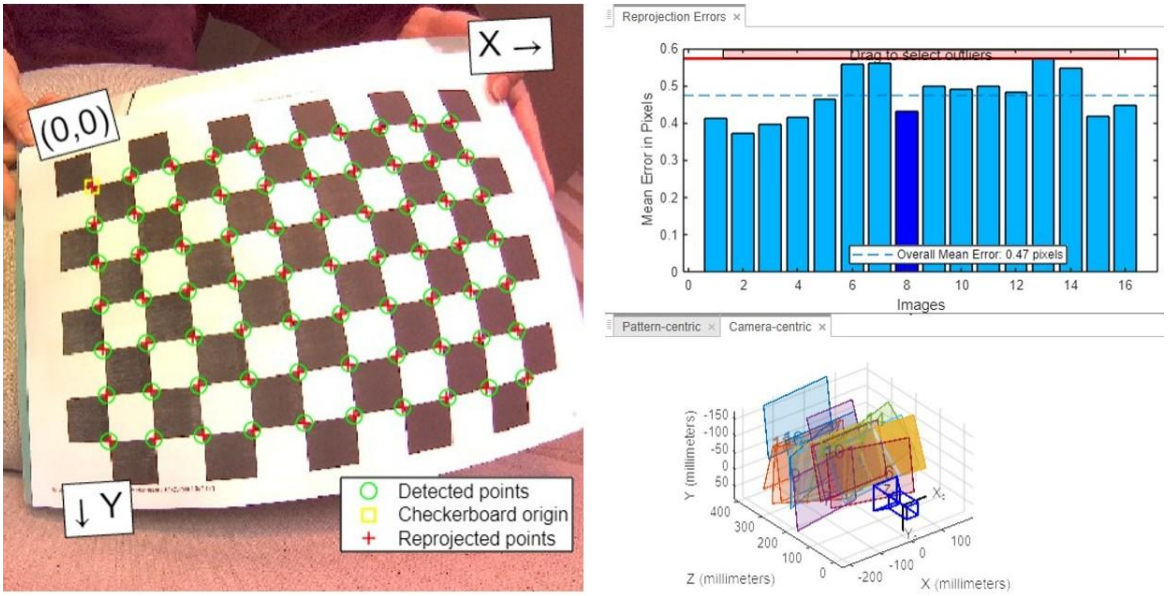

Abbildung 2-10: Kamerakalibrierung mit MATLAB

2. **Bestimmung der internen und externen Parameter**
   Nach der Erfassung der Referenzbilder werden die intrinsischen und extrinsischen Parameter der Kamera berechnet. Diese umfassen:
   - Brennweite(f_x, f_y)
   - Hauptpunkt (C_x, C_y)
   - Verzerrungskoeffizienten des Objektivs

   Diese Parameter beschreiben die internen Eigenschaften der Kamera und wurden bereits ausführlich in Abschnitt 2.1.1.2 behandelt.

Nach der erfolgreichen Kalibrierung der Kamera können die ermittelten Parameter genutzt werden, um Punkte vom globalen Koordinatensystem in das Kamerakoordinatensystem zu transformieren. Dadurch entsteht eine präzise Verbindung zwischen den realen 3D-Koordinaten und den daraus erzeugten 2D-Bildern.

Diese Kalibrierung bildet die Grundlage für komplexere geometrische Berechnungen in der Bildverarbeitung. Die exakte Koordinatentransformation ermöglicht es, die erfassten Bildinformationen effizient zu nutzen und weiterführende Schritte wie die 3D-Rekonstruktion präzise umzusetzen.

2.2 Bildverarbeitung und Feature-Extraktion

Bei der SfM-Methode wird versucht, sowohl die Kamerapositionen als auch die 3D-Positionen von Merkmalen aus einer Reihe von 2D-Bildern wiederherzustellen. Dazu werden Punkte in den Bildern identifiziert und miteinander verglichen. Durch diese Zuordnung kann die essentielle Matrix berechnet werden, welche die geometrische Beziehung zwischen den Kameraansichten beschreibt.

Diese Matrix wird in zwei Komponenten zerlegt:

- **R (Rotation):** Beschreibt die Drehung der Kamera zwischen zwei Bildern.
- **t (Translation):** Gibt an, wie sich die Kamera im Raum bewegt hat.

Sobald **R** und **t** bestimmt sind, lassen sich die 3D-Punkte der Szene berechnen, und eine Rekonstruktion der Szene aus den Bildinformationen wird möglich.

Damit dieser Prozess erfolgreich ist, müssen zunächst einige grundlegende Konzepte der Bildverarbeitung verstanden werden. Diese spielen eine wichtige Rolle, da sie eng mit der Merkmalserkennung und -zuordnung verbunden sind, die für eine präzise 3D-Rekonstruktion notwendig sind.

2.2.1 Feature-Erkennung und Matching

Der erste Schritt bei der 3D-Rekonstruktion besteht darin, korrespondierende Punkte in verschiedenen Frames zu identifizieren. Diese Keypoints stellen dieselben physikalischen Merkmale oder Objekte dar, die in mehreren Bildern aus unterschiedlichen Perspektiven erfasst wurden. Die genaue Erkennung und Zuordnung dieser Keypoints ist entscheidend, da sie die Grundlage für die spätere Berechnung der 3D-Struktur bildet Für diesen Prozess werden häufig fortschrittliche Bildverarbeitungsalgorithmen wie SIFT[4], SURF[5], oder ORB[6] verwendet. Diese Algorithmen ermöglichen es, markante Merkmale zu extrahieren und sie über die Bildfolgen hinweg abzugleichen.

Damit ein Punkt in verschiedenen Bildern zuverlässig erkannt werden kann, sollten seine Eigenschaften unveränderlich gegenüber perspektivischen Verzerrungen und Beleuchtungsänderungen sein. Dies stellt sicher, dass derselbe physische Punkt in der Szene unabhängig von der Kameraposition oder dem Blickwinkel dieselben Merkmalsvektoren erzeugt. Einfache Patches von RGB-Werten oder Luminanz Intensitäten sind dafür nicht geeignet, da sie empfindlich auf Änderungen reagieren. Aus diesem Grund sind fortschrittlichere Techniken zur Merkmalsextraktion erforderlich, um Bilder genau abzugleichen und eine präzise Rekonstruktion zu ermöglichen [Geiger, 2021].

(Abbildung 2-11) zeigt die Feature-Erkennung mit dem SIFT-Algorithmus, bei dem markante Punkte im Bild erkannt und mit den entsprechenden Punkten in anderen Bildern verglichen werden.

[4] SIFT ist ein Algorithmus zur Erkennung und Zuordnung robuster Bildmerkmale.

[5] SURF ist ein schneller Algorithmus zur Erkennung und Zuordnung von Bildmerkmalen.

[6] ORB ist ein effizienter und leichter Algorithmus zur Erkennung und Zuordnung von Bildmerkmalen.

Abbildung 2-11: Feature-Erkennung mit dem SIFT-Algorithmus

Da für dieses Projekt ein geeigneter Algorithmus ausgewählt werden muss, wurden verschiedene Methoden miteinander verglichen, um die optimale Lösung zu finden.

Scale-Invariant Feature Transform (SIFT):

Bei dieser Methode wird ein Bild in viele kleine Merkmalsvektoren umgewandelt, die gegen Veränderungen wie Verschiebung, Skalierung, Rotation und teilweise auch Helligkeitsänderungen unempfindlich sind.

Um diese skalierungsunabhängigen Merkmale zu finden, werden bestimmte Punkte im Bild gesucht, die in verschiedenen Größen besonders auffällig sind. Diese Punkte werden als Minima oder Maxima der Gaußschen Differenzfunktion erkannt. Für jeden gefundenen Punkt wird ein Merkmalsvektor erstellt, der die Umgebung dieses Punktes beschreibt. Diese sogenannten SIFT-Schlüssel sind robust gegenüber kleinen Veränderungen und Anpassungen im Bild.

Vorteile:

- **Hohe Präzision:** Zuverlässig bei Skalierung, Drehung, Helligkeitsänderungen und Rauschen.
- **Stabile Keypoints:** Liefert genaue Punkte, ideal für Abgleich und 3D-Rekonstruktion.

Nachteile:

- **Langsam:** Hoher Rechenaufwand, nicht echtzeitfähig.
- **Hohe Speicheranforderungen:** Große Deskriptoren, hoher Speicherbedarf.

Speeded-Up Robust Features(SURF)

Dieser Algorithmus funktioniert ähnlich wie SIFT, aber er ist jedoch schneller und effizienter. Er verwendet die Hessian-Matrix[7], um markante Punkte im Bild zu erkennen, und einen einfachen, verteilungsbasierten Deskriptor, um diese Punkte zu beschreiben. Dadurch eignet er sich gut für Echtzeitanwendungen. [Herbert Bay, 2006]

Vorteile:

- **Schneller als SIFT:** Nutzt Integralbilder für optimierte Verarbeitung. (Hessian-Matrix)
- **Effizient:** Bessere Wahl, wenn Geschwindigkeit entscheidend ist.

Nachteile:

- **Langsamer als ORB:** Nicht so schnell wie moderne Algorithmen wie ORB.
- **Schlechtere Leistung bei Lichtveränderungen:** Weniger robust bei Helligkeitsänderungen im Vergleich zu SIFT.

Oriented FAST and Rotated BRIEF(ORB)

ORB verbindet den FAST[8]-Keypoints-Detektor mit dem BRIEF[9]-Deskriptor und wurde so angepasst, dass er besser funktioniert. Zuerst findet FAST markante Punkte im Bild. Dann wählt das Harris-Maß[10] die wichtigsten Punkte aus. Außerdem wird eine Bildpyramide genutzt, um Merkmale in verschiedenen Größen zu erkennen. [OpenCV, 2025]

Vorteile:

- **Sehr schnell:** Optimiert für Echtzeit-Anwendungen, ideal für Embedded-Systeme.
- **Drehinvarianz:** Widerstandsfähig gegenüber Rotationen.

---

[7] Hessian-Matrix: Es handelt sich um eine quadratische Matrix, die aus den partiellen Ableitungen zweiter Ordnung einer skalar wertigen Funktion besteht. Diese Matrix beschreibt die lokale Krümmung einer mehrdimensionalen Funktion.

[8] FAST: (Features from Accelerated Segment Test) findet Keypoints (wie. Ecken) in einem Bild. Er sucht nach Pixeln, die sich stark von ihren Nachbarn unterscheiden.

[9] BRIEF: (Binary Robust Independent Elementary Features) beschreibt, wie die Umgebung eines Keypoints aussieht, indem er kleine Muster von Helligkeitsunterschieden verwendet.

[10] Harris-Maß:(Harris corner detector) wird verwendet, um die besten Ecken oder markanten Punkte in einem Bild auszuwählen.

Nachteile:

- **Weniger robust:** Nicht so präzise wie SIFT oder SURF bei großen Änderungen in Skalierung oder Helligkeit.
- **Keine gute Skalierungsinvarianz:** Weniger leistungsfähig bei starken Größenänderungen.

Sobald die Schlüsselpunkte identifiziert wurden, müssen sie in aufeinanderfolgenden Frames verglichen werden. Ziel ist es, herauszufinden, welche Punkte in einem Bild auch in einem anderen Bild vorhanden sind. Dieser Prozess wird als Feature Matching bezeichnet. Er ermöglicht es, Zusammenhänge zwischen den einzelnen Frames zu erkennen und zu bestimmen, welche Bereiche einer Szene in mehreren Bildern sichtbar sind.

Für diesen Abgleich wird der BF-Matcher (Brute-Force-Matcher) verwendet. Dieser Algorithmus vergleicht die Deskriptoren jedes Schlüsselpunkts mit denen der nachfolgenden Bilder, um die ähnlichsten Punkte zu finden. Bei diesem Algorithmus werden die Deskriptoren jedes Punktes mit denen der nachfolgenden Bilder verglichen, um den ähnlichsten Punkt zu finden. Der BF-Matcher vergleicht jeden Deskriptor in einem Bild mit allen Deskriptoren im nächsten Bild. Das beste Paar wird anhand eines Abstandsmaßes ermittelt (z. B. der euklidische Abstand für SIFT und SURF oder der Hamming-Abstand für ORB). Je kleiner der berechnete Abstand, desto wahrscheinlicher ist es, dass die beiden Punkte zur gleichen Eigenschaft in beiden Frames gehören. (Abbildung 2-12) zeigt das Ergebnis des Feature Matchings und verdeutlicht, wie Schlüsselpunkte zwischen Bildern verbunden werden.

Abbildung 2-12; Feature Matching zwischen zwei Frame

2.2.2 Epipolare Geometrie

Nachdem im vorherigen Abschnitt die Erkennung und Zuordnung von Merkmalen besprochen wurde, stellt sich die Frage, wie diese Punkte für die 3D-Rekonstruktion genutzt werden können. Um die geometrische Beziehung zwischen zwei Kamerabildern zu

beschreiben, wird das Konzept der Epipolargeometrie verwendet. Es hilft, den Rekonstruktionsprozess zu vereinfachen, indem es einschränkt, wo sich ein Punkt im zweiten Bild befinden muss, wenn er im ersten Bild erkannt wurde.

Die Epipolare Geometrie beschreibt die geometrische Beziehung zwischen zwei Bildern einer 3D-Szene, die aus unterschiedlichen Kamerapositionen aufgenommen wurden. Ihr Hauptziel besteht darin, die Suche nach korrespondierenden Punkten zu vereinfachen, indem sie die möglichen Positionen eines Punktes in der zweiten Ansicht auf eine Epipolarlinien beschränkt.

Ein Punkt **X** ist ein 3D-Punkt in der realen Welt, der von zwei Kamerapositionen erfasst wird. In den Bildebenen der Kameras erscheint dieser 3D-Punkt als zwei 2D-Projektionen:

- $\bar{x}_1$ in der Bildebene 1
- $\bar{x}_2$ in der Bildebene 2

Jede Kamera hat ein eigenes Kamerazentrum:

- C1 für Kameraposition 1
- C2 für Kameraposition 2

Die Ebene, die durch die beiden Kamerazentren und den 3D-Punkt **X** aufgespannt wird, wird als Epipolarebene bezeichnet. In (Abbildung 2-13) ist diese orange markiert. Die Verbindungslinie zwischen den beiden Kamerazentren wird als Baseline bezeichnet. Der Schnittpunkt dieser Linie mit den Bildebenen bildet die Epipol in jedem Bild:

- e_1 in Bild 1
- e_2 in Bild 2

Die Epipolarlinien ist die Linie in Bild 2, auf der sich die Entsprechung des Punktes $\bar{x}_1$ aus Bild 1 befinden muss. Sie entsteht durch den Schnitt der Epipolarebene mit der Bildebene. Dadurch muss der entsprechende Punkt nicht im gesamten Bild gesucht werden, sondern nur entlang der Epipolarlinien. (Abbildung 2-13) beschreibt die Epipolare Geometire.

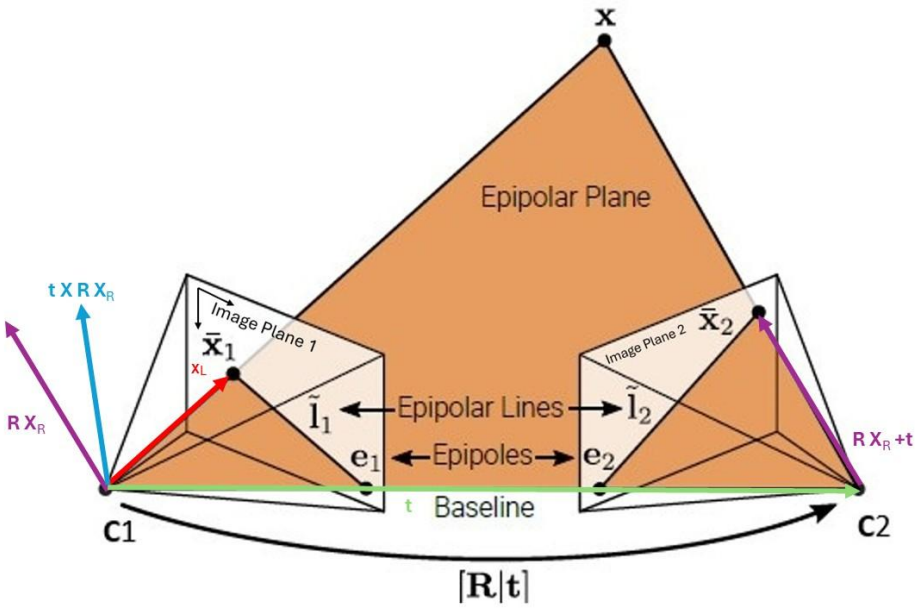

Abbildung 2-13: Epipolare Geometrie (modifiziert) [Geiger, 2021]

Mathematische Beschreibung der Epipolargeometrie

Um die Beziehung zwischen den beiden Kameras zu beschreiben, wird eine starre Transformation$[\boldsymbol{R}|\boldsymbol{t}]$ angenommen. Diese wird genutzt, um einen Punkt $\mathbf{X_R}$ im Koordinatensystem der Kamera $\mathbf{C_2}$ in das Koordinatensystem der Kamera $\mathbf{C_1}$ zu transformieren. Durch Berechnung des Kreuzprodukts

$$\boldsymbol{t} \times (\boldsymbol{R}\boldsymbol{x_R}) \tag{2-9}$$

erhält man einen Vektor, der senkrecht zur Epipolarebene steht. Da sich der Punkt $\mathbf{X_L}$ (rote Pfade) auf der Epipolarebene befindet, ergibt das Skalarprodukt mit diesem Vektor **Null**:

$$\boldsymbol{X_L} \cdot \left(\boldsymbol{t} \times (\boldsymbol{R}\boldsymbol{X_R})\right) = \mathbf{0} \tag{2-10}$$

Das Kreuzprodukt kann durch eine schiefsymmetrische Matrixdarstellung ausgedrückt werden:

$$\boldsymbol{x_L^T}[\boldsymbol{t}]_{\boldsymbol{X}}\boldsymbol{R}\boldsymbol{X_R} = \mathbf{0} \tag{2-11}$$

Diese Gleichung wird schließlich in eine kompakte Matrixdarstellung überführt:

$$\boldsymbol{X_L^T E X_R = 0} \tag{2-12}$$

Die resultierende Matrix **E** wird als Essential Matrix bezeichnet. Sie definiert eine Epipolare Einschränkung für alle Punktpaare zwischen den beiden Kameras, die auf denselben 3D-Punkt projiziert werden. Punktpaare, die diese Einschränkung nicht erfüllen, gelten als ungültige Korrespondenzen..

Die Essential Matrix wird anhand mehrerer Punktpaare geschätzt, die ein homogenes Gleichungssystem aufstellen erfolgt meist durch Singulärwertzerlegung (SVD[11]) oder Eigenwertzerlegung.

Fundamentalmatrix und Kamerakalibrierung

Bisher wurde angenommen, dass die Kameras **kalibriert** sind, also die Kalibrierungsmatrix **K** eine Einheitsmatrix ist. In der Realität müssen jedoch die tatsächlichen intrinsischen Kameraparameter berücksichtigt werden, da die Bilder spezifische Pixelgrößen und Brennweiten haben. Dazu wird die Inverse der Kalibrierungsmatrix **K** auf beide Seiten der Gleichung angewendet:

$$\bar{\boldsymbol{x}}_2^T \boldsymbol{E} \bar{\boldsymbol{x}}_1 = \boldsymbol{0}, \qquad \bar{\boldsymbol{x}}_2^T \boldsymbol{k}_2^{-T} \widetilde{\boldsymbol{E}} \boldsymbol{k}_1^{-1} \bar{\boldsymbol{x}}_1 = \bar{\boldsymbol{x}}_2^T \widetilde{\boldsymbol{F}} \bar{\boldsymbol{x}}_1 = \boldsymbol{0} \tag{2-13}$$

Die daraus hervorgehende Matrix $\widetilde{\boldsymbol{F}}$ wird als **Fundamentalmatrix** bezeichnet. Diese Matrix wird direkt aus einer ausreichenden Anzahl von Punktpaaren in Pixelkoordinaten geschätzt. Wenn die Kalibrierungsmatrizen $\mathbf{K_1}$ und $\mathbf{K_2}$ bekannt sind, wird daraus die Essential Matrix berechnet. In vielen Fällen reicht jedoch die Fundamentalmatrix aus, um als Epipolare Einschränkung zu dienen. [David Millán Escrivá, 2019]

Anwendung der Essential Matrix in der Bewegungsschätzung

Nachdem in zwei aufeinanderfolgenden Bildern ähnliche Punkte erkannt wurden, werden diese entlang der Epipolarlinien miteinander verknüpft. Für die Schätzung der relativen Bewegung zwischen den Frames ist eine Mindestanzahl von 8 guten Matches erforderlich. Der Grund dafür ist, dass die Essential Matrix 5 Freiheitsgrade besitzt (3 für die Rotation und 2 für die normalisierte Translation), und jedes Punktpaar eine unabhängige Gleichung liefert [Szeliski, 2022].

Durch die Visualisierung des Feature Matchings lassen sich mögliche Fehler in der Zuordnung sowie die Dichte der Übereinstimmungen analysieren. Sobald die Position der Punkte

[11] SVD: Singular value decomposition

und der Kameras bestimmt wurde, können die Rotation **R** und Translation **t** der Kamera relativ zur vorherigen Position durch Zerlegung der Essential Matrix berechnet werden. Die Berechnung erfolgt mit folgender Gleichung:

$$\boldsymbol{E} = [\boldsymbol{t}]_{\boldsymbol{X}}\boldsymbol{R} \tag{2-14}$$

2.2.3 Projektion:

Nachdem im vorherigen Abschnitt beschrieben wurde, wie die Epipolargeometrie hilft, korrespondierende Punkte in zwei Bildern zu bestimmen, stellt sich die Frage, wie aus diesen Bildpunkten die tatsächlichen 3D-Koordinaten berechnet werden können. Dafür werden Projektionsmatrizen verwendet, die die Beziehung zwischen einem 3D-Punkt in der realen Welt und seiner 2D-Projektion im Bild beschreiben.

Wenn zwei aufeinanderfolgende Bilder einer Szene vorliegen, können die Projektionsmatrizen P_1 und P_2 zur Rekonstruktion der 3D-Koordinaten verwendet werden. Zunächst wird die Projektionsmatrix des ersten Frames $\mathbf{P_1}$ als Referenzkoordinatensystem festgelegt. Dies bedeutet, dass in diesem Frame keine Rotation oder Translation vorliegt. Die Matrix ist daher definiert als:

$$\boldsymbol{P_1} = \boldsymbol{K}[\boldsymbol{I}|\boldsymbol{0}] \tag{2-15}$$

Hierbei:

- K ist die intrinsische Kameramatrix, die die internen Parameter der Kamera beschreibt.
- I ist die Einheitsmatrix, die zeigt, dass keine Rotation stattgefunden hat.
- 0 ist der Translationsvektor, der anzeigt, dass die Kamera im ersten Frame als fixiert betrachtet wird.

Im zweiten Frame muss die Projektionsmatrix $\mathbf{P_2}$ jedoch die zuvor berechnete Rotation und Translation der Kamera berücksichtigen. Die Matrix wird daher definiert als:

$$\boldsymbol{P_2} = \boldsymbol{K}[\boldsymbol{R}|\boldsymbol{t}] \tag{2-16}$$

Diese beiden Projektionsmatrizen P_1 und P_2 bilden die Grundlage für die Triangulation, mit der die 3D-Koordinaten der Bildpunkte berechnet werden können.

2.2.4 Triangulation

Im vorherigen Abschnitt wurde erläutert, wie die Projektionsmatrizen definiert werden und wie die Epipolargeometrie genutzt wird, um die Beziehung zwischen zwei Kamerabildern zu beschreiben. Aufbauend auf diesen Konzepten stellt sich nun die Frage, wie aus diesen Informationen die 3D-Koordinaten der Bildpunkte berechnet werden können.

Beim SfM-Verfahren werden die Kamerapositionen bestimmt und markante Punkte in der dreidimensionalen Szene extrahiert. Während die geometrische Beziehung zwischen zwei 2D Ansichten und der 3D Welt bereits untersucht wurde, ist die Methode zur genauen Rekonstruktion der 3D-Struktur aus 2D-Bildern noch nicht vollständig betrachtet worden. Ein zentrales Konzept in diesem Zusammenhang ist die Triangulation. Wird ein Punkt in zwei unterschiedlichen Ansichten erfasst, lassen sich die Strahlen, die von den optischen Zentren der Kameras durch die entsprechenden Bildpunkte verlaufen, rekonstruieren. Diese Strahlen erstrecken sich im dreidimensionalen Raum und schneiden sich idealerweise an der Position des gesuchten 3D-Punktes.

Sobald die Projektionsmatrizen P_1 und P_2 für zwei aufeinanderfolgende Frames definiert wurden, kann die Rekonstruktion der 3D-Koordinaten eines Punktes mithilfe der Triangulationsmethode durchgeführt werden. Diese Methode basiert auf der Tatsache, dass von den optischen Zentren der Kameras zwei Strahlen zu den entsprechenden Bildpunkten in den beiden Ansichten gezogen werden. Der gesuchte 3D-Punkt befindet sich dort, wo sich diese Strahlen schneiden.

Epipolargeometrie als Grundlage der Triangulation

Eine effektive Technik zur Berechnung der Tiefeninformationen nutzt dabei die Epipolargeometrie, wie in (Abbildung 2-14) dargestellt. Hierbei befindet sich ein Punkt aus Bild **L** auf einer bestimmten Epipolarlinien in Bild **R**. Diese Linie kann mathematisch exakt bestimmt werden, während die Herausforderung darin besteht, den Punkt in Bild **R** zu identifizieren, der am besten mit dem Punkt in Bild **L** übereinstimmt. Dieses Verfahren wird als stereoskopische Tiefenschätzung bezeichnet.

Da auf diese Weise Tiefeninformationen für nahezu jedes Pixel im Bild gewonnen werden können, handelt es sich in der Regel um eine dichte Rekonstruktion.

Um die Berechnungen zu vereinfachen, werden die Epipolarlinien so transformiert, dass sie exakt horizontal verlaufen. Dadurch kann die Punktzuordnung nur entlang der x-Achse erfolgen.

Ein entscheidender Vorteil dieser Konfiguration ist die sogenannte Disparität **d** - also der horizontale Versatz eines Punktes zwischen den beiden Bildern. Betrachtet man die geometrischen Zusammenhänge im Dreieck, ergibt sich folgende Beziehung:

$$\frac{B}{z} = \frac{d}{f} \tag{2-17}$$

Hierbei:

- B ist die Basislinie (Abstand zwischen den beiden Kameras).
- z ist die Tiefe des Punktes in der realen Welt.
- d ist die Disparität
- f ist die Brennweite der Kamera.

Durch Umstellen der Gleichung erhält man:

$$d = \frac{B \cdot f}{z} \tag{2-18}$$

Da sowohl die Basislinie **B** als auch die Brennweite **f** feste Werte sind führt dies zu einer zentralen Erkenntnis:

Die Disparität **d** ist umgekehrt proportional zur Tiefe **z**.

Das bedeutet:

- Je kleiner die Disparität, desto weiter entfernt ist der Punkt von der Kamera.
- Je größer die Disparität, desto näher befindet sich der Punkt zur Kamera.

Diese Beziehung zwischen Disparität und Tiefe bildet die Grundlage für alle stereoskopischen Algorithmen zur 3D-Rekonstruktion [David Millán Escrivá, 2019].

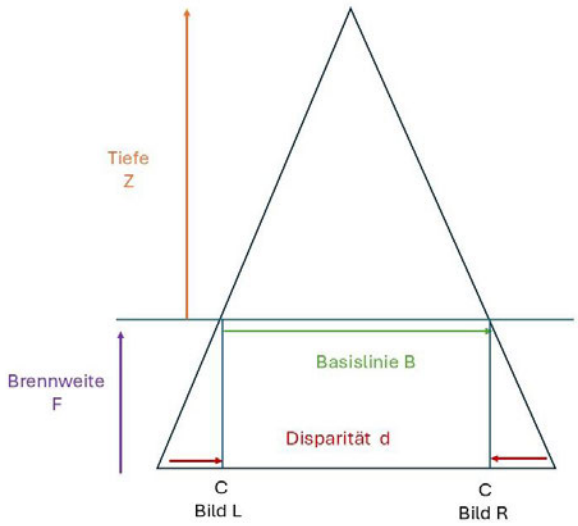

Abbildung 2-14: Triangulation [David Millán Escrivá, 2019]

Jetzt können die 3D-Positionen der Punkte bestimmt werden, wodurch die räumliche Struktur der Szene sichtbar wird. Damit ist das Hauptziel der SfM-Methode erreicht, da die Szene nun dreidimensional rekonstruiert werden kann.

2.3 SfM mit einer bewegten Monokamera

Bisher wurde erläutert, wie mithilfe der Epipolargeometrie, der Projektionsmatrizen und der Triangulation die 3D-Koordinaten von Punkten aus zwei unterschiedlichen Kameraperspektiven berechnet werden können. In klassischen stereoskopischen Systemen werden dazu zwei synchronisierte Kameras verwendet, die eine Szene aus leicht versetzten Positionen aufnehmen.

In diesem Projekt wird jedoch eine einzelne Kamera verwendet, wodurch keine direkte Tiefeninformation wie bei einem Stereo-Kamerasystem zur Verfügung steht. Stattdessen wird die Bewegung der Kamera über die Zeit genutzt, um die räumlichen Informationen zu rekonstruieren. Hierbei werden die Kamerapositionen zu zwei aufeinanderfolgenden Zeitpunkten t und t+1 betrachtet.

Durch den Vergleich der Bilder, die die Kamera an diesen Zeitpunkten aufnimmt, lassen sich korrespondierende Bildpunkte identifizieren. Da sich die Kamera zwischen den beiden Zeitpunkten bewegt, ändert sich auch der Blickwinkel, wodurch die Methode analog zur stereoskopischen Bildverarbeitung genutzt werden kann. Die Berechnung der 3D-Punkte erfolgt somit nicht durch zwei fest montierte Kameras, sondern durch die Bewegung einer einzelnen Kamera über die Zeit.

(Abbildung 2-15) veranschaulicht diesen Ansatz, indem sie die Kamerapositionen zu den Zeitpunkten t und t+1 sowie die zugehörigen Bildprojektionen zeigt. Der dargestellte 3D-Punkt wird in beiden Ansichten als 2D-Projektion auf die Bildebene der Kamera abgebildet. Durch die relative Bewegung der Kamera zwischen den beiden Aufnahmen entstehen zwei leicht unterschiedliche Perspektiven auf denselben Punkt, wodurch die Berechnung der Tiefeninformation möglich wird.

Dieser Ansatz ermöglicht die Anwendung des SfM-Algorithmus mit nur einer Kamera, indem die räumliche Bewegung der Kamera selbst als Basis für die Rekonstruktion dient. Durch die kontinuierliche Aufnahme und Verarbeitung der Bilder entsteht ein rekonstruierbares 3D-Modell der Umgebung, vergleichbar mit den Ergebnissen eines Stereo-Kamerasystems.

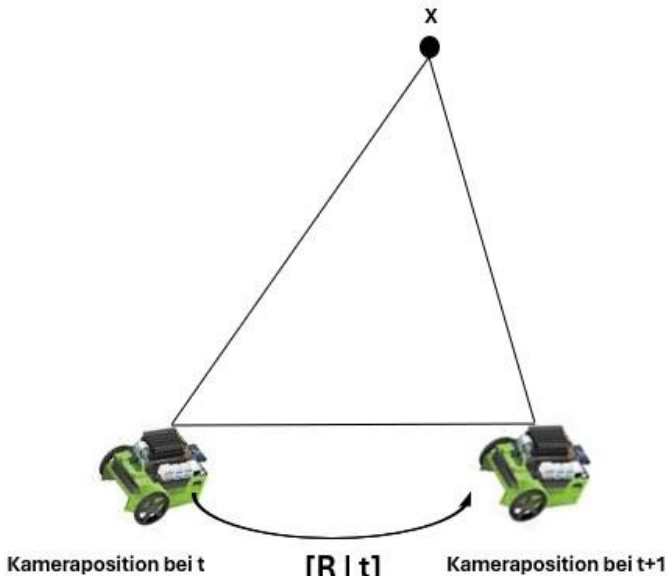

Abbildung 2-15: Kamerabewegung zur 3D-Rekonstruktion

2.4 Hinderniserkennung

Bisher wurde dargestellt, wie mithilfe der SfM Methode eine dreidimensionale Rekonstruktion der Umgebung mit einer bewegten Monokamera erstellt werden kann. Aufbauend auf diesen Erkenntnissen wird nun untersucht, wie diese Informationen zur Hinderniserkennung genutzt werden können. Eine präzise Identifikation von Hindernissen ist essenziell für die Navigation autonomer Systeme. Daher werden in diesem Abschnitt Verfahren zur Segmentierung der Punktwolke sowie zur Reprojektion der erkannten Strukturen vorgestellt, um eine zuverlässige Erkennung und Lokalisierung von Hindernissen zu ermöglichen.

2.4.1 Clusterbasierte Hinderniserkennung

Nach der Berechnung der dreidimensionalen Punkte durch Methoden wie die Triangulation entsteht eine Punktwolke, die die genauen Koordinaten jedes Punktes im dreidimensionalen Raum enthält. Diese Punktwolke bildet die Grundlage für die Rekonstruktion eines 3D-Modells und ermöglicht eine präzise Darstellung der geometrischen Strukturen sowie der Oberflächen von Objekten oder Szenen. Allerdings enthält die Punktwolke häufig verrauschte oder ungenaue Punkte, die durch verschiedene Filter- und Optimierungsverfahren bereinigt werden müssen [Chizhova, 2019].

Ein effektiver Ansatz zur Strukturierung dieser Punktwolke ist die Clusterbildung, bei der zusammenhängende Punktgruppen identifiziert und isolierte Punkte als Rauschen entfernt

werden. Besonders dichtebasierte Methoden wie DBSCAN[12] haben sich dabei als effizient erwiesen, da sie es ermöglichen, zusammenhängende Hindernisstrukturen zu identifizieren, ohne dass die Anzahl der Cluster im Voraus festgelegt werden muss.

Sobald die Punktwolke segmentiert wurde, kann sie für weiterführende Analysen genutzt werden. Dazu gehört unter anderem die Hinderniserkennung, die Berechnung von Entfernungen und die Erstellung dichter 3D-Modelle. Durch diese Methoden lassen sich Hindernisse im dreidimensionalen Raum zuverlässig identifizieren und markieren [Chunxiao Wang, 2019].

2.4.2 Reprojektion der Punktwolke

Es wurde bereits beschrieben, wie durch Clusterbildung zusammenhängende Hindernisstrukturen in der Punktwolke identifiziert werden. Damit diese Informationen effektiv zur Navigation genutzt werden können, muss die Entfernung der erkannten Cluster zur Kamera sowie zu anderen Objekten bestimmt werden.

Dazu werden die geclusterten Punkte aus dem dreidimensionalen Raum erneut auf die Bildebene projiziert (Reprojektion), um ihre genaue Position im ursprünglichen Bild zu analysieren. Dieser Schritt ermöglicht eine direkte Verbindung zwischen der berechneten 3D-Struktur und den originalen Bilddaten.

Durch die Reprojektion werden die 3D-Punkte aus der Triangulation wieder in das zweidimensionale Bild transformiert. Dies erlaubt eine präzisere Analyse der Clusterpositionen und erleichtert den Vergleich mit den Bildinformationen, wodurch eine genauere Abschätzung der Entfernung von Hindernissen und Objekten in der Szene erreicht wird. Da die 2D-Koordinaten der projizierten Punkte von den Kameraeigenschaften abhängen, erfolgt die Berechnung der Distanz zur Kamera anhand der Kalibrierungsmatrix K und projektiver Modelle. Dadurch können Hindernisse nicht nur erkannt, sondern auch ihre Abstände präzise bestimmt werden.

Die beschriebenen theoretischen Konzepte bilden die Grundlage für die praktische Umsetzung, die im folgenden Kapitel detailliert erläutert wird.

[12] DBSCAN ist ein dichtebasierter Clustering-Algorithmus, der nahe beieinander liegende Punkte gruppiert und Outliers anhand ihrer geringen Dichte als Rauschen markiert. Cluster werden als dicht besetzte Regionen erkannt, die durch Bereiche mit niedrigerer Dichte voneinander getrennt sind

3 Konzeption und Spezifikation

In diesem Kapitel wird das Konzept der Hinderniserkennung basierend auf Structure from Motion (SfM) entwickelt. Dabei werden die technischen Anforderungen definiert, die verwendete Hardware vorgestellt und die Strategie zur Hinderniserkennung beschrieben. Ziel ist es, ein robustes System zu entwerfen, das Hindernisse zuverlässig erkennt und ihre Position bestimmt, um eine sichere Navigation zu ermöglichen.

3.1 Zielsetzung

Diese Arbeit verfolgt das Ziel, ein Hinderniserkennungssystem zu entwickeln, das auf der SfM-Technologie basiert und speziell für den JetBot optimiert ist. Durch die Nutzung einer Fischaugenkamera soll ein erweitertes Sichtfeld geschaffen werden, um Hindernisse effizienter zu erkennen. Das System soll nicht nur Hindernisse identifizieren, sondern auch deren genaue Position und Entfernung zur Kamera bestimmen.

3.2 JetBot-Hardware.

Der JetBot ist ein Open-Source-Roboter, der auf dem NVIDIA Jetson Nano basiert. Für dieses Projekt wurde ein bereits programmierter und einsatzbereiter JetBot verwendet. Lediglich die technischen Spezifikationen der Kamera mussten angepasst und berücksichtigt werden.

Am JetBot ist eine IMX219-160-Kamera installiert, die mit dem Jetson Nano vollständig kompatibel ist. Sie bietet eine beeindruckende Auflösung von 8 Megapixeln (3280 x 2464 Pixel) und basiert auf dem IMX219-Sensor. Die Kamera verfügt über einen diagonalen Sichtwinkel (FOV) von 160 Grad, eine Blendenöffnung von 2,35 und eine Brennweite von 3,15 mm. Mit einer CMOS-Größe von 1/4 Zoll und einer geringen Verzerrung von unter 14,3 % liefert sie präzise und klare Aufnahmen. Das Objektiv hat außerdem kompakte Maße von 6,5 mm x 6,5 mm Objektivgröße: 6,5 mm x 6,5 mm. Durch diese Eigenschaften eignet sich die Kamera optimal für den Einsatz im JetBot [JetBot, 2025]

3.3 Konzeption der Hinderniserkennungsstrategie

3.3.1 Überblick über die Strategie

Diese (Abbildung 3-16) zeigt die Hauptschritte des entwickelten Hinderniserkennungssystems. Der Prozess beginnt mit der Aufnahme eines Videos durch die Fischaugenkamera,

gefolgt von der Kamerakalibrierung. Anschließend erfolgt die SfM-basierte 3D-Rekonstruktion, um eine Punktwolke der Umgebung zu erzeugen. Diese wird weiter verarbeitet, um Hindernisse mittels Cluster-Algorithmen zu segmentieren und ihre Position zu bestimmen. Schließlich werden die erkannten Hindernisse auf die Bildebene zurückprojiziert, um ihre Entfernung zur Kamera zu berechnen.

Zur Umsetzung dieser Strategie wird CMake als Build-System eingesetzt. Dies ermöglicht eine flexible und plattformunabhängige Kompilierung des Codes sowie eine effiziente Verwaltung der Abhängigkeiten. Die technischen Details zur Implementierung und den einzelnen Schritten der Code-Ausführung werden in Kapitel 4 beschrieben.

3.3.2 Herausforderungen und Lösungen

Die folgende Diagrammdarstellung (Abbildung 3-16) veranschaulicht die Herausforderungen, Lösungsansätze und die einzelnen Schritte dieses Projekts. Dabei werden die verwendeten Algorithmen, Bibliotheken und Filter sowie die jeweiligen Ergebnisse jeder Phase strukturiert dargestellt. Diese visuelle Übersicht bietet eine klare Darstellung des Hinderniserkennungsprozesses vor der Kamera mithilfe des SfM-Algorithmus.

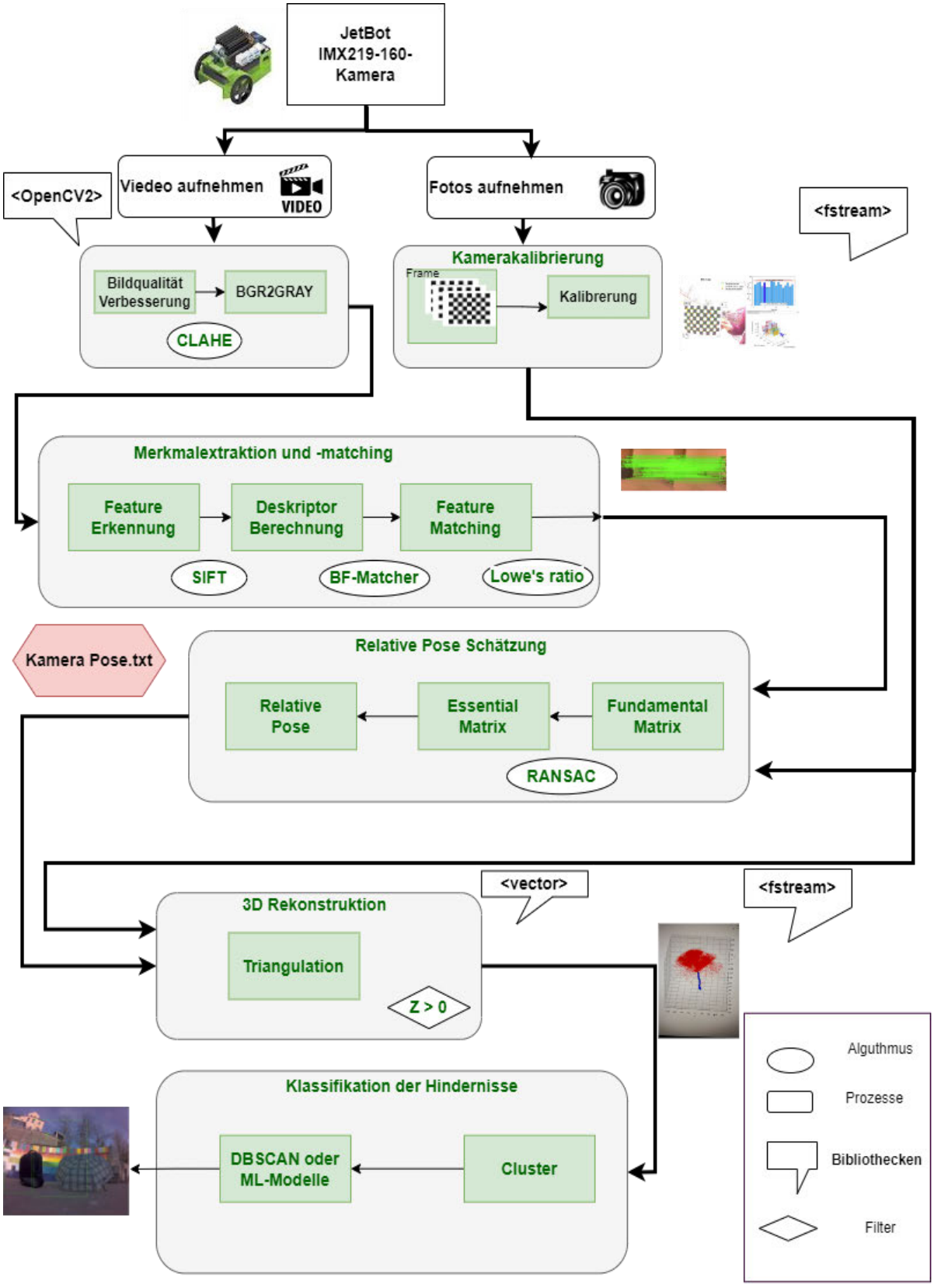

Abbildung 3-16: Übersicht der Hinderniserkennungsstrategie

In Tabelle 1 werden die in diesem Projekt verwendeten Algorithmen sowie die zugehörigen OpenCV-Module und deren Anwendungen dargestellt.

Tabelle 1: Übersicht der Algorithmen, Module und Anwendungen

Bibliothek	Modul	Algorithmus	Beschreibung & Anwendung
OpenCV	xfeatures2d	SIFT	Findet markante Punkte im Bild und beschreibt sie so, dass sie unter verschiedenen Größen und Drehungen erkannt werden können.
	features2d	BFMatcher	Vergleicht Merkmale aus zwei Bildern und sucht nach den besten Übereinstimmungen.
		KNN Match	Sucht für jedes Merkmal im ersten Bild die zwei ähnlichsten Merkmale im zweiten Bild und wählt die beste Übereinstimmung.
	core	RANSAC	Identifiziert Muster in Daten und filtert fehlerhafte Punkte heraus, um robustere Modelle zu erstellen.
	imgproc	CLAHE	Verbessert den Kontrast von Bildern, indem es lokale Helligkeitsunterschiede verstärkt.
	Opencv	findFundamentalMat	Berechnet die Beziehung zwischen zwei Kameraperspektiven anhand von Bildpunkten.
		findEssentialMat	Ermittelt, wie sich die Kamera zwischen zwei Aufnahmen bewegt hat.
		recoverPose	Bestimmt aus einer Matrix die Position und Ausrichtung der Kamera.
		triangulatePoints	Berechnet aus zwei Kameraperspektiven die dreidimensionalen Positionen von Bildpunkten.
	ml	DBSCAN	Gruppiert nahe beieinanderliegende Punkte, um Cluster zu bilden, die z. B. Hindernisse darstellen

Nachdem die in diesem Projekt verwendeten Algorithmen identifiziert wurden, sind in Tabelle 2 die für das Erreichen des Ziels programmierten Funktionen in der Reihenfolge ihrer Aufrufe im Code dargestellt.

Tabelle 2: Sequenzielle Aufrufstruktur und Funktionsbeschreibung

	Funktion	Beschreibung
1	main	Startet die SfM-Verarbeitung mit der angegebenen Videodatei und speichert die Ergebnisse.
2	structureFromMoti	Implementiert die gesamte SfM-Pipeline: Feature-Extraktion, Pose-Schätzung, Triangulation und Hinderniserkennung.
3	triangulatePoints	Berechnet 3D-Koordinaten von Bildpunkten aus zwei verschiedenen Kameraperspektiven.
4	remove_ground_plane	Entfernt den Boden aus der Punktwolke mit einer RANSAC-basierten Ebenen Erkennung.
5	dbscan_clustering	Gruppiert 3D-Punkte mithilfe des DBSCAN-Algorithmus zur Hinderniserkennung.
6	visualize_obstacle_clusters	Zeichnet erkannte Hindernis-Cluster als Bounding-Boxen auf das Bild.
7	reproject_points	Projiziert 3D-Punkte zurück auf das 2D-Bild mit einer Kamera-Projektionsmatrix.

4 Implementierung

In diesem Kapitel wird die Implementierung des SfM-Algorithmus zur Hinderniserkennung detailliert beschrieben. Die Umsetzung basiert auf den theoretischen Konzepten aus Kapitel 2 sowie den entwickelten Strategien aus Kapitel 3.

Zunächst wird die Entwicklungsumgebung eingerichtet bevor die einzelnen Schritte der Bildverarbeitung durchgeführt werden. Dazu gehören die Merkmalsextraktion, die Berechnung der relativen Kameraposition, die 3D-Rekonstruktion und schließlich die Erkennung von Hindernissen.

4.1 Einrichtung der Entwicklungsumgebung

Für die Implementierung des Algorithmus wurde die Programmiersprache C++ in Kombination mit der OpenCV-Bibliothek genutzt. Damit das System korrekt ausgeführt werden kann, müssen vor der eigentlichen Bildverarbeitung und Kamerakalibrierung die Kompilierung und der Ablauf des Codes erläutert werden.

4.1.1 Projektstruktur und wichtige Dateien

Das Projekt besteht aus mehreren zentralen Dateien, die unterschiedliche Aufgaben in der Implementierung übernehmen:

- **main.cpp** – Enthält den vollständigen Code für die SfM-Pipeline, einschließlich Feature-Erkennung, Bewegungsschätzung, Triangulation und Clustering der Punktwolke.
- **SfM.avi** – Eine Videodatei, die als Eingangsdatenquelle dient und zur Extraktion der Bewegungsinformationen genutzt wird.
- **CMakeLists.txt** – Die CMake-Konfigurationsdatei, die für die Kompilierung und das Verlinken der erforderlichen Bibliotheken benötigt wird.

4.1.2 Verwendung von CMake als Build-System

Für die Kompilierung des Codes wird CMake verwendet, ein plattformunabhängiges Build-System, das eine effiziente Verwaltung von Abhängigkeiten ermöglicht. Die Datei CMakeLists.txt definiert die grundlegenden Konfigurationen des Projekts. (Abbildung 4-17) zeigt den Inhalt der CMakeLists.txt-Datei und die darin enthaltenen Konfigurationsanweisungen.

```
cmake_minimum_required(VERSION 3.1)
project(SfM_Project)

# OpenCV-Bibliothek hinzufeugen
find_package(OpenCV REQUIRED)

# Include directories
include_directories(${OpenCV_INCLUDE_DIRS})


# Setze den Standard auf C++17
set(CMAKE_CXX_STANDARD 17)
set(CMAKE_CXX_STANDARD_REQUIRED ON)

#
add_executable(sfm main.cpp)


# linke mit OpenCV-Bibliotheken
target_link_libraries(sfm ${OpenCV_LIBS})

```

Abbildung 4-17: CMake.txt

Diese Konfiguration stellt sicher, dass der Code mit OpenCV kompiliert und als ausführbare Datei sfm generiert wird.

4.1.3 Kompilierung und Ausführung des Codes

Um das Programm auszuführen, müssen die folgenden Schritte in einem Terminal oder einer Shell eingegeben werden. Diese Schritte ermöglichen es, die C++-Dateien mit CMake zu kompilieren und das generierte ausführbare Programm zu starten:

1. **Öffne ein Terminal, erstelle einen Build-Ordner und Generieren der Makefiles:**
   `mkdir build && cd build`
   Dieser Schritt erstellt einen separaten Ordner für die kompilierten Dateien und wechselt in diesen Ordner.
   `cmake ..`
   Dieser Befehl sucht nach der Datei CMakeLists.txt und erstellt die erforderlichen Makefiles für die Kompilierung.
2. **Kompilieren des Codes:**
   `Make`
   Nach erfolgreicher Kompilierung wird eine ausführbare Datei mit dem Namen sfm erzeugt.
3. **Ausführen des Programms:**
   `./sfm`

Sobald das Programm gestartet wird, beginnt die Verarbeitung der Videodatei SfM.avi, und die berechneten Ergebnisse werden in den entsprechenden Dateien gespeichert.

Nach erfolgreicher Ausführung werden verschiedene Ausgabedateien generiert, die zur Analyse der SfM-Ergebnisse genutzt werden können.

4.1.4 Kamerakalibrierung und Vorbereitung der Bildverarbeitung

Die verwendete IMX219-160-Kamera ist mit einem Fischaugenobjektiv ausgestattet, wodurch eine starke Verzerrung der aufgenommenen Bilder entsteht. Um diese Verzerrung zu korrigieren und genauere Ergebnisse zu erhalten, muss die Kamera zuerst kalibriert werden. Wie bereits in Abschnitt 2.1.4 beschrieben, ist die Kalibrierung sehr wichtig für die Genauigkeit des gesamten Systems. Besonders die internen Kameraeinstellungen und die Verzerrungsmatrix spielen eine große Rolle bei der späteren Bildverarbeitung. Um möglichst präzise Kalibrierungswerte zu bekommen, wurde die Kalibrierung mit drei verschiedenen Methoden durchgeführt: Python, C++ und MATLAB. Da alle drei Kalibrierungsmethoden getestet wurden, wird hier die genaueste und effektivste Methode beschrieben, die Bestimmung der inneren Kameraparameter und der Verzerrung mit MATLAB.

Die Kamera nimmt Bilder eines Musters aus verschiedenen Winkeln und Entfernungen auf, sodass ausreichend Referenzpunkte in den Bildern sichtbar sind. In diesem Projekt wird ein A4-Schachbrettmuster mit Quadraten von 25 mm Größe und einer Anordnung von 10 x 7 Quadraten verwendet. Die Anzahl der aufgenommenen Fotos hängt von der verwendeten Software ab – in MATLAB werden 10 bis 20 Fotos empfohlen. Das Ziel besteht darin, bestimmte Punkte, wie die Ecken des Musters, in den Bildern zu identifizieren. Zur Erkennung dieser Schlüsselpunkte werden Bildverarbeitungsalgorithmen eingesetzt. (Abbildung 4-18) zeigt die in einem der für die Kalibrierung verwendeten Bilder identifizierten Punkte und reprojektierten Punkte, die Position jedes Fotos relativ zur Kamera sowie ein Diagramm des Reprojektion-Fehlers.

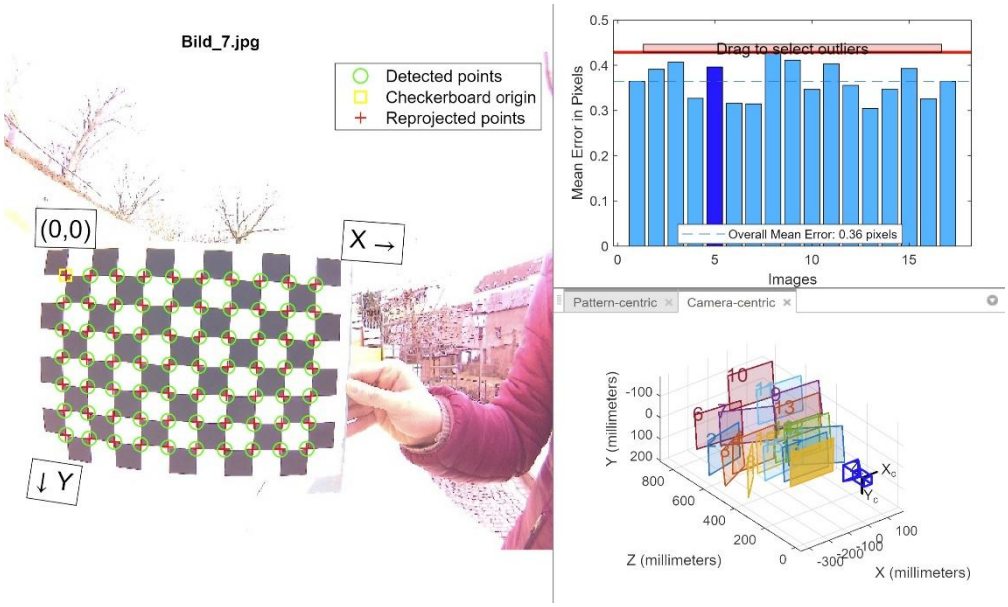

Abbildung 4-18: Kalibrierung mit Parameter

Sowie die Kamera erfolgreich kalibriert wurde, folgt der nächste Schritt: die Aufnahme eines Videos, das als Eingabedaten für die weitere Verarbeitung im Algorithmus verwendet wird. Diese Videodaten bilden die Grundlage für die Merkmalsextraktion, die Berechnung der Kamerabewegung und schließlich die Rekonstruktion der 3D-Punktwolke.

Damit die Kalibrierung möglichst genaue Werte liefert, sollte die Aufnahme unter denselben Bedingungen und am selben Ort gemacht werden, an dem später auch das Video aufgenommen wird. Das ist wichtig, weil der Kamerawinkel und die Beleuchtung die Bildqualität und die Berechnung der Kameraparameter beeinflussen können. Die höchste Auflösung der Kamera (**3280×2464 Pixel**) wurde für die Videoaufnahme eingestellt. Allerdings konnte das Video wegen der Hardware-Grenzen des Jetson Nano nicht mit 21 Bildern pro Sekunde aufgenommen werden. Um dieses Problem zu lösen, wurde die Bildrate auf 6 Bilder pro Sekunde reduziert.

Diese Änderung hatte keinen Einfluss auf die Erkennung von Schlüsselpunkten und das Feature-Matching, da diese Algorithmen jeweils nur auf einzelne Fram angewendet werden. Allerdings kann eine geringere Bildrate dazu führen, dass bestimmte Blickwinkel verloren gehen, was die 3D-Rekonstruktion und Hinderniserkennung beeinflussen könnte. Da die Algorithmen DBSCAN und die Clusterbildung für Hindernisse von der Qualität der 3D-Punkte abhängen, könnte sich die reduzierte Bildrate auch auf die Genauigkeit der Hinderniserkennung auswirken.

Um starke Unterschiede zwischen den aufeinanderfolgenden Bildern zu vermeiden und eine gute Qualität der 3D-Rekonstruktion zu erhalten, wurde die Geschwindigkeit des Roboters während der Videoaufnahme reduziert. Die Geschwindigkeit des Roboters wurde

auf das kleinste mögliche Niveau eingestellt, bei dem eine Bewegung noch möglich ist (robot.forward (speed = 0.2)).

4.2 Implementierung der SfM

Sobald die Entwicklungsumgebung eingerichtet und die Kamera erfolgreich kalibriert wurde, folgt nun die eigentliche Implementierung des SfM-Algorithmus.

4.2.1 Kamera Parameter

Der erste Implementierungsschritt besteht darin, die Kameraparameter und Entzerrungsparameter einzulesen, weil in der Bildverarbeitung die Helligkeit der Umgebung und die Bildqualität die wichtigsten und einflussreichsten Faktoren sind.

In Abbildung 4-19 werden die Kalibrierungsparameter für eine Fisheye-Kamera dargestellt. Bei dieser Fisheye-Kamera wird eine negative radiale Verzerrung (Barrel Distortion) beobachtet. Die radiale Verzerrung wurde ausführlich in Abschnitt 2.1.3 erläutert.

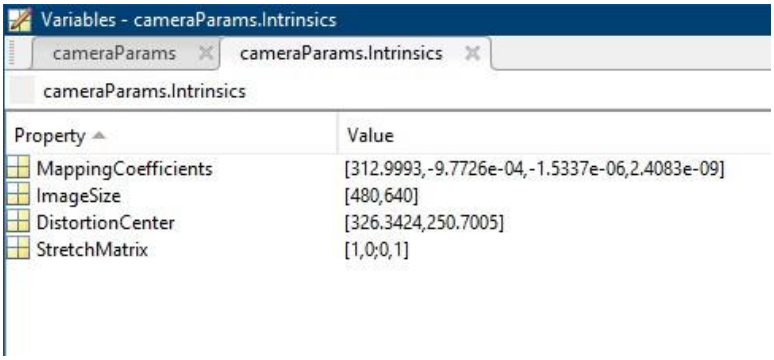

Abbildung 4-19: Kamera Parameter

In der Kalibrierung mit dem Fisheye-Modell wurden verschiedene Parameter bestimmt. Der erste Wert in MappingCoefficients gibt die Brennweite an, während die folgenden Werte die Verzeichnung (Distortion) beschreiben. ImageSize zeigt die Bildauflösung in Pixeln. DistortionCenter gibt die Koordinaten des Zentrums der Verzerrung im Bild an. Die StretchMatrix wird für die Skalierung der Bildkoordinaten verwendet.

Mithilfe der Formeln 2-3 und 2-4 in Abschnitt 2.1 und der Informationen zu den Kameraspezifikationen, die in Abschnitt 3.2 erläutert werden, kann die Kalibrierungsgenauigkeit gemessen werden. Bei einer Auflösung von **640 x 480** für die Kalibrierung lauten die gewünschten Werte für Brennweite und Bildmitte wie folgt:

$$f_x = \frac{3.15 \times 640}{3.6} = 560$$
$$f_y = \frac{3.15 \times 480}{2.7} = 560 \quad (4\text{-}19)$$

$$c_x = \frac{640}{2} = 320,\ c_y = \frac{480}{2} = 240 \quad (2\text{-}20)$$

Der theoretisch berechnete Wert für $\mathbf{f_x=f_y=560}$ Pixel unterscheidet sich deutlich von dem nach der Kalibrierung $\mathbf{f_x=f_y=312.9993}$ Pixel ermittelten Wert. Hingegen liegen die Werte für $\mathbf{C_x= 326.3424}$, $\mathbf{C_y= 250.7005}$ im akzeptablen Bereich. Daher kann dieser Unterschied nicht direkt als Kalibrierungsfehler betrachtet werden, sondern ist hauptsächlich auf die starke Verzerrung der Linse und die Fisheye-Eigenschaften zurückzuführen.

Um dies zu überprüfen, wird das aufgenommene Bild mit diesen Parametern entzerrt. Falls das korrigierte Bild ohne sichtbare Verzerrungen dargestellt wird, kann davon ausgegangen werden, dass die Kalibrierung korrekt war. Abbildung 4-20 zeigt die Korrektur der Bildverzerrung anhand der ermittelten Parameter.

Zusätzlich bietet der Reprojektion-Fehler eine Möglichkeit zur Bewertung der Kalibrierungsgenauigkeit. In diesem Test beträgt der Fehler etwa 0.45 Pixel, was als akzeptabel gilt.

Abbildung 4-20: Entzerrtes Bild nach der Korrektur

4.2.2 Videodatei einlesen

Nach der Eingabe der Kameraparameter wird als nächstes das Video geladen und für die weitere Verarbeitung vorbereitet. Dafür wird die OpenCV-Klasse VideoCapture genutzt, die

das Video Bild für Bild einliest. Falls die Datei nicht geöffnet werden kann, erscheint eine Fehlermeldung. Sobald das Video erfolgreich geladen wurde, steht es für weitere Verarbeitungsschritte, wie das Auslesen von Kameraparametern und die Merkmalsextraktion, zur Verfügung.

4.2.3 Extrahieren Frames aus dem Video

Bei der Bildverarbeitung werden alle Berechnungen auf einzelnen Frames durchgeführt. Deshalb muss das Video nach dem Laden in einzelne Frames zerlegt werden, damit die Verarbeitung Bild für Bild erfolgen kann.

4.2.4 In Graustufen konvertieren

Sobald das erste Bild geladen und überprüft wurde, dass es korrekt eingelesen wurde, wird der Kontrast verbessert. Dafür kommt der CLAHE[13]-Algorithmus zum Einsatz. Mit dieser Methode kann der Bildkontrast erhöht werden, sodass wichtige Merkmale in späteren Schritten besser erkannt werden.

Der CLAHE-Algorithmus wird zunächst auf das Farbbild angewendet, bevor es in Graustufen umgewandelt wird. Dadurch wird die Bildqualität verbessert, ohne dass unerwünschte Farbveränderungen auftreten. Wenn der Algorithmus direkt auf Graustufenbilder angewendet wird, können hingegen Farbabweichungen entstehen.

Um die Farben zu bewahren, wird das Bild zuerst in den Lab-Farbraum umgewandelt. Dabei wird nur die Helligkeitskomponente (**L**) bearbeitet, während die Farbinformationen unverändert bleiben. Diese Methode sorgt für eine bessere Detaildarstellung, ohne die Farben zu verändern. Besonders bei schwacher oder ungleichmäßiger Beleuchtung ist dieser Ansatz sehr hilfreich.

Im Lab-Farbraum gibt es drei Kanäle:

- L (labChannels[0]): Enthält die Helligkeitsinformation des Bildes.
- a (labChannels[1]): Speichert den Farbunterschied zwischen Grün und Rot.
- b (labChannels[2]): Enthält den Farbunterschied zwischen Blau und Gelb.

Während der Verarbeitung wird der CLAHE-Algorithmus nur auf den L-Kanal (labChannels[0]) angewendet, um die Helligkeit und den Kontrast zu verbessern. Die Farbinformationen bleiben dabei unverändert. Diese Methode hilft, die Bildqualität zu steigern und Details klarer sichtbar zu machen, ohne die natürlichen Farben zu verändern. Besonders bei

[13] Der CLAHE-Algorithmus (Contrast Limited Adaptive Histogram Equalization) stellt eine optimierte Form der adaptiven Histogramm-Equalisierung dar und dient zur Kontrastverbesserung in Bildern.

schwierigen Lichtverhältnissen kann sie dazu beitragen, Bilder für die weitere Verarbeitung besser nutzbar zu machen.

Farbbilder haben drei Kanäle (R, G, B), während Graustufenbilder nur einen Helligkeitskanal enthalten. Aus diesem Grund ist die Bildverarbeitung von Graustufenbildern schneller und erfordert weniger Rechenleistung. Außerdem können wichtige Bildmerkmale wie Kanten und Strukturen klarer erkannt werden, während störendes Rauschen reduziert wird.

4.2.5 Feature Detektion und Matching

Sowie der Frame in ein Graustufenbild umgewandelt wurde, erfolgt die Extraktion von Schlüsselpunkten und Bildmerkmalen. Dafür wird der SIFT-Algorithmus verwendet.Mit SIFT können Schlüsselpunkte unabhängig von Maßstabs- und Rotationsänderungen erkannt werden. Im Vergleich zu anderen Algorithmen zur Merkmalserkennung ist SIFT zudem robuster gegenüber Helligkeitsveränderungen. Da in diesem Projekt starke Helligkeitsschwankungen auftreten, wurde dieser Algorithmus als effiziente und geeignete Lösung gewählt. In Abbildung 4-21 werden die Schlüsselpunkte gezeigt, die in einem Frame mithilfe des SIFT-Algorithmus erkannt wurden.

Um die Rechenleistung zu optimieren, wurde die Anzahl der extrahierten Merkmale auf maximal 1500 Schlüsselpunkte begrenzt. Dadurch wird ein gutes Gleichgewicht zwischen Erkennungsgenauigkeit und Verarbeitungsgeschwindigkeit erreicht und eine optimale Bildanalyse ermöglicht.

Abbildung 4-21: Merkmalserkennung in einem Frame

Nachdem die Schlüsselpunkte erkannt und ihre Deskriptoren extrahiert wurden, müssen die Merkmale des vorherigen und des aktuellen Frames miteinander verglichen werden, um gemeinsame Punkte zwischen den beiden Frames zu identifizieren.

Im Anschluss an die Erkennung der Schlüsselpunkte und die Extraktion der Deskriptoren werden die Merkmale des vorherigen und des aktuellen Frames miteinander verglichen, um gemeinsame Punkte zwischen den beiden Frames zu identifizieren.

Dafür wird der BFMatcher-Algorithmus verwendet. Bei diesem Verfahren wird jeder Deskriptor aus dem ersten Frame mit allen Deskriptoren des zweiten Frames verglichen. Anschließend wird das nächstgelegene Merkmal bestimmt, wobei die NORM_L2-Methode genutzt wird. Dabei wird der euklidische Abstand (L2-Norm) berechnet, um die Ähnlichkeit der SIFT-Deskriptoren zu bestimmen.

Zusätzlich wird durch eine weitere Eingabe gesteuert, ob die Merkmalszuordnung einseitig oder beidseitig erfolgen soll. In dieser Implementierung wurde (false) gewählt, was bedeutet, dass eine einseitige Zuordnung verwendet wird.

Um die Genauigkeit der Merkmalsanpassung zu verbessern, wird zusätzlich der KNN Match[14] Algorithmus eingesetzt. Dabei werden die zwei nächstgelegenen Merkmale für den Abgleich ausgewählt, wodurch eine präzisere Übereinstimmung zwischen den Frames ermöglicht wird. In diesem Programm werden sowohl BFMatcher als auch KNN Matching kombiniert, um die bestmögliche Merkmalsanpassung zu erreichen.

Nach der ersten Zuordnung wird der Lowe's Ratio Test[15] angewendet, um ungeeignete Zuordnungen zu entfernen. Dabei wird der Abstand zwischen der besten und der zweitbesten Übereinstimmung verglichen. Wenn der Abstand der besten Übereinstimmung weniger als 75 % des zweiten Abstands beträgt, wird die Übereinstimmung als zuverlässig betrachtet.

Mit **(bf.knnMatch())** werden zunächst die zwei besten Übereinstimmungen für jede Merkmalszuordnung gefunden. Anschließend wird der Lowe's Ratio Test durchgeführt, um nur die zuverlässigsten Übereinstimmungen beizubehalten. danach werden die erkannten Merkmale angezeigt, wenn vorherige Daten vorhanden sind und mindestens eine gültige

[14] KNN Match: K-Nearest Neighbors Matching

[15] Lowe's Ratio Test: Der Lowe's Ratio Test ist eine Filtermethode, die zur Verbesserung der Genauigkeit bei der Merkmalserkennung eingesetzt wird. Dabei wird für jedes Merkmal aus dem ersten Bild die zwei nächstgelegenen Übereinstimmungen aus dem zweiten Bild ermittelt. Anschließend wird der Abstand dieser beiden Übereinstimmungen verglichen. Wenn die beste Übereinstimmung (d_1) deutlich besser ist als die zweitbeste (d_2) genauer gesagt, wenn das Verhältnis $d_1/d_2 < 0.75$ beträgt, wird die Übereinstimmung als zuverlässig betrachtet. Ist dieses Verhältnis jedoch größer, bedeutet das, dass die gefundene Übereinstimmung möglicherweise nicht korrekt ist oder durch Rauschen beeinflusst wurde. In diesem Fall wird sie verworfen. Durch diese Methode bleiben nur die stärksten und verlässlichsten Merkmale erhalten, während zufällige oder falsche Übereinstimmungen aussortiert werden. Dies trägt maßgeblich zur Verbesserung der Leistung von Computer-Vision-Algorithmen bei.

Übereinstimmung festgestellt wurde. Abbildung 4-22 stellt die ermittelten Feature-Matches zwischen zwei aufeinanderfolgenden Frames des Projekts dar.

Abbildung 4-22: Visualisierung der Übereinstimmungen zwischen zwei Bildern

4.2.6 Merkmalsbasierte Bewegungsberechnung

Direkt nach der Identifikation der gemeinsamen Merkmale zwischen zwei aufeinanderfolgenden Frames erfolgt die Berechnung der relativen Kamerabewegung. Mit der Essential Matrix können die Rotation (**R**) und die Translation (**t**) der Kamera berechnet werden. Diese Methode funktioniert jedoch nur genau, wenn die Kamera zuvor kalibriert wurde. Schließlich werden die berechneten Werte im globalen Koordinatensystem (Global Frame) berücksichtigt, und die Kameraposition wird im Laufe der Zeit aktualisiert.

Die Essential Matrix ermöglicht es, die Bewegung der Kamera ohne Tiefeninformationen zu bestimmen, da sie lediglich die Übereinstimmungen zwischen zwei Frames nutzt. Für die Berechnung sind mindestens 8 korrekte Übereinstimmungen (Matches) erforderlich.

Da in diesem Projekt nur eine einzelne Kamera (Monocular Camera) verwendet wird, kann die absolute Tiefe nicht direkt berechnet werden. Die Essential Matrix gibt lediglich die relative Bewegung der Kamera an, während der absolute Maßstab (Scale) unbekannt bleibt.

Um Fehler zu reduzieren und falsche Übereinstimmungen (Outliers) zu entfernen, wird der RANSAC[16]-Algorithmus verwendet. Dieser Algorithmus erkennt fehlerhafte oder

[16] Der RANSAC-Algorithmus (Random Sample Consensus) ist ein iteratives Verfahren zur Bestimmung eines mathematischen Modells aus einem Datensatz, der auch Ausreißer enthalten kann. Dabei identifiziert der Algorithmus zunächst die Ausreißer und nutzt anschließend ausschließlich die verbleibenden Datenpunkte, um das gewünschte Modell zu schätzen.

verrauschte Matches und filtert sie heraus, sodass nur zuverlässige und präzise Merkmale erhalten bleiben.

Schließlich werden mit der recoverPose()-Funktion in OpenCV die Werte für die Rotation ($\mathbf{R}_{est}$) und die Translation ($\mathbf{t}_{est}$) der Kamera aus der Essential Matrix berechnet und als Teil der Bewegungsbahn der Kamera weiterverwendet.

Sobald die relative Rotation ($\mathbf{R}_{est}$) und die Translation ($\mathbf{t}_{est}$) zwischen zwei aufeinanderfolgenden Bildern berechnet wurden, wird die globale Pose der Kamera aktualisiert. Dabei müssen diese Veränderungen mit der vorherigen Kameraposition kombiniert werden, um ihre Lage und Orientierung im dreidimensionalen Raum zu bestimmen.

Da die Bewegung der Kamera relativ ist, muss die neue Rotation auf die vorherige angewendet werden. Dadurch wird sichergestellt, dass sich die Gesamtausrichtung der Kamera in Bezug auf den Ausgangspunkt kontinuierlich anpasst.

$$\boldsymbol{R_{glob} = R_{est} * R_{p_{rev}}} \tag{4-21}$$

Die Aktualisierung der Position erfolgt auf ähnliche Weise, jedoch mit einem entscheidenden Unterschied: Da die berechneten Translationswerte relativ sind, müssen sie in das Koordinatensystem der vorherigen Rotation transformiert werden. Würde t_{est} einfach direkt zur vorherigen Position addiert, würde die Bewegung ohne Berücksichtigung der vorherigen Rotationen erfasst, was zu einem falschen Bewegungspfad führen würde.

$$\boldsymbol{t_{glob} = R_{Prev^*} t_{est} + t_{Prev}} \tag{4-22}$$

Betrachtet man nur ein einzelnes Bild, ist zwar bekannt, wie weit sich die Kamera bewegt hat, jedoch nicht, welchen gesamten Pfad sie zurückgelegt hat. Mit dieser Methode wird jede neue Bewegung kontinuierlich mit den vorherigen kombiniert, sodass die tatsächliche Kameratrajektorie im dreidimensionalen Raum korrekt rekonstruiert wird. Abbildung 4-23 zeigt die zweidimensionale Trajektorie der Kamerabewegung basierend auf den berechneten Positionen.

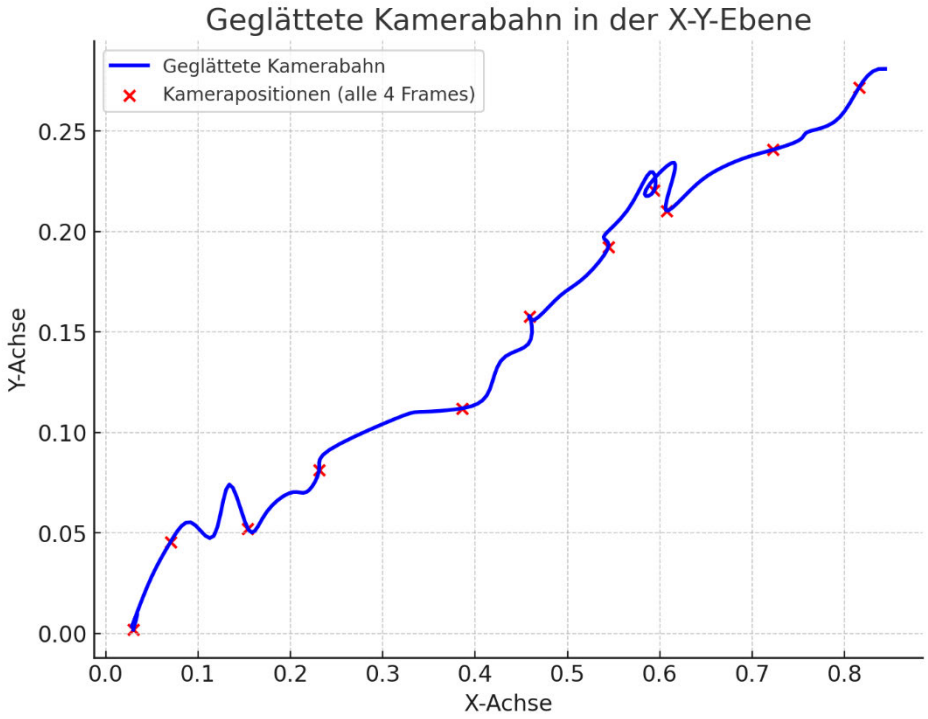

Abbildung 4-23: 2D-Trajektorie der Kamerabewegung

4.2.7 Triangulation

Im letzten Schritt der SfM-Rekonstruktion werden die dreidimensionalen Punkte unter Verwendung der Kamerabewegung, dargestellt durch R_{glob} ,t_{glob}, sowie der übereinstimmenden Merkmale in aufeinanderfolgenden Frames rekonstruiert. Zudem spielen die in Abschnitt 2.2.3 beschriebenen Projektionsmatrizen $\mathbf{P_1}$,$\mathbf{P_2}$ eine zentrale Rolle in diesem Prozess.

Zur Berechnung der 3D-Koordinaten eines Punktes werden zunächst dessen zweidimensionale Positionen in zwei aufeinanderfolgenden Frames ermittelt Pts1 im ersten und Pts2 im zweiten Frame. Mithilfe der Triangulation werden die entsprechenden Bildpunkte als Strahlen im dreidimensionalen Raum modelliert, deren Schnittpunkt die geschätzte Position des 3D-Punktes bestimmt. Dieser Vorgang wird für alle als Inlier klassifizierten Merkmalspaare durchgeführt.

Am Ende dieses Prozesses entsteht für jedes Frame eine dichte Punktwolke (Point Cloud), die die rekonstruierte 3D-Struktur repräsentiert. Diese Punktwolken werden für jedes einzelne Frame im PLY-Format gespeichert, sodass sie in einem späteren Schritt zur Bestimmung der Entfernung von Hindernissen zur Kamera verwendet werden können. Abbildung 4-24zeigt die erkannten 3D-Punkte im Frame 34. Dabei ist deutlich erkennbar, dass die Punktdichte in einem Bereich besonders hoch ist.

Abbildung 4-24: 3D Punkte Frame 34

4.3 Hinderniserkennung

4.3.1 3D-Rekonstruktion und Visualisierung

Um Hindernisse in der Punktwolke zu identifizieren, werden die extrahierten dreidimensionalen Punkte in jedem Frame mit dem DBSCAN-Algorithmus gruppiert. Zunächst wird sichergestellt, dass die Liste der Punkte nicht leer ist, und jedem Punkt wird standardmäßig das Label (**Null)** zugewiesen. Punkte, die als Rauschen erkannt werden, erhalten das Label **(–1)**, während den Punkten, die einem Cluster zugeordnet werden, die entsprechende Cluster-Nummer zugeteilt wird. Anschließend wird der euklidische Abstand zwischen einem Punkt und den übrigen Punkten berechnet, und es werden die Nachbarpunkte innerhalb eines definierten ε-Radius ermittelt. Liegt die Anzahl der Nachbarpunkte unter dem Schwellenwert **minPts**, so wird der Punkt als Rauschen klassifiziert; andernfalls wird ein neues Cluster begonnen und der Punkt als Ausgangspunkt dieses Clusters betrachtet. Dieses Verfahren dient dazu, dichte Bereiche in der Punktwolke zu identifizieren, die auf das Vorhandensein eines Objekts hindeuten.

Im weiteren Verlauf werden die gruppierten Punkte in einer Datenstruktur als Liste von Listen **(Point3f)** gespeichert, und die Anzahl der erkannten Cluster wird im Terminal ausgegeben. Auf diese Weise werden Hindernisse in der Szene automatisch anhand der Punktdichte erkannt und von Rauschpunkten getrennt. Die so gewonnenen Ergebnisse werden in einem nächsten Schritt für weitere Analysen sowie zur Bestimmung der Entfernung der Hindernisse zur Kamera herangezogen.

4.3.2 Reprojektion

Nachdem die Hindernisse in der Punktwolke erkannt und gruppiert wurden, erfolgt die visuelle Darstellung dieser Cluster im ursprünglichen Kamerabild. Es werden nur die Cluster berücksichtigt, deren Größe den festgelegten Mindestwert **(minClusterSize)** überschreitet, um die Anzeige kleiner und irrelevanter Gruppen zu vermeiden.

Für jedes dieser gültigen Cluster werden die zugehörigen 3D-Punkte mithilfe der Kameraparameter in 2D-Bildkoordinaten rückprojiziert. Dieser Prozess basiert auf der globalen Rotationsmatrix $\mathbf{R}_{glob}$, dem Translationsvektor $\mathbf{t}_{glob}$ und der Kalibrierungsmatrix **K**. Nach der erfolgreichen Reprojektion wird eine Begrenzungsbox (Bounding Box) für jedes Cluster berechnet, die den äußeren Bereich der projizierten Punkte im Bild definiert. Schließlich wird diese Box in grüner Farbe auf das Bild gezeichnet, sodass die Hindernisse visuell dargestellt werden. Abbildung 4-25 zeigt das Endergebnis der Verarbeitung eines Frames.

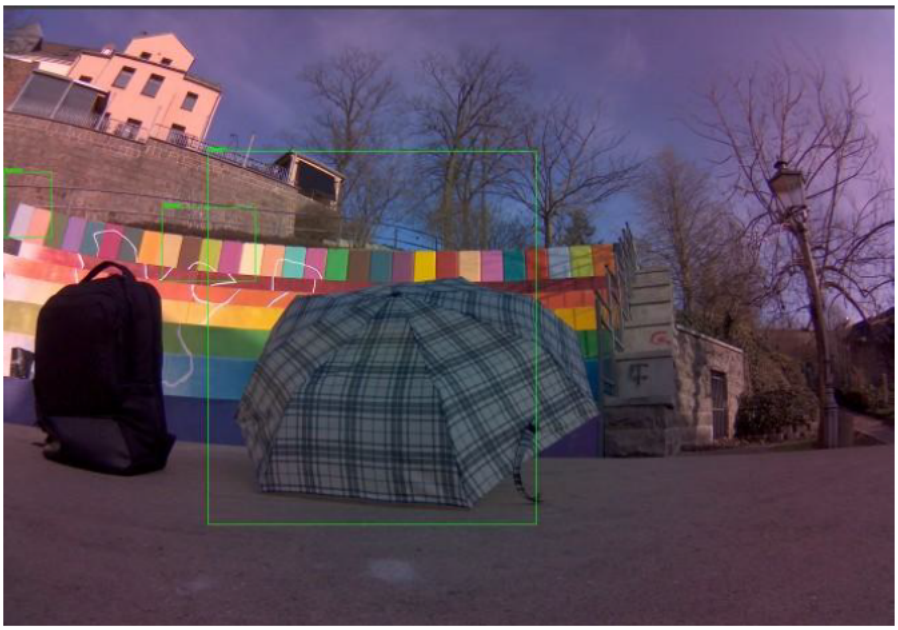

Abbildung 4-25: Visualisierung des endgültigen Ergebnisses für einen Frame

Durch diesen Prozess wurde das Hauptziel des Projekts, nämlich die Hinderniserkennung mithilfe von SfM, erfolgreich erreicht. Die Hindernisse in der Szene wurden nun präzise identifiziert und in der Ausgabe mit grünen Rahmen markiert. Darüber hinaus ermöglichen die ermittelten 3D-Koordinaten der Punkte die Bereitstellung relevanter Informationen für den Roboter, sodass er die Hindernisse erkennen und seine Bewegungsbahn entsprechend anpassen kann, um Kollisionen zu vermeiden.

5 Experiment, Validierung und Anwendbarkeit

In diesem Abschnitt werden der Testprozess sowie die Bewertung der SfM-Methode zur Hinderniserkennung mit einer Fischaugenkamera beschrieben. Dabei werden verschiedene Herausforderungen, die während der Implementierung und praktischen Tests auftraten, analysiert und Lösungsansätze vorgestellt, um die Systemleistung zu verbessern.

5.1 Experiment und Validierung

Während der Durchführung dieses Projekts traten verschiedene Herausforderungen auf – einige konnten erfolgreich gelöst werden und führten zu besseren Ergebnissen, während andere weiterhin bestehen. In diesem Abschnitt werden die erzielten Ergebnisse und die aufgetretenen Herausforderungen Schritt für Schritt analysiert.

5.1.1 Überprüfung der Kamerakalibrierung

Obwohl die theoretische Analyse der Kameraparameter in Kapitel 2 und 4 bereits durchgeführt wurde, um die Brennweite und die Bildmitte zu bestimmen, war eine zusätzliche experimentelle Überprüfung notwendig. Ziel war es, die Genauigkeit der ermittelten Kalibrierungsparameter zu validieren und sicherzustellen, dass die berechneten Werte auch in der praktischen Anwendung zuverlässig bleiben.

Dafür wurde ein kreisförmiges Objekt mit einer bekannten Entfernung zur Kamera an verschiedenen Positionen im Raum platziert. Anschließend wurden Bilder aus unterschiedlichen Blickwinkeln aufgenommen. Diese Bilder wurden mithilfe der zuvor berechneten Kalibrierungsparameter verarbeitet, um das Zentrum des Kreises exakt zu bestimmen.

Abbildung 5-26 zeigt das Ergebnis dieser Berechnung. Der Vergleich zwischen der tatsächlich bekannten und der berechneten Position des Kreismittelpunkts bestätigte, dass die Kalibrierung unter idealen Bedingungen korrekt durchgeführt wurde und die ermittelten Parameter zuverlässig waren.

Doch während dieser experimentellen Überprüfung wurde deutlich, dass sich in der realen Anwendung neue Herausforderungen ergaben, die die Genauigkeit der Kameraparameter beeinflussten. Diese Unterschiede und ihre Auswirkungen auf das Gesamtsystem werden im Folgenden genauer analysiert.

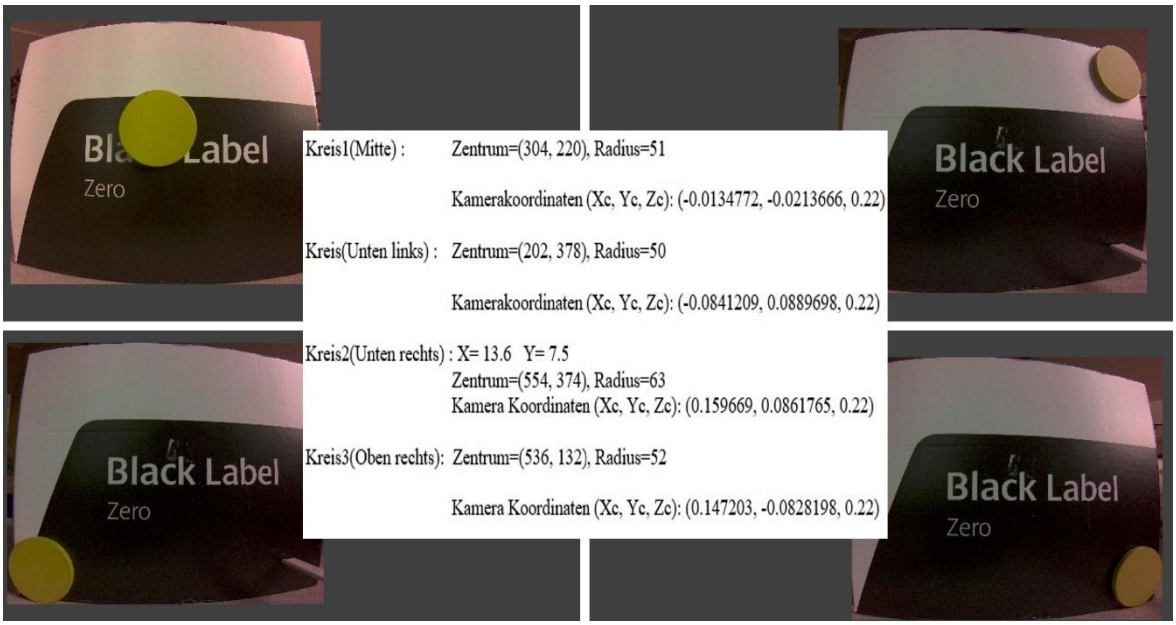

Abbildung 5-26: Kalibrierungstest: Erkennung des Kreismittelpunkts

Einfluss der Bildqualität auf die Kalibrierungsgenauigkeit:

Die Unterschiede in der Bildqualität zwischen der Kalibrierungsphase und der tatsächlichen Anwendung können die Genauigkeit der Kalibrierungsparameter negativ beeinflussen. Diese Abweichungen entstehen hauptsächlich durch die begrenzte Rechenleistung des Jetson Nano sowie durch veränderte Aufnahmebedingungen während der Bewegung des Roboters. Die wichtigsten Faktoren, die zu dieser Ungenauigkeit führen, sind:

a) Erhöhtes Bildrauschen und verringerte Bildqualität während der Bewegung

Während der Kalibrierung wurden die Bilder unter optimalen Bedingungen aufgenommen – mit hoher Auflösung, stabiler Belichtung und minimalem Rauschen. Sobald sich der Roboter jedoch bewegte, nahm die Bildqualität deutlich ab. Dies lag daran, dass die Kamera während der Bewegung Vibrationen ausgesetzt war und die Rechenleistung des Jetson Nano nicht ausreichte, um eine Echtzeitverarbeitung mit gleichbleibend hoher Qualität sicherzustellen.

Diese Veränderung führte zu mehreren Problemen:

- Die erkannten Bildmerkmale (Feature Points) unterscheiden sich zwischen der Kalibrierung und der realen Anwendung.
- Das Feature Matching wird ungenauer, was zu einer erhöhten Fehlerrate bei der Punktzuordnung führt.

b) Einfluss von variabler Bildrate und Auflösung auf die Kalibrierungsgenauigkeit

Ein weiteres Problem bestand in der unterschiedlichen Bildrate und Auflösung zwischen Kalibrierungs- und Echtzeitbetrieb. Während der Kalibrierung wurden Bilder mit einer stabilen und konstanten Bildrate aufgenommen. In der realen Anwendung

hingegen schwankten diese Werte je nach Systemauslastung, Beleuchtung und Bewegungszustand des Roboters.
Diese Schwankungen hatten folgende Auswirkungen:

- Eine erhöhte Fehlerquote bei der Berechnung der Kameraparameter, insbesondere bei der Bestimmung des Bildzentrums und der Linsenverzerrung (Distortion).
- Veränderungen des Rauschpegels und der Bildverzerrung, bedingt durch unterschiedliche Lichtverhältnisse und variierende Verarbeitungsraten.

Besonders kritisch war die Tatsache, dass der Algorithmus zur Verzerrungskorrektur auf gleichbleibende Kalibrierungswerte angewiesen war. Sobald sich die Aufnahmebedingungen änderten, führte dies dazu, dass die Korrektur weniger präzise arbeitete und somit Fehler in der rekonstruierten 3D-Szene entstanden. Aus diesem Grund wurde beschlossen, die unbearbeiteten Bilder ohne Korrektur der Verzerrung zu verwenden. Dies ermöglichte eine stabilere Verarbeitung, da die Algorithmen direkt mit den Rohbildern arbeiteten und nicht durch inkonsistente Kalibrierungsanpassungen beeinflusst wurden.

c) Verminderte Genauigkeit bei der Merkmalserkennung und Fehlerausbreitung im gesamten System

Neben den direkten Auswirkungen auf die Kameraparameter beeinflusste die schwankende Bildqualität auch die Merkmalserkennung (Keypoint Detection). Während der Kalibrierung wurden die Schlüsselpunkte aus hochwertigen Bildern extrahiert, wodurch eine zuverlässige Erkennung sichergestellt wurde. In der realen Anwendung jedoch führte die verringerte Bildqualität dazu, dass nicht mehr dieselben Merkmale eindeutig erkannt wurden.
Dies hatte mehrere Folgen:

- Die Genauigkeit des Feature Matching zwischen aufeinanderfolgenden Frames nahm ab, was zu fehlerhaften Punktzuordnungen führte.
- Die Berechnung der Kameraposition wurde instabil, da falsche Merkmale in die Positionsbestimmung einflossen.

Zusätzlich breiten sich Fehler, die in einem Teil des Systems entstehen, auf andere Bereiche aus. Wenn beispielsweise Schlüsselpunkte beim Feature Matching nicht korrekt erkannt werden, hat dies direkte Auswirkungen auf die Berechnung der Kameraposition. Diese Fehler summieren sich über aufeinanderfolgende Frames, was dazu führt, dass die Position der Kamera zunehmend ungenau berechnet wird.
Diese Ungenauigkeiten wirken sich anschließend auf weitere Prozesse aus, darunter:

- Triangulation, bei der die 3D-Punkte falsch berechnet werden.
- Clustering, wo Hindernisse möglicherweise nicht korrekt erkannt werden.
- Reprojektion, die zu fehlerhaften Darstellungen im Bild führen kann.

5.1.2 Einfluss der Videoqualität auf die Genauigkeit des Algorithmus

Bereits in den ersten Testläufen wurde deutlich, dass die Qualität des aufgenommenen Videos eine entscheidende Rolle spielt. Um erste Messwerte zu erhalten, wurde die Videoauflösung auf 680×420 Pixel bei einer Bildrate von 21 FPS eingestellt. Doch unter diesen Bedingungen war die Anzahl der erkannten Schlüsselpunkte begrenzt. In einigen Frames zeigte sich sogar ein deutlicher Rückgang der erkannten Merkmale, was in Abbildung 5-27 veranschaulicht wird.

Trotz einer Begrenzung der maximalen Anzahl extrahierter Schlüsselpunkte auf 1500 führte die geringe Videoqualität zu einer erhöhten Fehlerquote im Feature-Matching-Prozess. Viele Schlüsselpunkte wurden falsch zugeordnet, was sich negativ auf die gesamte Verarbeitungskette auswirkte.

Abbildung 5-27: Unterschied zwischen zwei Frames in Bezug auf die Anzahl der erkannten Feature Point

Zusätzlich stellte sich die begrenzte Rechenleistung des Jetson Nano als Engpass heraus. Während der gleichzeitigen Verarbeitung des Videos und der Bewegung des Roboters verschlechterte sich die Bildqualität spürbar, und es kam zu Rauschen in der Beleuchtung. Abbildung 5-28 zeigt den deutlichen Unterschied zwischen einem statischen und einem bewegten Roboterzustand.

Abbildung 5-28: Unterschied in der Bildqualität zwischen einem statischen und einem bewegten Roboterzustand

Optimierung der Aufnahmebedingungen:

Da die bisherigen Aufnahmen keine zufriedenstellende Bildqualität lieferten, wurde entschieden, das Video im Freien bei natürlichem Licht aufzunehmen. Dies führte zu einer Verbesserung der Erkennung der Schlüsselpunkte, da die Beleuchtung gleichmäßiger war und weniger Artefakte durch künstliche Lichtquellen entstanden. Abbildung 5-29 zeigt ein Beispiel eines Frames, der unter diesen Bedingungen erfasst wurde.

Abbildung 5-29: Outside mit 640 x 480

Allerdings zeigte sich, dass die Bildqualität weiterhin nicht stabil genug war. Besonders direktes Sonnenlicht führte zu unerwarteten Problemen: Durch die starke Helligkeit entstanden Reflexionen auf verschiedenen Oberflächen, und das Licht verursachte zusätzliche Bildrauschen, das sich negativ auf die Feature-Erkennung auswirkte. Abbildung 5-30 veranschaulicht die negativen Effekte, die durch starke Lichtquellen entstanden.

Die besten Ergebnisse wurden erzielt, wenn die Videos an bewölkten Tagen oder in schattigen, aber hellen Bereichen aufgenommen wurden. Während starkes Sonnenlicht zu mehr Bildrauschen und Überbelichtung führte, sorgte eine leichte Bewölkung oder Schatten für eine ausgewogene Belichtung, wodurch die Merkmale stabiler erkannt wurden. Dennoch blieb die Bildqualität nicht konstant genug, um eine zuverlässige Ausführung des Algorithmus zu gewährleisten.

Abbildung 5-30: Bild im Sonnenlicht vs. schatten

Anpassung der Videoauflösung und Bildrate:

Da sich durch eine Veränderung der Lichtverhältnisse allein keine zufriedenstellenden Ergebnisse erzielen ließen, wurde die Videoauflösung auf 3280×2464 Pixel erhöht. Diese Maßnahme führte jedoch zu neuen Problemen: Das System fror während der Aufnahme ein, und das Programm stürzte ab. Ursache war die begrenzte Rechenleistung des Jetson Nano, der nicht in der Lage war, gleichzeitig eine hohe Auflösung und eine angemessene Bildrate zu verarbeiten.

Um die Stabilität des Systems zu gewährleisten, wurde die Bildrate auf 6 FPS reduziert. Doch auch diese Änderung hatte unerwartete Konsequenzen. Durch die niedrigere Bildrate wirkte es in der Analyse so, als würde sich der Roboter sehr schnell bewegen. Dies führte zu plötzlichen Änderungen zwischen den Frames, wodurch die Schätzung der Kameraposition ungenauer wurde und sich die Fehlerrate beim Feature Matching erhöhte.

Kompromiss zwischen Bildqualität und Stabilität:

Da die Reduzierung der Bildrate das Problem nicht vollständig löste, wurde eine weitere Anpassung vorgenommen: Die Geschwindigkeit des Roboters wurde verringert, um eine bessere Synchronisation mit der niedrigen Bildrate zu erreichen.

Diese Maßnahme brachte jedoch neue Herausforderungen mit sich:

- Eine Verringerung der Raddrehzahl reduzierte die Motorleistung, wodurch der Roboter bei kleinsten Unebenheiten (z. B. auf Asphalt oder Fliesen) Schwierigkeiten beim Bewegen hatte.

Obwohl durch diese Optimierungen eine Verbesserung bei der Schlüsselpunktextraktion erzielt wurde, war weiterhin eine Feinabstimmung erforderlich, um eine optimale Balance zwischen Bildqualität, Rechenleistung und Stabilität des Algorithmus zu finden.

5.1.3 Einfluss von Umgebungsmerkmalen und Objekten auf die Genauigkeit der 3D-Rekonstruktion

Ein weiteres wichtiges Thema war die Auswahl der Objekte, die in der Szene platziert wurden. Es stellte sich heraus, dass die Qualität der Ergebnisse besser war, wenn mehr Objekte mit komplexeren Formen vorhanden waren. Der Grund dafür liegt darin, dass solche Objekte mehr Ecken und Kanten besitzen, wodurch die Anzahl der erkannten Schlüsselpunkte (Features) steigt. zeigt diesen Unterschied anhand einer Box, einer Tasche und einer Pflanze im Bild.

Abbildung 5-31: Unterschied in der Anzahl der erkannten Schlüsselpunkte in verschieden Objekte

Besonders deutlich wurde dieser Effekt bei einem Regenschirm, der als Testobjekt verwendet wurde. Aufgrund seiner starken Textur und komplexen Oberfläche konnten erheblich mehr Schlüsselpunkte erkannt werden. Dies bestätigt, dass Textur, Kontrast und die Form eines Objekts einen direkten Einfluss auf die Genauigkeit des Feature Matching haben. Abbildung 5-32 veranschaulicht diesen Effekt.

Abbildung 5-32: Regenschirms als Objekt

5.1.4 Herausforderungen bei der Verwendung einer Monokamerasystem anstelle eines Stereosystems

In den meisten SfM-Methoden werden Stereo-Kamerasysteme eingesetzt. Diese arbeiten ähnlich wie das menschliche Sehen, indem der feste Abstand zwischen den beiden Kameras genutzt wird, um die Entfernung eines Punktes direkt zu berechnen.

In diesem Projekt kam jedoch nur eine Monokamera zum Einsatz, wodurch diese Möglichkeit entfiel. Stattdessen wurde die Position der Kamera zu den Zeitpunkten t und t+1 genutzt, wie bereits in Abschnitt 2.3 beschrieben. Da sich die Kamera an der Vorderseite des Roboters befindet und sich dieser in Fahrtrichtung bewegt, sind die Unterschiede zwischen aufeinanderfolgenden Frames sehr gering. Dies führte zu Fehlern bei der Triangulation, selbst wenn viele Feature Matches vorhanden waren. Abbildung 5-33 zeigt den JetBot mit einer Kamera an der Vorderseite, während Abbildung 5-34 die entstehenden Fehler veranschaulicht.

Probleme durch Kameraabstand und Blickwinkel:

Die Qualität der Triangulation hängt stark von der Position und dem Abstand der Kamera ab.

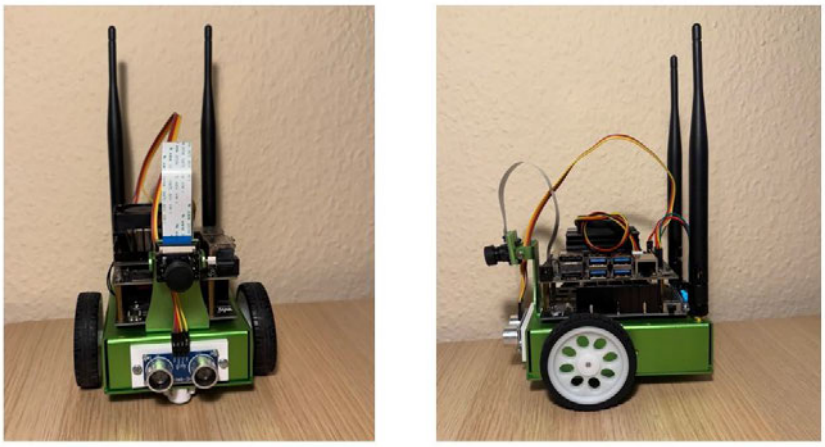

Abbildung 5-33: Jet Bot mit Kamera an der Vorderseite

- Wenn der Abstand zwischen den beiden Kamerapositionen zu gering ist oder ein Punkt zu weit entfernt liegt, verlaufen die Lichtstrahlen fast parallel. Dadurch wird der Schnittpunkt ungenau und instabil, was zu großen Fehlern bei der Berechnung der 3D-Position führt (siehe rechtes Bild in Abbildung 5-34.
- Ist der Abstand zwischen den Kamerapositionen zu groß, wird die Berechnung zwar genauer, aber es wird schwieriger, gemeinsame Merkmale in beiden Frames zu finden. Dies liegt daran, dass sich die Blickwinkel der Kamera zu stark unterscheiden, was die Zuordnung der Punkte erschwert (siehe linkes Bild in Abbildung 5-34).

- Der optimale Abstand liegt zwischen diesen beiden Extremen. Er muss so gewählt werden, dass die Unsicherheit reduziert wird, aber trotzdem genügend gemeinsame Merkmale in beiden Bildern erkannt werden (siehe mittleres Bild in Abbildung 5-34) [Geiger, 2021].

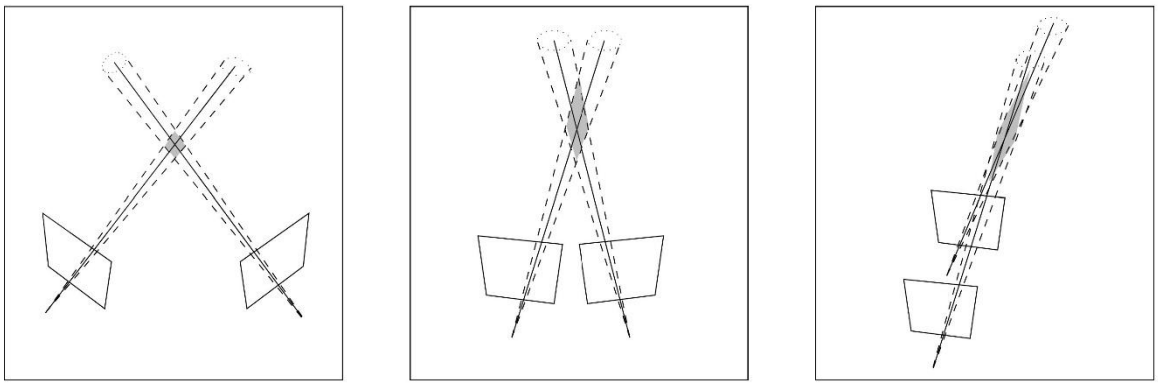

Abbildung 5-34: Einfluss des Kameraabstands auf die Triangulationsgenauigkeit

Ein weiteres Problem entsteht durch die Fisheye-Kamera, die in diesem Projekt verwendet wurde. Die starke Verzerrung (Distortion) dieser Kameraart führt zu zusätzlichen Fehlern und erschwert die präzise Berechnung der Punktkoordinaten.

Lösungsansatz: Optimierung der Kameraposition:

Um diese Probleme zu minimieren, wurde vorgeschlagen, die Kamera von der Vorderseite des Roboters an die Seite zu verlagern. Dadurch würde die Kamera in zwei aufeinanderfolgenden Frames ähnlicher zu einem Stereo-Kamerasystem arbeiten, was die Triangulation deutlich verbessern würde. Abbildung 5-35 zeigt diese alternative Kameraposition.

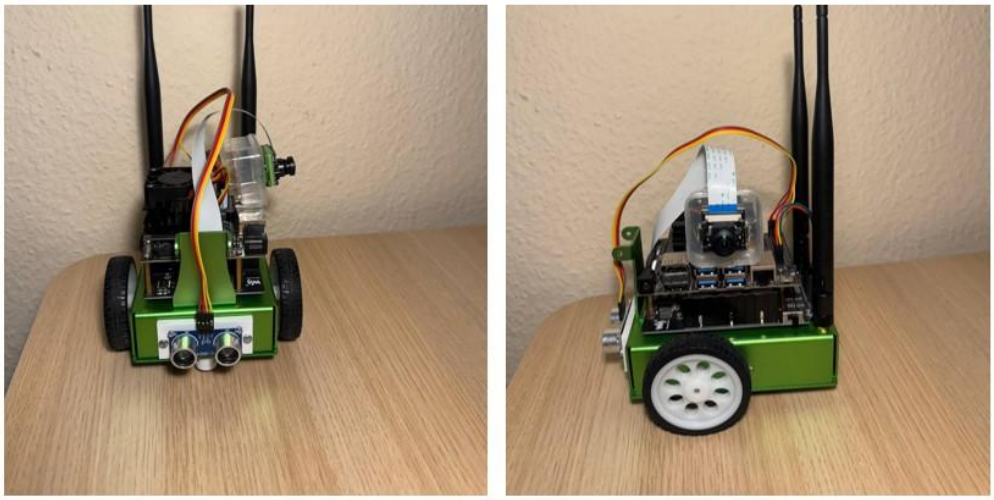

Abbildung 5-35: Jet Bot mit Kamera an der Seite

5.1.5 Überprüfung der Kameraposition und Fehleranalyse

In den vorherigen Abschnitten wurde erklärt, dass die Nutzung einer Monokamera anstelle eines Stereo-Kamerasystems Probleme bei der genauen Positionsbestimmung der Kamera verursachen kann. Eine der größten Schwierigkeiten war, dass die berechnete Kameraposition in jedem Frame nicht immer korrekt war. In einigen Fällen änderte sich die Richtung der Kamera plötzlich und unerwartet.

Da in diesem Projekt eine Fisheye-Kamera verwendet wird und die Verarbeitung direkt auf rohen Bildern ohne Verzerrungskorrektur erfolgt, wurde vermutet, dass die Fehler bei der Berechnung der Kameraposition entstehen. Normalerweise wird die Kamera-Bewegung aus der Essential-Matrix (E-Matrix) berechnet. Diese hängt jedoch von den Kamera-Parametern ab. Da die Verzerrungskorrektur nicht durchgeführt wurde, könnte dies zu großen Fehlern führen.

Einsatz der Fundamental-Matrix als Vorfilter:

Um dieses Problem zu untersuchen, wurde zuerst die Fundamental-Matrix berechnet, bevor die Essential-Matrix verwendet wurde. Der Unterschied ist, dass die Fundamental-Matrix nur die Pixelpositionen der Merkmale nutzt und nicht von den Kamera-Parametern abhängig ist. Dadurch kann sie falsche Punkte (Outliers) herausfiltern und nur gültige Punkte behalten.

Nachdem dieser Filter angewendet wurde, wurde die Essential-Matrix aus der Fundamental-Matrix berechnet, um zu sehen, ob sich die Genauigkeit verbessert.

Test der Essential- und Fundamental-Matrix:

Um sicherzugehen, dass die Matrizen richtig berechnet wurden, wurden mehrere Tests durchgeführt. Zuerst wurde die Determinante beider Matrizen überprüft. Ihr Wert war nahe Null, was zeigte, dass die Struktur der Matrizen korrekt war. Dann wurde die Singulärwertzerlegung (SVD) auf beide Matrizen angewendet. Die Ergebnisse zeigten zwei gleiche Werte und einen dritten Wert nahe Null. Dies bestätigte, dass die mathematische Berechnung der Matrizen korrekt war.

Ein besonders wichtiger Test war die Überprüfung der Singularwerte der Essential-Matrix. Die Werte müssen folgende Bedingungen erfüllen:

- Die ersten beiden Werte sollten etwa 0.707 sein.
- Der dritte Wert sollte fast Null sein.

Die Tests zeigten, dass diese Bedingungen erfüllt waren. Das bedeutet, dass die Essential-Matrix richtig berechnet wurde.

Fehler in der Kamerabewegung (t_{est}) erkennen:

Nachdem die Essential- und Fundamental-Matrix überprüft wurden, wurde die Bewegung der Kamera (t_{est}) analysiert. Dabei traten jedoch Probleme auf:

- In einigen Frames zeigte t_{est} eine Bewegung in die positive X-Richtung, wechselte aber plötzlich in den nächsten Frames in die negative X-Richtung.
- Diese plötzlichen Änderungen deuten darauf hin, dass es ein Problem bei der Berechnung der Kameraposition gibt, das zu falschen und ungenauen Bewegungen führt.

Verbesserung der Kamerabewegung:

Um Rauschen und Fehler zu reduzieren, wurde der t-Vektor normalisiert. Dadurch wurden die Bewegungsänderungen stabiler. Danach wurde die Kamerabewegung im globalen Koordinatensystem aktualisiert. Zusätzlich wurde eine adaptive Alpha-Variable eingeführt, um plötzliche große Änderungen in der Bewegung zu vermeiden.

Falls die Kamera-Translation plötzlich eine große Änderung zeigt, wird sie reduziert, damit die Bewegung ruhiger und stabiler bleibt.

Diese Optimierungen verbesserten die Kamerabewegung, wie in Abbildung 5-36 zu sehen ist. Im Vergleich zur ursprünglichen Bewegung in Abbildung 5-37 sind nun weniger Sprünge und plötzliche Richtungswechsel sichtbar.

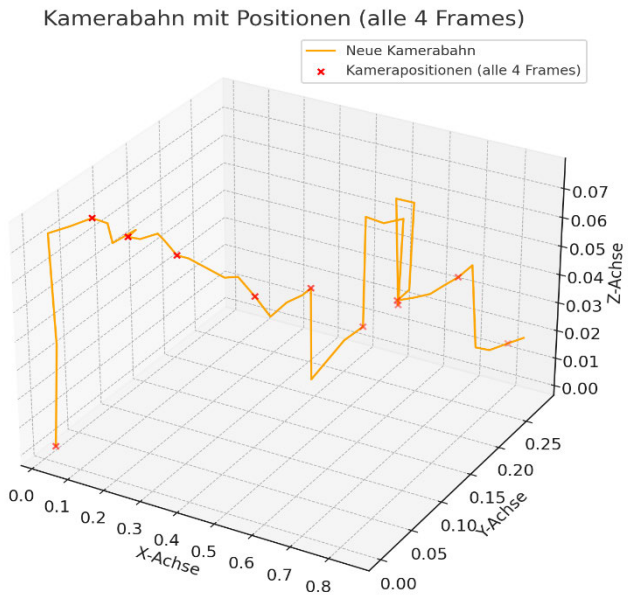

Abbildung 5-36: Verbesserten die Kamerabewegung

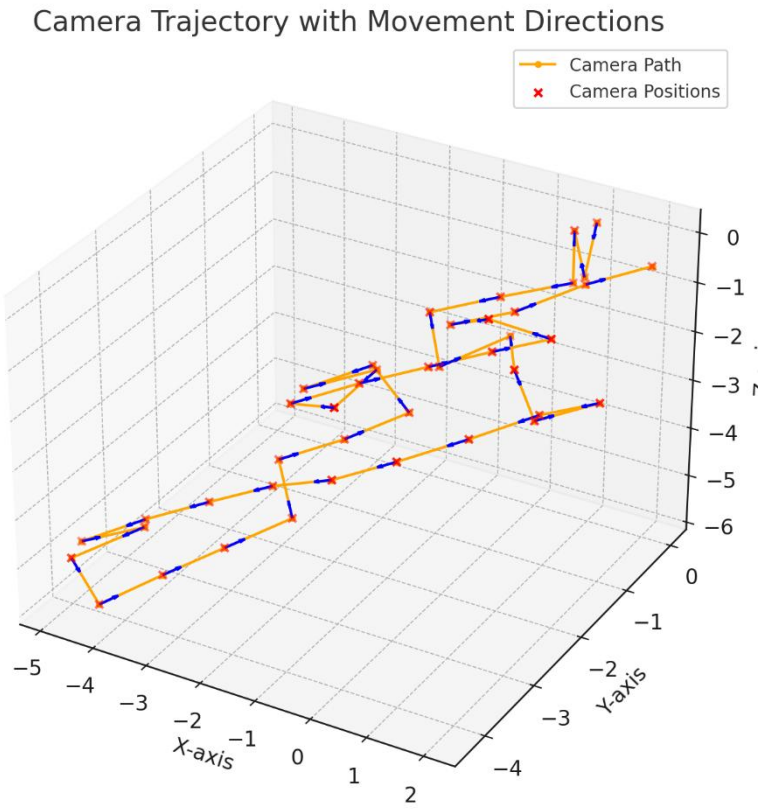

Abbildung 5-37: Kamerabahn mit keine Optimierung

In der zweidimensionalen Darstellung(Der X-Y-Ebene) der Kamerabewegung ist die Verbesserung noch klarer zu sehen. Durch die Korrekturen ist die berechnete Bewegung stabiler und realistischer geworden. Dies wird deutlich in Abbildung 4-23 in Abschnitt 4.2.6. Trotz dieser Verbesserungen ist die Kamerabewegung noch nicht perfekt, was Auswirkungen auf die Hinderniserkennung hat.

5.1.6 Fehleranalyse bei der 3D-Rekonstruktion und Verbesserung der Punktwolke

Obwohl das Problem der Kameraposition nicht vollständig gelöst werden konnte, wurde die Analyse der Rekonstruktion fortgesetzt. Der nächste wichtige Schritt bestand darin, die 3D-Punkte aus den erkannten Merkmalen zu berechnen und deren Qualität zu überprüfen. Die Genauigkeit dieser Punktwolke beeinflusst direkt die spätere Hinderniserkennung, weshalb es notwendig war, mögliche Fehlerquellen zu identifizieren und zu minimieren.

Während der Analyse der rekonstruierten Punktwolke fiel auf, dass einige Punkte an geometrisch nicht sinnvollen Positionen lagen. In manchen Fällen befanden sich Punkte sogar hinter der Kamera, was darauf hindeutete, dass Fehler während der Triangulation auftraten.

Eine genauere Untersuchung zeigte, dass diese Fehler nicht zufällig auftraten, sondern mit mehreren Faktoren zusammenhingen, die die Qualität der Rekonstruktion beeinflussten

Einer der Hauptgründe lag in der Feature-Erkennung und dem Feature-Matching. Wenn in diesem Schritt Rauschen oder fehlerhafte Punkte enthalten waren, verstärkten sich die Fehler während der Triangulation. Selbst wenn die meisten erkannten Merkmale korrekt waren, führten einzelne fehlerhafte Zuordnungen dazu, dass bestimmte Punkte in der 3D-Rekonstruktion an unrealistischen Positionen erschienen. Um diese Fehler zu reduzieren, wurde die Bildqualität weiter verbessert und zusätzlich eine Filterung ungültiger Punkte vorgenommen. Dabei wurden maskierte Bereiche definiert, um sicherzustellen, dass nur zuverlässige Features in die Berechnung einflossen. Diese Optimierungen trugen dazu bei, dass die meisten Fehler in der Feature-Erkennung minimiert wurden.

Trotz dieser Verbesserungen blieben jedoch weiterhin Fehler in der Punktwolke bestehen. Besonders problematisch war der geringe Winkel zwischen aufeinanderfolgenden Frames. Da sich die Kamera in vielen Fällen nur minimal bewegte, überlappten sich die Bilder stark, wodurch sich die Perspektive nur geringfügig änderte. Dies hatte zur Folge, dass die Triangulation instabil wurde, selbst wenn eine große Anzahl von Feature Matches vorhanden war. Um diese Instabilität zu reduzieren, wurde die Kameraposition angepasst. Statt die Kamera direkt in Fahrtrichtung auszurichten, wurde sie leicht zur Seite versetzt. Dadurch änderte sich die Perspektive zwischen den Frames deutlicher, was eine stabilere Triangulation ermöglichte. Wie bereits in Abbildung 5-35 gezeigt wurde, führte diese Änderung zu einer spürbaren Verbesserung der Punktrekonstruktion.

Neben der Anpassung der Kameraperspektive wurden auch die rekonstruierten 3D-Punkte gefiltert, um die Qualität der Punktwolke weiter zu verbessern. Punkte, die geometrisch nicht sinnvoll waren, wurden entfernt, um Fehler in der späteren Berechnung zu vermeiden. Hierbei wurden insbesondere drei Kriterien berücksichtigt:

- Punkte mit einem negativen Z-Wert wurden ausgeschlossen, da sie sich hinter der Kamera befanden und somit nicht korrekt rekonstruiert wurden.
- Punkte, die mehr als zwei Meter von der Kamera entfernt waren, wurden entfernt, da sie für die Hinderniserkennung nicht relevant waren.
- Punkte, die sich unterhalb der Kamerahöhe befanden, wurden nicht weiter berücksichtigt, da sie nicht als Hindernisse klassifiziert wurden.

Durch diese Filterung wurde die Punktwolke insgesamt stabiler und präziser. Die Kameraposition konnte besser geschätzt werden, was sich wiederum positiv auf die spätere Hinderniserkennung auswirkte. Dennoch blieb die Positionsschätzung der Kamera nicht vollkommen fehlerfrei. Die Einschränkungen einer Monokamera führten dazu, dass die Tiefeninformation nicht so präzise bestimmt werden konnte wie in einem Stereo-Kamerasystem. Zusätzlich begrenzte die Rechenleistung des Jetson Nano die Genauigkeit der Echtzeit-Berechnung.

Diese verbleibenden Fehler hatten auch Auswirkungen auf die Hinderniserkennung. In einigen Frames wurden keine Cluster erkannt, da die Positionsberechnung nicht exakt genug war. In anderen Fällen wurden Hindernisse leicht versetzt dargestellt. Wie in Abbildung 5-38 zu sehen ist, wurden Objekte wie ein Regenschirm und eine Tasche zwar korrekt erkannt, jedoch mit einer leichten Verschiebung dargestellt.

Abbildung 5-38: Fehlplatzierte Objekterkennung

Obwohl durch umfangreiche Tests und verschiedene Optimierungen eine Verbesserung erreicht werden konnte, war es mit der vorhandenen Hardware nicht möglich, die gewünschte Präzision vollständig zu erreichen.

5.2 Anwendbarkeit

Die SfM-Methode mit einer Monokamera hat großes Potenzial in verschiedenen Bereichen. Allerdings gibt es einige technische Herausforderungen, die die Anwendung in der realen Welt einschränken.

5.2.1 Potenzielle Anwendungen

Die 3D-Rekonstruktion mit einer einzelnen Kamera kann in kostengünstigen und leichten Systemen nützlich sein. Wichtige Anwendungsbereiche sind:

1. Einsatz in autonomen Robotern zur Hinderniserkennung

In robotischen Systemen ist eine präzise Hinderniserkennung entscheidend, um Kollisionen zu vermeiden und eine sichere Navigation zu ermöglichen. Die SfM-Methode mit einer einzelnen Kamera kann als kostengünstige Alternative zu teuren Sensoren wie LiDAR eingesetzt werden. Der Hauptvorteil dieser Methode ist die Reduzierung der Hardwarekosten.

Allerdings gibt es eine zentrale Herausforderung: Die Verwendung einer einzelnen Kamera erschwert die Skalierung der Entfernungen. In Stereo-Systemen dient der feste Abstand zwischen zwei Kameras als bekannte Größe für die Berechnung der Tiefeninformationen. Bei SfM mit einer Monokamera wird die Entfernung hingegen durch die Bewegung der Kamera geschätzt, was zu hohen Fehlern führen kann. Deshalb funktioniert diese Methode besser in strukturierten Umgebungen mit stabilen Bewegungsbedingungen.

2. 3D-Kartierung in Innen- und Außenbereichen

In Bereichen wie digitale Kartierung, Architektur und Archäologie besteht ein wachsender Bedarf an 3D-Scans und Umgebungserfassung. Viele der derzeit verwendeten Methoden basieren auf teuren Geräten wie Laserscannern oder Stereo-Kamerasystemen. Die SfM-Technologie mit einer einzelnen Kamera bietet hier eine kostengünstige Alternative, um 3D-Modelle mit einfacher Hardware zu erstellen.

Diese Methode liefert besonders gute Ergebnisse, wenn die Umgebung viele strukturelle Merkmale und Texturen enthält, wie beispielsweise Gebäudefassaden oder unregelmäßige Oberflächen. In Szenen mit wenigen Kontrasten oder glatten Flächen kann die Rekonstruktion jedoch fehlerhaft sein.

3. Einsatz in industriellen Robotern zur Umgebungsanalyse

In Industrie- und Fertigungsumgebungen benötigen Roboter visuelle Systeme, um Hindernisse zu erkennen, ihren Pfad anzupassen oder Objekte zu identifizieren. Wenn der Einsatz von LiDAR oder Stereo-Kameras zu teuer oder unpraktisch ist, kann eine einzelne Kamera mit SfM-Algorithmen als kosteneffiziente Lösung genutzt werden.

Allerdings gibt es auch hier einige Herausforderungen:

- Starke Lichtreflexionen von metallischen oder glänzenden Oberflächen können die Feature-Erkennung erschweren.
- Variable Lichtverhältnisse in Fabrikhallen können die Algorithmen negativ beeinflussen.
- Schwierigkeiten bei der Erkennung von beweglichen Objekten, da die SfM-Methode auf die Bewegung der Kamera angewiesen ist und die Position von beweglichen Objekten schwer zu bestimmen ist.

5.2.2 Einschränkungen und Herausforderungen

Trotz des hohen Potenzials gibt es einige wesentliche Einschränkungen, die den Einsatz der Methode in bestimmten Szenarien begrenzen. Die wichtigsten Herausforderungen sind:

1. Abhängigkeit von Lichtverhältnissen und Bildqualität

Ein Hauptproblem von SfM mit einer einzelnen Kamera ist die starke Abhängigkeit von der Beleuchtung. In Umgebungen mit schwachem Licht werden weniger Bildmerkmale erkannt, wodurch die Genauigkeit der 3D-Rekonstruktion deutlich abnimmt.

Auch in Szenen mit sehr starken Lichtquellen oder direkter Sonneneinstrahlung kann es zu störenden Reflexionen und Bildrauschen kommen, was die Feature-Matching-Genauigkeit reduziert. Diese Herausforderung ist besonders in Outdoor- und Industrieumgebungen problematisch.

2. Skalierungsprobleme und Tiefengenauigkeit

In Stereo-Systemen ist der Abstand zwischen den Kameras eine bekannte Größe, die als Referenz für die Tiefenberechnung dient. Bei SfM mit einer einzelnen Kamera muss die Skalierung ausschließlich durch die eigene Bewegung der Kamera geschätzt werden. Dadurch können erhebliche Fehler bei der Tiefenbestimmung und der Größenwahrnehmung von Objekten auftreten.

Mögliche Folgen:

Objekte erscheinen näher oder weiter entfernt als sie tatsächlich sind.

Wiederholende Muster oder glatte Oberflächen können zu fehlerhaften Rekonstruktionen führen.

3. Einfluss von Bildrauschen und Hardwarebeschränkungen

Falls die aufgenommenen Bilder eine niedrige Qualität oder hohes Bildrauschen aufweisen, kann das Feature Matching unzuverlässig werden, was die Fehler in der 3D-Rekonstruktion verstärkt. Dies ist besonders in Systemen mit niedriger Kameraauflösung oder begrenzter Rechenleistung (z. B. bei Jetson Nano) problematisch.

Zusätzlich kann die Bewegung des Roboters zu leichten Vibrationen führen, wodurch sich Bewegungsunschärfe in den Bildern ergeben. Dies kann die Präzision der Algorithmen weiter verringern.

6 Fazit und Ausblick

Trotz der bestehenden Herausforderungen und Schwächen dieses Projekts führte die gewählte Struktur zur Zielerreichung zu einem klaren Ergebnis. Die durchgeführten Tests und Optimierungen zeigten sowohl die Möglichkeiten als auch die Grenzen der entwickelten Methode.

Im Folgenden werden die Ergebnisse zusammengefasst, bestehende Probleme und Einschränkungen erläutert sowie ein Ausblick auf zukünftige Verbesserungen gegeben.

6.1 Fazit

Diese Arbeit befasste sich mit der Implementierung der SfM-Methode zur Hinderniserkennung mithilfe einer einzelnen Fischaugenkamera. Durch die Kombination von Merkmalsdetektion, Epipolargeometrie, Kamerabewegungsschätzung und Triangulation konnte eine 3D-Rekonstruktion der Umgebung erstellt werden, die zur Identifikation von Hindernissen genutzt wurde.

Die entwickelte Methode wurde anhand einer aufgezeichneten Videosequenz getestet, die mit einem JetBot während der Bewegung aufgenommen wurde. Während der Experimente zeigte sich, dass die begrenzte Rechenleistung des Jetson Nano sowohl die Videoaufnahme als auch die Verarbeitung einschränkte. Hochwertige Videoaufnahmen waren mit mehr als 6 FPS nicht möglich, da das System andernfalls überlastet wurde und nicht mehr stabil lief. Gleichzeitig stellte die Berechnung der SfM-Pipeline eine hohe Anforderung an die Rechenleistung, sodass eine Echtzeitverarbeitung nicht möglich war.

Eine Reduzierung der Bildrate unter 6 FPS hätte die Systemstabilität verbessert, hätte aber auch zu weiteren Berechnungsfehlern geführt. Durch die geringere Anzahl an Frames wären die Unterschiede zwischen aufeinanderfolgenden Bildern größer gewesen, was die Schätzung der Kamerabewegung, die Triangulation und schließlich die Hinderniserkennung negativ beeinflusst hätte.

Die Alternative, die Auflösung der Kamera zu reduzieren, hätte ebenfalls Nachteile mit sich gebracht. Durch die niedrigere Bildqualität wäre das Bildrauschen erhöht worden, was zu Unschärfen geführt und die Feature-Erkennung und das Matching erschwert hätte.

Die Ergebnisse zeigen daher, dass die Methode mit dem Jetson Nano in Echtzeit nicht umsetzbar ist. Eine Echtzeitverarbeitung wäre nur durch den Einsatz einer leistungsstärkeren Hardware oder eine Reduzierung der zu verarbeitenden Datenmenge möglich.

Letzteres würde jedoch die Genauigkeit weiter verschlechtern, weshalb es keine geeignete Lösung darstellt.

Zusätzlich zeigten die finalen Ergebnisse, dass weiterhin wesentliche Herausforderungen bei der genauen Bestimmung der Kameraposition bestehen. Trotz mehrfacher Optimierungen blieben Fehler in der Positionsschätzung, die sich direkt auf die Hinderniserkennung auswirkten. Dies führte zu zwei Hauptproblemen:

1. Bildrauschen wurde in einigen Fällen fälschlicherweise als Hindernis erkannt.
2. Tatsächliche Hindernisse wurden teilweise nicht korrekt detektiert.

Diese Einschränkungen verdeutlichen, dass die Methode noch weiterentwickelt werden muss, um zuverlässigere und stabilere Ergebnisse zu erzielen.

6.2 Ausblick

Obwohl die entwickelte Methode vielversprechende Ergebnisse lieferte, zeigte sich, dass eine Echtzeitverarbeitung mit dem Jetson Nano nicht möglich ist. Die geringe Rechenleistung führte dazu, dass entweder die Bildrate reduziert oder die Auflösung verringert werden musste, was jeweils die Genauigkeit der Rekonstruktion beeinträchtigte.

Eine naheliegende Verbesserung wäre daher der Einsatz von leistungsstärkeren Plattformen wie Jetson Xavier oder einer GPU-basierten Implementierung. Mit einer solchen Hardware könnte das System in Echtzeit ausgeführt werden, ohne dass die Bildqualität oder die Berechnungsschritte reduziert werden müssen.

Zusätzlich könnten auch alternative Algorithmen getestet werden, um die Stabilität der Punktzuordnung zu verbessern. Während klassische Methoden wie SIFT bereits gute Ergebnisse lieferten, könnten moderne Verfahren wie SuperPoint oder tiefenlernende Modelle die Merkmalserkennung robuster gestalten. Gleichzeitig könnten fortgeschrittene Filteralgorithmen zur Rauschunterdrückung dazu beitragen, dass fehlerhafte Punktzuweisungen minimiert werden und die Punktwolke insgesamt stabiler wird.

Ein weiteres Problem, das in dieser Arbeit identifiziert wurde, ist die Notwendigkeit, Fischaugenverzerrungen zu korrigieren, bevor die SfM-Pipeline angewendet wird. Eine interessante Alternative wäre die Nutzung von COLMAP, das nicht-lineare Fischaugenmodelle wie Equidistant und Kannala-Brandt unterstützt. Dies würde es ermöglichen, die Rekonstruktion ohne vorherige Verzerrungskorrektur durchzuführen, wodurch Skalierungsfehler reduziert und die Berechnung der Kameraposition stabiler werden könnte.

Zusammenfassend lässt sich sagen, dass die hier entwickelte Methode mit stärkeren Hardware-Komponenten und optimierten Algorithmen weiter verbessert werden könnte. Während eine Echtzeitverarbeitung mit dem Jetson Nano nicht realisierbar ist, könnten

durch leistungsfähigere Systeme oder effizientere Berechnungsmodelle genauere und stabilere Ergebnisse erzielt werden.

7 Zusammenfassung

Diese Arbeit untersuchte die Anwendung von Structure from Motion (SfM) zur Hinderniserkennung mit einer Fischaugenkamera. Ziel war es, eine kostengünstige Alternative zu hochentwickelten Sensoren wie LiDAR für mobile Roboter bereitzustellen.

Zunächst wurde die verwendete Kamera kalibriert und eine Videoaufnahme der Umgebung durchgeführt. Anschließend wurden Bildmerkmale extrahiert und mit Methoden wie SIFT und BFMatcher über aufeinanderfolgende Frames hinweg abgeglichen. Danach wurde mit Epipolargeometrie und Triangulation eine 3D-Rekonstruktion der Umgebung erstellt und der Kamerapfad geschätzt. Abschließend wurden Clustering-Algorithmen zur Trennung von Hindernissen und Rauschen angewendet.

Die während des Prozesses gesammelten Daten wurden systematisch gespeichert. Die Feature-Matches und korrespondierenden Keypoints wurden in einer Datei namens Image_matches.json gespeichert, während die berechneten Kamerapositionen in Camera_Pose.txt dokumentiert wurden. Zudem wurden die 3D-Punkte jeder einzelnen Frame separat als .ply-Dateien in dem Ordner savePly abgelegt.

Während der Umsetzung traten mehrere Herausforderungen auf:

- Geringe Genauigkeit bei der Kamerapositionierung: Da nur eine einzelne Kamera verwendet wurde, war die Tiefenschätzung und genaue Positionierung der Kamera fehleranfällig.
- Falsche Hinderniserkennung: In einigen Fällen wurden Rauschpunkte als Hindernisse erkannt, während echte Hindernisse nicht immer korrekt identifiziert wurden.
- Begrenzte Rechenkapazität: Die Jetson Nano-Plattform konnte hochauflösende Daten nicht effizient genug verarbeiten, was zu reduzierter Bildrate und geringerer Verarbeitungsqualität führte.
- Lichteinflüsse: Starke Beleuchtungsänderungen beeinflussten die Feature-Erkennung und erhöhten die Fehlerquote bei der Punktzuordnung.

Zur Verbesserung der Methode wurde vorgeschlagen, zusätzliche Sensoren (z. B. LiDAR) zu integrieren, Bundle Adjustment zur Optimierung zu verwenden, leistungsstärkere Recheneinheiten einzusetzen und COLMAP für Fischaugenbilder ohne Entzerrung zu nutzen.

Insgesamt zeigt diese Arbeit, dass SfM eine vielversprechende Technik zur kamerabasierten Hinderniserkennung ist, jedoch weitere Optimierungen erforderlich sind, um die Genauigkeit und Stabilität zu erhöhen.

8 Literaturverzeichnis

[Chizhova, 2019] Chizhova, Maria. 2019. Virtuelle 3D-Rekonstruktion von zerstörten russisch-orthodoxen Kirchen aus unvollständigen Punktwolken. 2019.

[Chunxiao Wang, 2019] Chunxiao Wang, Min Ji, Jian Wang, Wei Wen, Ting Li, Yong Sun. 2019. An Improved DBSCAN Method for LiDAR Data Segmentation with Automatic Eps Estimation. 2019, Sensors.

[David Millán Escrivá, 2019] David Millán Escrivá, Prateek Joshi, Vinícius G. Mendonça, Roy Shilkrot. 2019. Explore Structure from Motion with the SfM Module. Building Computer Vision Projects with OpenCV 4 and C++. 2019, S. 321-342.

[Fabrica, 2025] Creative Fabrica. [Online] 07. 02 2025. https://www.creativefabrica.com/de/product/camera-logo-design-inspirations/.

[Geeksforgeeks, 2024] Geeksforgeeks. [Online] 16. 12 2024. https://www.geeksforgeeks.org/calibratecamera-opencv-in-python/.

[Geiger, 2021] Geiger, Prof. Dr.-Ing. Andreas. 2021. Computer Vision-Image Formation. University of Tübingen / MPI-IS : s.n., 2021

[Guangli Jiang, 2015] A 127 fps in full HD accelerator based on optimized AKAZE with Efficiency and Effectiveness for Image Feature Extractio. Guangli Jiang, Leibo Liu, Wenping Zhu, Shouyi Yin, and Shaojun We. 2015. 2015. 52nd ACM/EDAC/IEEE Design Automation Conference (DAC).

[Herbert Bay, 2006] SURF: Speeded Up Robust Features. Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. 2006. 2006.

[Hitam, et al., 2013] Mixture contrast limited adaptive histogram equalization for underwater image enhancement. Hitam, Muhammad Suzuri, et al. 2013. Sousse, Tunisia : IEEE, 2013. 10.1109/ICCAT.2013.6522017.

[JetBot, 2025] jetbot. [Online] 28. 01 2025. https://jetbot.org/master/index.html.

[Johnson, et al., 2014] Rapid mapping of ultrafi ne fault zone topography with structure from motion. Johnson, Kendra, et al. 2014. 2014, Geosphere.

[Jonathan Courbon, 2012] Jonathan Courbon, Youcef Mezouar ∗ and Philippe Martinet. 2012. Evaluation of the Unified Model of the Sphere for Fisheye Cameras in Robotic Applications. Advanced Robotics. July 2012.

[MARKER, 2025] ONLINE LOGO MARKER. [Online] 07. 02 2025. https://www.onlinelogomaker.com/blog/5-steps-create-camera-ready-video-production-logo/.

[Mathworks, 2024] Mathworks. [Online] 12 2024. https://de.mathworks.com/help/vision/ug/using-the-single-camera-calibrator-app.html.

[Mengcheng Song, 2018] Robust 3D Reconstruction with Omni-directional Camera based on Structure from Motion. Mengcheng Song, Hiroshi Watanabe, Okubo, Shinjuku-ku, Junich Hara, Shimo-Imaizumi, Ebina, Kanagawa. 2018. Thailand : s.n., 2018. International Workshop on Advanced Image Technology (IWAIT).

[OpenCV, 2025] OpenCV. 2025. OpenCV. [Online] 09. 01 2025. https://docs.opencv.org/4.x/d1/d89/tutorial_py_orb.html.

[Szeliski, 2022] Szeliski, Richard. 2022. Computer Vision - Algorithms and Applications 2nd Edition. 2022.

[Varun Ravi Kumar, 2023] Surround-View Fisheye Camera Perception for Automated Driving: Overview, Survey & Challenges. Varun Ravi Kumar, Ciarán Eising , Christian Witt , Senthil Yogamani. 2023. 2023, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, S. 3638 - 3659

[Waveshare, 2025] Waveshare. [Online] 28. 01 2025. https://www.waveshare.com/wiki/IMX219-160_Camera.

[Yi Hou, 2020] Fisheye Images Correction Based on Different Angle of Views. Yi Hou, LiPi Niu,YanMing Zhao,ShanZhen Lan. 2020. s.l. : 978-1-7281-5244-8/20/$31.00 ©2020 IEEE, 2020. International Information Technology and Artificial Intelligence Conference(ITAIC).

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Mittweida, den 10.03.2025

Mahsa Khabbazi